A FRAMEWORK FOR UNDERSTANDING THE VISION, GOALS, INSTRUMENTS AND USES OF SOFTWARE SERVICES

Martin Henkel
Department of Computer and Systems Sciences, Stockholm University and Royal Institute of Technology, Stockholm, Sweden

Software services have been suggested for use in several areas such as business-to-business communication, system integration and as an enabler for virtual enterprises. However, it has not been clear if the use of services in these different areas is guided by a common vision, and if they share goals and technical background. By introducing a framework that relates the vision and goals of software services to different categories of service use, this work presents an overview of the instruments needed for the development of software services. The framework spans across four interrelated parts; the vision, goals, instruments and categories of service use. Firstly a single vision for software services is proposed. This vision is then divided into the three goals of modularity, integration and discovery. Furthermore, the high-level instruments needed to achieve each goal are presented. The use of services is classified into four categories depending on reach and extent i.e.: point wise-internal, point wise-external, infrastructure-internal and infrastructure-external use. The framework presented in this paper shows how these categories of service utilise the defined instruments. As an example of framework use, the instruments needed to create a process-centric service infrastructure for intra-enterprise use is proposed

Keywords: *Service oriented computing, e-services, software services*

1. Introduction

The terms service, e-service and Web service have begun to be promoted in product launches as well as academic and industrial research. Standards have been agreed upon to define service technologies such as SOAP, WSDL and UDDI (Gudin *et al.*, 2002; Chinnici *et al.*, 2003, Bellwood *et al.*, 2002). Drafts of standards are quickly incorporated in new and existing products. Research has extended the service domain with additions such as repositories and agent-based service brokers. Clearly, services are drawing a lot of attention.

The vision driving the development of software-services is somewhat ambiguous and not always obvious. The vendors of Enterprise Application Integration (EAI) servers tout the use of services as a way to reuse and connect existing applications. In Business-to-Business communication (B2B) services are used as enablers of a global, open e-market. Identifying and understanding the vision of software services is important as a first step towards identifying what is required to enable the application of software-services.

Starting with the vision of software services this paper identifies three basic goals for software services. These goals are in effect drivers for the development of new products as well as being the impetus for initiating research into services. To achieve these goals, a set of instruments can be utilised.

Instruments can be both of technical nature such as a standard technical infrastructure, or of a more prescriptive nature such as the different ways to decompose and compose software services. The instruments can be used to solve common problems, such as providing a way to modularise a system based on services. The instruments required to fulfil each of the three goals are identified and described in this paper.

When using software services, one can strive towards one or more of the identified goals. Striving towards several of the defined goals demands the utilisation of several instruments. By classifying uses of services into categories it is possible to propose a set of instruments that, for each category, is suitable for goal fulfilment.

By identifying vision, goals, instruments, categories of uses and their interrelation it is possible to get a structured overview of the uses of software services, along with a set of instruments that are required to successfully use software services. An overview of the parts in the framework is presented in Figure 1.

The paper is structured as follows: In the first section the term service is defined and services are put in context by comparing services and components. In the next section a vision for software services is proposed. This vision is broad by purpose, so that it encompasses current research issues and industry trends. In the following section the vision is divided into three goals, each characterizing a typical issue within the field of software services. Then, the instruments that are needed to fulfil these goals are presented. These instruments represent a range of differentiated solutions from both academic research and existing commercial products. In the next section uses of services are categorized, the chosen criteria of categorization are situations of service use that greatly impact the need for instruments. Furthermore, each use category is related to the defined goals and instruments. The paper ends with an example of how to use the framework.

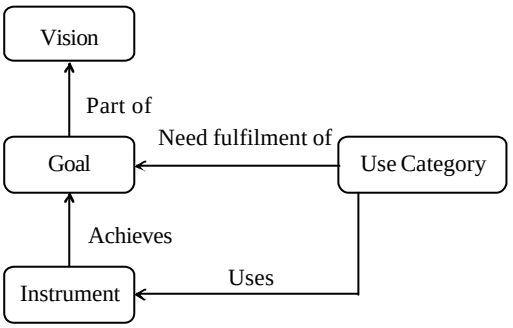

Fig. 1 Overview of the framework

2. Definitions

Disregarding the software domain, a service can be defined as an "act or performance offered by one party to another" (Lovelock *et al.*, 1996). From a business perspective, a service can be viewed as a process that produces value for the consumer of the service. The parties involved in this definition are not limited to software, i.e. a party can be either human or software. Regardless of whether a service is used and provided by humans or software, a service does not result in any ownership (Bennett *et al.*, 2001b; Dumas *et al*., 2001). This means that a service is an offering that can be used, compared to a commodity that can be owned and bought.

An application service provider (ASP) is an example of a service where the provider is a software system, and the consumer is a human.

In this paper, the term *software* service is used to denote a service that can be used by software systems, or speaking in software terms, a service can be invoked from other software systems. In this definition both parties are software programs that communicate, potentially over a network. Just as for non-software services the consumer and provider of a software service can belong to separate enterprises. The remainder of this paper will focus on software services.

2.1. Services and Components

Services share many properties with components. Firstly, both services and components offer a well-defined interface that hides the details of implementation. Secondly, both services and components are modular so that they can be combined to form new components and services, respectively. The important difference is that a service is a run-time entity offering a service interface while a component is a physical/binary entity that needs to be installed before use. Users of components need to buy them, install them and then use them, while a service user simply finds the service and uses it (Herzum, 2001). This difference implies that the provider of a service is responsible for it, even in run-time, while the provider of a component does not necessarily have any responsibility and control of the run-time environment.

Component technologies like COM+ (Platt, 1999) and EJB (Monson-Haefel, 2001) are useful for building software services. An EJB component for example, installed and running on a server can be offered as a service. It can be said that components are developed, but they should be provided as services (Allen and Frost 1998). This means that components are primarily used as building blocks in the development phase, whereas services exist in run-time.

Not all components of a system should be provided as services since the user of an application (software or human) is seldom interested in the inner workings of the system. A black-box view is preferred. By selecting the components to be provided as services, an uncluttered interface to the system can be obtained. Thus, the exposed services are commonly (but not necessarily) more coarsely grained than the components that together implement the service.

Due to the run-time properties of services, it is natural to focus on the service interface and on run-time communication when discussing services. When one on the other hand is discussing components, the development, deployment and run-time environments come into focus.

There are then four properties of components and services that can be compared. The comparison based on these properties is summarized in Table 1.

Table 1 Components and services compared

Property	Components	Services
Use	Buy-install-use	Use-pay
Responsibility of provider	Construction and delivery	Run-time service availability
Granularity	Application building block	Application interface
Focus of interest	Implementation	Communication

3. Service Vision

A long-term vision behind software services is to *enable a global market of well-structured, well-defined and modular services that are accessible for consumers around the world*. In this vision, services are highly modular resources that can be composed to form new systems and services (Piccinelli and Stammers, 2001; Péraire and Coleman, 2000; Bennet *et al.*, 2001b). Furthermore, an important part of the vision is that consumers should be able to select and combine services offered by providers that might reside on different geographical locations and be provided by different organisations. The vision includes that services should be platform and program language independent (Fremantle *et al.*, 2002).

By using services it is possible for organisations to out-source complex functionality to other organisations (Harumi, 2000), thereby forming virtual organisations or "networked enterprises" (Yang, 2001).

4. Service Goals

The vision of software services can be divided into sub-visions, or goals. Firstly, the functionality needs to be divided into modular resources. Secondly, these resources must use an interconnection format that makes them globally useable. Thirdly, consumers must be able to discover and select the service to use.

Thus, the vision for software services can be divided into three goals; modular resources, global interfaces and dynamic discovery.

Modular resources: This goal is to provide each resource as a service, instead of encapsulating and tying them into monolithic applications. The major benefit of creating modular services is that new systems can be built by combining existing services. This makes it easier to implement changes derived from evolving requirements. A typical example is the development of new business processes that need IT support. The notion of services as building blocks for systems presented by Wald and Stammers (2001) is an example of how to benefit from services as modular resources.

Global interfaces: The goal of services as global interfaces means that services should define and be used as a standard for connecting software systems. This goal includes removing interconnection boundaries such as differences in platform, program languages, and geographical location. As a further step the differences in the way organisations do business that hinder communication can be overcome by defining a standard way to handle business agreements for services. Standardisation and description of interfaces and protocols are essential requirements of this goal. Work being conducted on ebXML and Web service technologies is an example of steps towards achieving this goal (Webber and Kotok, 2001).

Dynamic discovery: At least a partial fulfilment of the goal of modular resources and global interfaces are fundamental for creating a global market of well-structured, well-defined services that are accessible for consumers around the world. The goal of dynamic discovery of services consists of this market, together with consumers who can actively search the market for suitable services offered by providers. Consumers can select the provider who offers the best price, performance etc, etc. By constantly searching the market for new providers the consumer can change provider, giving optimal performance at every moment. The architecture for dynamic evolving systems based on services presented by Bennet *et al.* (2001a) is an example of the implementation of this goal.

5. Service Instruments

To achieve the defined goals, development and provisioning of services can utilise a set of instruments. The following sections state the requirements for each goal and describe a set of instruments that can be utilised to achieve each goal.

5.1. Instruments for Modular Resources

Representing a major part of an organisation's resources as modular services requires that the functionality can be decomposed into services. Decomposition into services should be done in a way that enables consumers to freely combine services, without the risk of duplicating functionality, and without the need for excessive customisation to make the services fit together. Constructing a set of services that are meant to be combined requires stringent decomposition criteria. A set of high-level criteria for decomposition is thus valuable instruments when striving for the goal of modular resources.

A major criterion for decomposition is if the services should be aimed at either supporting a technical aspect or an organisational aspect. Technical aspects include (database) transaction handling, queues and events, persistence etc. Organisational aspects include business processes and tasks. A service built to support a technical aspect is called a horizontal service (Stahl, 2002), whereas services that support organisational aspects are called vertical services. Using vertical and horizontal decompositions is a useful instrument for the development of modular services.

Horizontal services are independent on the business domain. The main benefit of horizontal services is that they can be used by several applications in different domains, thereby providing a technical framework for system development. Common horizontal services include error and transaction handling, asynchronous event handling and security (Brown, 2000). Horizontal services such as transaction co-ordination and events are commonly provided by middleware products such as COM+ and products implementing CORBA (Orfali *et al.*, 1996). When developing an application the decision can be made to simultaneously develop horizontal services that support the entire application. The horizontal services then become an important part of the technical infrastructure in the application (Herzum and Sims 2000).

Vertical services are constructed in a context of a business domain. The criterion for structuring functionality into vertical services is that the service should represent a part of the organisation, or a need of the organisation. Two examples of decomposition are letting the services represent business functions (Wald and Stammers 2001), or letting the service interface represent a business process. A vertical decomposition of services is useful when using services as means of business process integration. Vertical services are equally important in inter and intra organisational integration.

Decomposition into vertical and horizontal services can be used in combination to utilise the advantages of both approaches.

5.2. Instruments for Global Interfaces

The use of services as a way to interconnect possibly heterogeneous systems requires the use of an "interconnection stack" ranging from technical network protocols and interface descriptions to semantic descriptions of how to use the service. Defining and standardising the levels in such an interconnection stack give consumers and providers a common means for communication. To provide a solution for communication differences on each level in the stack is a vital instrument to reaching the goal of services as global interfaces.

The interconnection stack can be divided into three distinct levels. The first level, technical infrastructure, consists of the basic "plumbing" needed for services to communicate, regardless of business domain and provider. The second layer, interface descriptions, is needed to communicate with a specific service. The third level, the semantic level, is needed to understand the context of service use. The levels are described further in the following paragraphs.

Technical infrastructure descriptions define how to communicate with the service. These descriptions include which communication protocols the service can use, such as TCP/IP or UDP. Technical descriptions also define the use of higher-level communication protocols such as IIOP (Orfali *et al.*, 1996), SOAP and RMI (Monson-Haefel, 2001). In brief, the technical description is sufficient for the consumer to connect to the service.

Interface descriptions define the programmatic interface of the services, that is, the methods that the service exposes. The description includes method names, parameters and data structures needed to call the service. These descriptions are commonly done with an interface definition language (IDL). An example of such a language is the Web Service Description Language (WSDL). In conjunction with the technical infrastructure the interface description makes it technically possible for a consumer to connect and call the service.

Semantic descriptions contain the context in which the service can be used, including the meaning of the methods in the service interface. A simple example of a semantic description for the method "add(x,y)" is that the method adds two numbers and returns the result. The consumers can certainly call the method without knowing the semantics, but they can never use it in a meaningful way without the semantic description. Semantic descriptions may include a set of conditions for use, for example by referring to standardised business contracts. Meta-data description languages such as the Ontology Web Language (OWL) can be used to extend the interface descriptions with descriptions at the semantic level (Smith *et al.*, 2002). Another example of high-level descriptions of services is process descriptions that define how a set of services can be combined to form a process. An example of a language for process description is the Business Process Execution Language for Web Services, BPEL4WS (Curbera *et al.*, 2002). BPEL4WS or other process description languages can be used to describe typical interaction schemes for consumers and providers. These interaction descriptions can be seen as a form of semantic description of the context of service use.

All of the above service descriptions are necessary in order to use a service. However, the descriptions need not be defined in an explicit way or in a way that is interpretable by software. Making service descriptions interpretable by software systems allows consumers to examine the service and to adapt to changes in technical infrastructure, interfaces and semantics.

5.3. Instruments for Dynamic Discovery

The goal of dynamic discovery is that a service consumer can switch between providers offering suitable services. In order to be of any benefit for the consumer, dynamic discovery requires a market of well-structured services.

Dynamic discovery requires that the consumers can perform searches for suitable services. During the search the consumer needs are compared to existing services offered by providers. The process of matching the consumer needs with the offered services is a form of brokering, or "matchmaking". Brokering is dependent on a comparable description of both the consumer needs and of the provided services. Description of consumer needs and services can be done on the technical, interface and semantic levels.

Brokering can be done by allowing the providers to register their service descriptions in a registry. A broker that matches the requirements with the contents of the registry can utilise different approaches to the brokering process. Besides the approach of the broker, the consumer's way of using the broker can also impact the outcome of dynamic service discovery.

Approaches to brokering, and different use of brokering are important instruments when implementing dynamic discovery. Approaches to brokering and consumer's use of brokering are described in the following sections.

5.3.1. Approaches to Brokering

The ability of the broker to find the requested service is dependent on the content of the registry and on the matching algorithms applied to find the service. Given a request from the consumer the broker searches its registry to find an appropriate provider. The brokering algorithm can be divided into three basic types; frame-based, specification-based and brokering based on mediator schemes.

Frame-based brokering is a way to search the service descriptions for values of certain fixed attributes such as name and type of the service (Klein and Bernstein, 2001). Both the service description and the query for services can be described as collections, or frames of attribute-value pairs. An example of brokers that are frame-based is brokers that implement the Universal Discovery, Description and Integration (UDDI) standard.

Specification brokering, or deductive brokering is an extension to frame-based brokering. By providing the broker with a set of rules relating to how to interpret service descriptions, the broker can find services by applying the rules to the service descriptions. An example of specification brokering is if the broker can deal with generalisations. For example, a service that provides global packet deliveries is a generalisation of a local packet delivery service. When a consumer searches for a local packet delivery service, the broker can find and return both local and global delivery services by applying the rule of generalisation. Specification brokering can also be done at the method level, which has been proposed as an aid in searching software repositories (Rollings and Wing, 1991). Compared to the frame-based brokering approach the specification-based approach requires a detailed description of the services on the interface level, and a description meta-model on the semantic level.

Mediator schemes for brokering lets both the provider and consumer actively participate in the brokering process. The advantage of using a mediator scheme is that the provider and/or the requestor can actively participate in the brokering process, thereby extending the brokering algorithm. Active participation can be achieved by using agents representing the provider and/or the requestor. An example of a broker with agents representing the providers is presented by Helal *et al.* (2001).

5.3.2. Consumer Use of Brokering

The broker contains the brokering algorithm and a registry of the providers. Consumers can control the outcome of the brokering process by altering the service request. Another important aspect that the consumer can affect is the binding time.

Bind time affects how often the consumer searches for a provider. If the search for providers is performed for each time the service is needed, the consumer can find a new provider as soon as it is available. This kind of binding is called ultra-late-binding (Bennett *et al.*, 2000) or *just-in-time binding* (JIT) (Andrade and Fiadeiro, 2001). JIT binding is not always feasible since it is associated with a performance overhead. Instead of finding a new provider for each call the requestor can bind to a provider for the duration of a session. A single session can span across multiple calls, thereby alleviating the overhead of JIT binding. Yet another approach is to use the same provider in a fixed time-frame. The fixed time-frame can be chosen to minimise the performance penalties, or be stated in a contract between the provider and the consumer.

Table 2 summarises the goals and the high-level instruments that can be used to achieve the goals.

Table 2 Goals and instruments

Goal	Instruments
Modular resources	The use of vertical and/or horizontal decomposition. Let services represent business functions or business processes. Use horizontal services for transactions, events and security.
Global Interfaces	Describe and standardise the technical infrastructure, interface and semantic aspects of provided services.
Dynamic discovery	Usage of frame, specification or mediator/agent based brokers. The use of fixed, session or JIT binding.

6. Applying Services

Users of services need not embrace all of the defined goals. Instead, partially fulfilling the goals can be enough in certain situations. Fulfilling part of the goals can be done by selecting the instruments to be utilised. By selecting and combining instruments, service-based systems can be tailored to a certain need. For example, a fixed point-to-point integration between two systems can initially use a loosely-defined vertical-service decomposition (an instrument to achieve the goal of modular systems) with a well-defined communication standard on the technical and interface level (an instrument to achieve the goal of global interfaces). Later, when the need to integrate more systems arises, the solution can be extended by using more instruments such as a simple frame-based broker (part of the dynamic-discovery goal).

By examining the situation where services are to be used, it is possible to identify which service goals to strive for. Then, using the defined goals it is possible to select the instruments that is best suited for the given situation.

The next section presents a categorisation of service uses. For each category, the service goals and instruments that are suitable for application are identified.

6.1. Categories of Service Use

Uses of services can be examined along two dimensions, the extent of service use, and the reach of service use. How extensive service use is, is determined by deciding if services are used as the key infrastructure/architecture for system construction, or if services are used for solving single, point-wise problems. The reach of service use can be divided into use within an organisation and use between organisations.

The differentiation between infrastructure use and point-wise use of services is important as it impacts on the need for precise decomposition and standardised service descriptions. Using a small set of services as point-wise solutions does not require precise decomposition criteria and interconnection standards. The small amount of services makes it possible to handle differences in composition and interconnection formats on a case-to-case basis. However, managing a large set of services with different communication standards and decompositions is expensive. Hence, if services are to be used as the main infrastructure, the importance of service decomposition and descriptions is increased. To summarise, using services as the main infrastructure/architecture relies on fulfilment of the goal of modular resources and partial fulfilment of the goal of global interfaces.

The difference between using services within an organisation and between organisations affects the need to have precise service descriptions and the need to utilise service brokers. Communication within an organisation can rely on implicit and partially-specialised protocols. When communicating with several organisations, the importance of standardized agreements such as global communication standards and standard business contracts becomes increasingly important. Utilising a service broker introduces a layer of indirection between the organisations, allowing the organisations to modify the service implementation and location without affecting the other party. Hence, using services to communicate between organisations relies on partial fulfilment of the goals of global interfaces and dynamic discovery.

Combining the dimensions of reach and service-use extent gives four combinations of service use; point-wise-internal, infrastructure-internal, point-wise-external, and infrastructure-external. Figure 2 summarises how the four uses of services are aided by fulfilment of the three service goals In this classification of service use each of the two dimensions, reach and extent, is divided into two areas of service use. This rough division is used to highlight differences in service use. However, examining an entire organisation's use of services might very well require a more detailed grading of the two dimensions.

Each of the four categories of service use and which instruments they utilise are described in more detail in the following section.

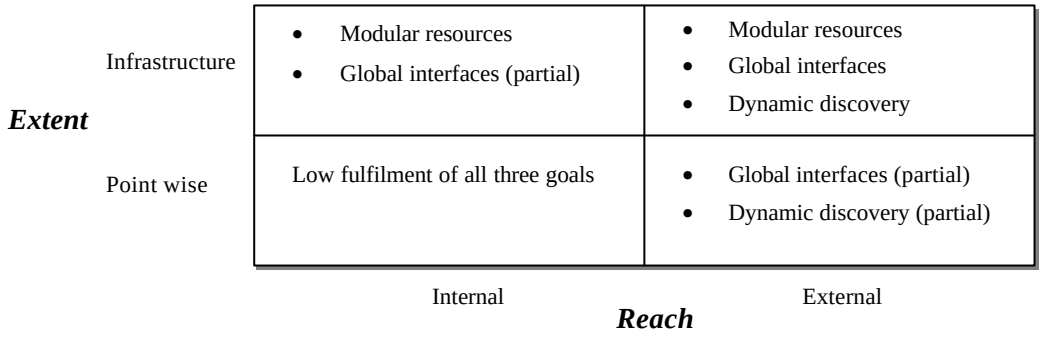

Fig. 2 The service use categories and their required goal fulfilment

6.2. Service Use and Instruments

A ***Point-wise-internal*** use of services is when services are used for single solutions within an organisation, such as when integrating existing legacy systems. Since the use of services is sparse, the need for precise service decomposition rules is low. The use of standardised technical infrastructures can help development, but since the use is internal, non-standardised solutions can be used. This kind of service use can be classified as enterprise application integration (EAI) with services.

An ***Infrastructure-internal*** use of services is when services are used as the key architecture for system development within an organisation. Using clear service decompositions, by, for example, vertically structuring the services according to business function makes the services easier to combine. As the amount of services grows, the need for standardised descriptions on the technical and interface levels increase. Use of services as the main architecture for structuring internal resources will create an internal collection of ready-to-use services, this approach combined with a message-based architecture is called an enterprise service bus (ESB) (Chappel, 2002).

A ***Point-wise-external*** use of services is when services are used for single point application integration between two organisations. Compared to point-wise-internal use, point-wise-external use is categorised by its increased need for standardised technical infrastructures and interface descriptions. Agreeing upon a non-standardised infrastructure can be difficult, since the two organisations might use different techniques internally. Using a frame-based broker with fixed or session binding alleviates the need for a tight coupling between single machines on the consumer and provider sides. This use of services as a point-wise bridge between organisations can be seen as a static business-to-business (B2B) connection.

An ***Infrastructure-external*** use of services is when services are used to connect multiple organisations in an automated way. Using services as an external infrastructure enables service consumers to find services on a global service market. Finding services is made possible by utilising service brokers. Using specification-based or mediator/agent-based brokering combined with JIT or session binding enables consumers to find and change providers. Since consumers can communicate with many providers, it is important to use highly standardised service descriptions and technical infrastructures. The use of service descriptions at the semantic level enables automatic negotiation of contracts for service use. Use of services as a global, dynamic interconnection for business can be seen as a form of dynamic business-to-business (B2B) communication.

The four categories of service-use and their required instruments are summarized in Figure 3.

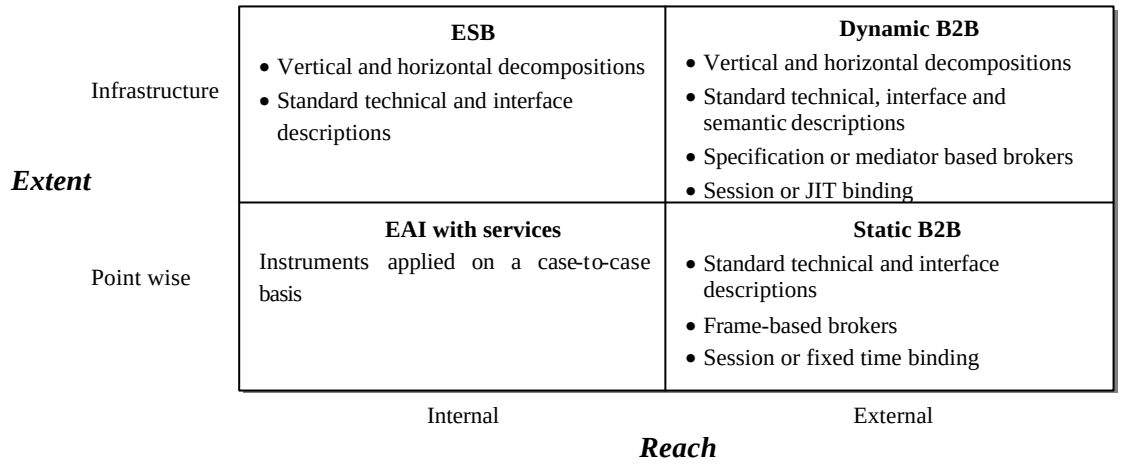

Fig. 3 The service use categories and needed instruments

7. Applying the Framework

The framework presented in the previous sections can be used as an aid for selecting suitable instruments when developing services. The following four-step process can be used as a guideline:

(1) Identify the problem domain.
(2) Map the problem domain to one of the four categories of use, as presented in the framework.
(3) Use the framework for selecting the instruments that need to be used.
(4) Decide how to implement the required instruments.

The next section presents an example of framework use. The example shows how to use the framework to guide the creation a process-centric infrastructure for enterprise use.

7.1. A Process Centric Infrastructure

A problem that organisations face when moving towards a process-centric structure is that current IT solutions are organised in a stovepipe-like architecture where each application supports a single department within the organisation. The result is isolated applications that are problematic to integrate (Johannesson and Perjons, 2000). A process-centric organisation requires that all of the applications in the organisation can be integrated and combined to support new business processes.

By following the four-step guideline previously outlined, it is possible to identify the category of use and select the appropriate instruments needed when developing services for a process-centric organisation. The following four paragraphs describe how to follow the guidelines:

(1) *Identify the problem domain*. In this case the problem is to support a process-centric organisation with services that can be easily combined and extended to support new business processes.

(2) *Map problem domain to use category.* Firstly, the described problem involves only resources internal to the organisation. Secondly, the wish to be able to combine a potentially large set of services suggests that services should be the main way of representing the business IT support. The category of service use that resembles these requirements is infrastructure-internal use of services.

(3) *Select instruments.* By using the framework, it can be concluded that infrastructure-internal use of services requires a well-defined decomposition criteria, along with the use of standardised service

descriptions on the technical infrastructure and interface levels. However, the need for advanced brokering is not an issue in this situation since the services are for intra-organisational use only. Thus the instruments needed in this situation are well-defined decomposition criteria, as well as service description and standards on the technical infrastructure and interface level.

(4) *Decide how to implement the instruments.* When the required instruments are selected, it is first necessary to select a criterion for decomposition. The organisation in this situation is process-centric, so that a suitable decomposition criterion might be a vertical decomposition where each service provides support for a single business process. An additional layer of services can be organised to support the underlying business functions (another vertical decomposition). After selecting the decomposition criterion a suitable technical infrastructure needs to be selected. Several vendors provide both standardised and proprietary technical infrastructures. An example of a suitable technical infrastructure would be Web-services, using SOAP on top of HTTP as the means for communication. Using Web-services as the infrastructure, a suitable interface description language would be WSDL.

The suggested solution can be extended with the use of instruments for service descriptions on the semantic level, such as OWL. Furthermore, the solution can be augmented with an UDDI compliant broker for intra-organisational use.

8. Conclusion and Future Work

This paper has presented a vision behind software services, and identified three distinct goals that are a prerequisite for achieving this vision. The instruments required for achieving each of these goals were presented. Furthermore, a classification of service uses was presented, and it was shown how different categories of service usage could take advantage of the defined instruments.

These elements have been combined to form a framework for the interconnection of service vision, goals, instruments and uses. This framework gives us a context for discussing, developing and using software services.

The potential use of the framework has been demonstrated by proposing a set of instruments needed to create process-centric services for intra-enterprise use.

The presented framework is a coarse-grained overview of the vision, goals, instruments and uses of software services. Future work will entail the extension of the framework both in detail and scope. Two possible extensions to the detail of the framework are to develop guidelines for selecting and combining decomposition criteria, and to further examine the situations where advanced brokering is suitable. The scope of the framework can be extended by including a mapping of the described high-level instruments to currently available technologies such as Web services, ontologies and ebXML. The scope could also be extended by mapping the use categories to requirements on non-functional aspects such as security and quality.

9. References

Allen, P., and Frost, S., 1998, “Component-Based Development for Enterprise Systems: Applying the Select Perspective,” Cambridge University Press.

Andrade, L., and Fiadeiro, J., 2001, ”Coordination Technologies for Web-Services,” OOPSLA Workshop on Object-Oriented Web Services.

Bellwood, T., Clément, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y. L., Januszewski, K., Lee, S., McKee, B., Munter, J., and von Riegen, C., 2002, "UDDI Version 3.0," Published Specification, 19 July 2002, http://uddi.org/pubs/uddi_v3.htm, Accessed 7 Jan 2003.

Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L., and Munro, M., 2000, “Service-based software: the future for flexible software,” Proceedings of the 7th Asia-Pacific Conference on Software Engineering, APSEC 2000, pp. 214-221.

Bennet, K., Munro, M., Gold, N., Layzell, P., Budgen, D., and Brereton, P., 2001a, "An Architectural Model for Service-Based Software with Ultra Rapid Evolution," Proceedings of IEEE International Conference on Software Maintenance, pp. 292 – 300.

Bennett, K., Jie Xu, Munro, M., Zhuang Hong, Layzell, P., Gold, N., Budgen, D., and Brereton, P., 2001b, "An architectural model for service-based flexible software," 25th Annual International Conference on Computer Software and Applications, COMPSAC, pp. 137 -142.

Brown, A. W., 2000, "Large-Scale Component-based Development," Prentice Hall Inc.

Chappell, D., 2002, "Asynchronous Web Services and the Enterprise Service Bus," http://www.webservices.org/index.php/article/articleview/352/4/24/, Accessed 7 Jan 2003.

Chinnici, R., Gudin, M., Moreau, J., and Weerawarana, S., 2003, "Web Services Description Language (WSDL) Version 1.2," W3C Working Draft 24 January 2003.

Curbera, F., Goland, Y., Klein, J., Leymann, F., Roller, D., Thatte, S., and Weerawarana, S., 2002, "Business Process Execution Language for Web Services, Version 1.0," Public draft release 31 Jul 2002, http://www-106.ibm.com/developerworks/library/ws-bpel, Accessed 3 Feb 2003.

Dumas, M., O'Sullivan, J. , Heravizadeh, M., Edmond, D., and Ter Hofstede, A., 2001, "Towards a Semantic Framework for Service Description," Proceedings of the 9th IFIP Conference on Database Semantics.

Gudin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H. F., 2002, "SOAP Version 1.2," W3C Candidate Recommendation 19 December 2002.

Fremantle, P., Weerawarana, S., and Khalaf, R., 2002, "Enterprise Services," Communications of the ACM, October 2002, Vol. 45, No 10.

Harumi, K., 2000, "Surveying the E-Services Technical Landscape," Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, pp. 94-101.

Helal, S., Wang, M., Jagatheesan, A., and Krithivasan, R., 2001, "Brokering based self organizing e-service communities," Proceedings of the 5th International Symposium on Autonomous Decentralized Systems, pp. 349-356.

Herzum, P., 2001, "Web Services and Service- Oriented Architectures," Distributed Enterprise Architecture Advisory Service, Executive Report Vol. 4, No. 10.

Herzum, P., and Sims, O., 2000, "Business Component Factory," OMG Press.

Johannesson, P., and Perjons, E., 2000, "Design Principles for Application Integration," 12th Conference on Advanced Information Systems Engineering, Springer LNCS.

Klein, M., and Bernstein, A., 2001, "Searching for Services on the Semantic Web Using Process Ontologies," Proceedings of the Semantic Web Working Symposium.

Lovelock, C., Vandermerwe, S., and Lewis, B., 1996, "Services Marketing," Prentice Hall Europe.

Monson-Haefel, R., 2001, "Enterprise JavaBeans," O'Reilly & Associates.

Orfali, R., Harkey, D., and Edwards, J., 1996, "The Essential Distributed Objects Survival Guide," John Wiley & Sons Inc.

Péraire, C., and Coleman, D., 2000, "Modeling for E-Service Creation," Technical Report, SRI international.

Piccinelli, G., and Stammers, E., 2001, "From E-Process to E-Networks: and E-Service-oriented approach," OOPSLA Workshop on Object-Oriented Web Services.

Platt, D. S, 1999, "Understanding COM+," Microsoft Press.

Rollings, E., and Wing, J., 1991, "Specifications as Search Keys for Software Libraries," Proceedings of the 8th International Conference on Logic Programming.

Smith, M. K., McGuinness D., Volz, R., and Welty, C., 2002, "Web Ontology Language (OWL) Guide Version 1.0," W3C Working Draft 4 November 2002.

Stahl, M. , 2002, "Beyond Component-Based Computing," Communications of the ACM, October 2002, Vol. 45, No 10.

Wald, E., and Stammers, E., 2001, "Out of the Alligator Pool: A Service-Oriented Approach to Application Development," EAI Journal, March 2001.

Webber, D., and Kotok, A., 2001, "ebXML: The New Global Standard for Doing Business on the Internet," New Riders Publishing.

Yang , J., van den Heuvel, W. J., and Papazoglou, M. P., 2001, "Service Deployment for Virtual Enterprises," Australian Computer Science Communications, Vol. 23, Iss. 6.