Modular Service Architectures: A Concept and Method for Engineering IT services[1]

Tilo Böhmann, Markus Junginger
University of Hohenheim
70593 Stuttgart (Germany)
{boehmann|markjung}@uni-hohenheim.de

Helmut Krcmar
Technical University of Munich
85748 Garching (Germany)
krcmar@in.tum.de

Abstract

The demise of first generation service models of application service providers (ASP) indicates that moving from individualized services (e.g. many outsourcing offerings) to fully standardised services (e.g. first generation ASP models) leaves many customer requirements unfilled. In our paper we argue that a possible solution for building innovative e-services is the use of modular service architectures.

The paper introduces the characteristics of IT services, based on a case study with a leading German provider of application hosting services in the ERP market. We pay particular attention to the general composition of IT services, the role of external factors (how customers and their resources participate in service operations), and varying customer preferences. Referring to these considerations, the paper explains the concept of modularity and the application of the concept in the domain of IT services. We specifically address how the concept of modular service architectures addresses external factors and varying service level requirements. We give an example how the concept of modular service architectures can be leveraged to improve the design and to allow for improved customization of IT services. We conclude with identifying possible further research questions that need to be pursued to achieve the vision of semi-standardised, mass customizable solutions for IT service provision.

1. Introduction

Following a forecast of the IDC, the IT services market will increase at a healthy rate of 12,4% until 2006 [1]. The challenge for IT-service providers is the identification, engineering and implementation of innovative service models to achieve a prime competitive position. Now some industry analysts predict a fundamental shift from individually designed IT services towards mass customized solutions [2]. In this paper we argue that a possible solution for the development of mass-customized IT services is the use of modular service architectures. Modular service architectures enable service providers to customize services and reuse modules across multiple service products. We therefore start our paper with identifying the characteristics and drivers for variety of these services based on a case study with a leading German provider of application hosting services. Following these considerations we have a close look at the concept of modularity and modular service architectures and show the application of the concept for the development of IT service architectures. We conclude the paper with perspectives for further research in this area.

2. Characteristics of IT services

2.1. General Composition

According to the Gartner Group IT services are defined as "… the application of business and technical expertise to enable organizations to create, manage, optimise or access information and business processes" [2]. Following Krcmar's [3] model of information management, service relationships in IT services comprise three different levels. Corporate information sources, information systems and IT resources. Irrespective of which of these three levels a service entails, there are governance and management activities that plan, coordinate and control service customisation and delivery. The following considerations are focusing on the development of new methods for an efficient, innovative and successful service engineering in the IT services market segment of application hosting which is located on the both levels information systems and IT

[1] This research is part of the pro-services project that is funded by the German Ministry for Education and Research (Contract No. AT-ATHG011112700-01HG0067). Members of the project consortium are the University of Dortmund, Hohenheim University, and various industry partners.

resources of the information management model.

This research is part of an action research project conducted in collaboration with ALPHA corporation[1], a leading German provider of application hosting services. Action research [4] aims at producing highly relevant results because it is grounded in practical action. Its objective is solve an immediate problem situation while carefully informing theory [5]. As part of the project we conducted a case study [6, 7] of a successful ERP hosting service (SAP R/3) to analyse the value proposition, service model and customer demands. Currently the ERP hosting service hosts more than 320 SAP R/3 systems for 120 customers. These findings provide a foundation for developing and field testing a method for designing modular service architectures.

2.2. Characteristics of the Application Hosting Service

2.2.1. Value Proposition. ALPHA's value proposition of application hosting service comprises the capabilities to implement and manage the operations of standard business applications for small and medium-sized enterprises. Overall, ALPHA's portfolio of services covers both application-centred (e.g. SAP R/3) and industry-centred services (e.g. banking applications). As opposed to traditional IT outsourcing, application hosting services have a reduced set of customer interfaces so that the internal performance of the service provider takes place as efficiently and standardised as possible. For example, the strategic orientation of IT remains the responsibility of the customer. The applications that are run in this model are often seen as "commodities" whose efficient operation is not seen as a source of competitive advantage. According to a senior manager of ALPHA, the financial volume of contracts for application hosting services is only 10% compared to strategic outsourcing agreements while being 40% more cost-effective for the IT service provider. In comparison to the application service providing-Model (ASP) that offers highly standardized applications via Internet in a leasing model [1], application hosting provides the opportunities to operate systems that are specifically configured for every customer and the systems are accessed via a dedicated line to the company's local network. The service is charged by a monthly fee which is based on the number of stipulated users (seats per month).

A look at the standardized catalogue of the IT service provider with the most important options for SAP R/3 application hosting projects shows the classes for an individualized configuration of different customer systems (Table 1). They are the basis for the further implementation and service level definition. ALPHA regards its long-ranging experience with application hosting (since 1975) and the ability to react individually to customer needs as its unique selling proposition. ALPHA's goal is to create business value via economies of scale, e.g. as a result of shared use of data storage components, network infrastructure for all systems and more favorable procurement conditions by suppliers. Additionally, ALPHA's services complement the capabilities of small and medium-size IT departments, e.g. through providing better contingency planning and facilities.

Table 1: Classes of Options for SAP R/3 Application Hosting Services

Project Management	Implementation (new system)
	System transfer (existing application)
System Configuration	System landscape
	Hardware
	Operating System
	Database
	SAP R/3
	Network-connection
	Data storage
	Technical approval
Performance Index	Operation-time
	Availability
	System response time
Operations Management	Monitoring
	System-Management Tools
	Data storage
	Tuning
Maintenance	Hardware
	Release Migration
Support	Service Center
	Hotline SAP R/3 Basis
	Hotline SAP R/3 Application
	Advanced Support
Infrastructure	General
	Tools
	Security

2.2.2. Service Model. After describing the value proposition of ALPHA's application hosting services we now turn to their service model. The service model analyses the constituent parts of the service and their coordination that are necessary to implement and run the ERP system. We firstly focus on the process view of the service model that comprises three distinct processes: the sales process, the implementation process and the operating process.

During the sales process potential customers are addressed and advised on service portfolios and the advantages of the application hosting service. The sales department develops a configuration of a system that is specially adjusted to the customer needs in cooperation with system specialists of the pre-sales group. This process results in an offer for the customer and possibly in the conclusion of a contract. The first part of the implementation process is assembly and installation of

[1] Real name disguised to maintain anonymity

the basic infrastructure (e.g. server hardware, operating system, database server). The second step, the application system is either set up and configured or transferred from the customer to the service provider. This step may entail the migration of the application system from other operating system environments or database systems into ALPHA's standard environments. The implementation process ends with a test phase and a formal going-life of the system. The process requires from 2 up to 9 months depending on the scale and complexity of the project. The following operation process monitors the system in operation, performs system management and recovery activities. While generally a customer service centre attends to customer requests during the operation process, larger customers are assigned service managers for coordination. The smooth transition between these processes is ensured by the project kick off and the going life meeting in which the members of the related processes provide information about the new customer system and hand out the generated documentations, especially the specifications of the implementation are documented in the operation handbook. All processes have a distinct customer interface with a specifically assigned main partners for interaction. For most customers, the intensity of interaction is reduced and the standardization of interaction increased when transferred from the implementation process to the operation process.

The task-oriented view on the integration process allows to identify service functions. The integration processes connect the service functions and bridge interdependencies between them. Figure 1 describes the main service functions of ALPHA's ERP hosting service. Every service function is customized during the sales phase, implemented by different responsible team members of the implementation project team and transferred to the operation process afterwards.

2.3. Drivers for Variety of Application Hosting Services

2.3.1. External Factors. The value proposition and service model underscore the semi-standardised nature of ALPHA's ERP hosting service that mixes adaptable and standardised elements. We therefore have a closer look at the drivers for variety of the hosting service, mainly external factors and variations of customer requirements. These drivers help to explain the limitations for standardising IT services.

To deliver a service, service providers often need to incorporate personnel, processes or technical resources of the customer into their service operations. These resources are called external factors [8] because the service provider has only limited control of their characteristics. Thus, they need the capability to flexibly adapt service processes to the characteristics of individual external factors. The external factors may be necessary resources for service production or the object of service performance. Typical external factors of application hosting services are the individually customized application system, other systems the application interfaces with, the technical infrastructure of a customer as well as processes of IT governance and use. As a result, application hosting services need to be adjustable to external factors to a great extent.

For ALPHA's ERP hosting service the main external factors are the specifically customized application systems of the customers. Consequently, ALPHA is able to operate the standard application for every customer with its customer specific customizing. Furthermore, ALPHA has capabilities for individually setting up new application systems and transfer and migrate existing systems into its operating environment. This customizing refers to basic system values as well as to specific application extensions and queries. Additionally, customers often integrate the ERP system with other business applications. In this case ALPHA needs to be able to implement and operate these interfaces. Another external factor to be considered is

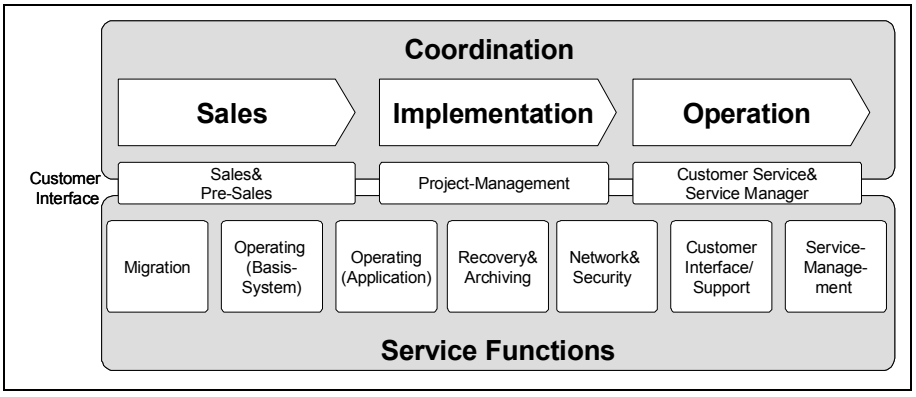

Figure 1: Coordination and Integration of Service Functions

the customer's IT infrastructure. This infrastructure comprises, for example, local networks at different geographical sites of the customer with their capacity and protocols. Some customers furthermore have existing wide-area networks. Therefore ALPHA needs the ability to cooperate with other service providers in connecting to the customer's network via dedicated lines or virtual private networks. Likewise the capability of the customer's on-site end-user devices (e.g. desktop systems) is an important external factor especially in client/server ERP solutions. Weak client systems may lead to considerable problems in application performance that impair the perceived quality of service delivery.

In the case of ALPHA's ERP hosting service, the governance of IT systems and their use remains with the customer. Therefore the customer needs to be able to modify and extend the application in the future, e.g. to extend the existing systems by an e-shop or e-procurement solution. Furthermore, ALPHA must be able to accommodate the individual release upgrade policy of a customer that defines the timing of upgrades during the service relationship. Also the monitoring and the recognizing of the variation of the external factor challenges the service provider. An increased transaction volume may require scaling the capacity of the basic infrastructure and the application system. In sum, the wide implications of external factors in ALPHA's ERP hosting service creates forceful drivers for variety in the service that designers of new services need to take into consideration.

2.3.2. Varying Customer Preferences. Not only external factors are a source for variety but also heterogeneous customer demands that require an individual service configuration to match those demands. The ERP systems ALPHA hosts vary considerably with regard to the number of users and, as a result, the necessary processing power and data volume of the basic infrastructure. Furthermore, customers have different requirements regarding the release level of the application system. Some customers maintain an relatively old release level because of specific extensions or license fees associated with upgrades. ALPHA, as a result, currently has to operate SAP R/3 systems ranging from release level 3.1 to 4.6.

Moreover, ALPHA addresses varying customer demand by offering different options for reaction time, times of system availability, reliability of the system, response time of the system, application support and network connectivity. Depending on the business environment of the customer there are different needs in regard to the reaction time in the system- and application support. Depending on the criticality of the application for the customer's business processes, ALPHA has to be able to classify and solve support requests within the reaction time specified. Likewise there are different requirements to the times of availability of the support. Customers also have different preferences for system availability and response times of application systems. For customers of the banking sector it is most of time indispensable to use highly available systems with a backup data centre whereas smaller manufacturing firms or companies which only use single modules of the software generally do not require and are unlikely to be ready to pay for. This is similar to the case of response times of the application systems. Some customers that have to process high volumes of transaction (e.g. order taking) require consistently short response times even during peak times.

In addition to these core features of application hosting services ALPHA also flexibly addresses temporary customer requirements, such as providing additional servers for quality assurance during larger development projects related to the hosted application. Even demands for intervals and type of service level reporting can vary if customers want to integrate those reports with internal quality management processes. Consequently, ALPHA needs be able to accommodate the varying customer requirements and to adapt service functions according to the customer's needs through customizing a service individually for a customer.

External factors and customer preferences, the drivers of variety, shed light on the underlying dilemma of the design of corporate IT services: while striving for economies of scale in service operations through standardization of services, from a customer's point of view the service provider has to offer services which represent individual solutions that are customized to customer needs. Other industries, such as the automotive industry, have employed the concept of modularity to move more into a direction of customizable products build from a standardized base. In the following we demonstrate how the concept of modularity can also be applied to the design of customizable IT services.

3. Modular IT services

3.1. General Principles of Modular Service Architectures

Several characteristics are typical for modular services. The first characteristic is decomposition. [9-12]. Separating a complex service into several units simplifies its development and operations. For decomposing a service into units one needs to precisely describe the role each unit plays in the service. Using an abstraction, e.g. a model, of each unit for decomposition reduces the complexity of defining and managing the service [10, 13]. As each separate unit may itself be divisible into further subunits, a nested hierarchy of

decomposition characterises modular systems [10, 14, 15]. As designing modules is a design task requiring certain knowledge and effort, the detail at the upper and lower bounds of the hierarchy is for business enterprises defined judging the additional benefits of furthering modularisation against the cost of analysis and design this requires [15].

To arrive at a modular service one needs to decompose the service into units are loosely coupled. Loose coupling means that each unit is relatively independent of other units. This can be done by clustering those elements of a service that have strong interdependencies within units so that units have only limited interdependencies among each other [10]. Closely related is the principle of information hiding [16]. Specifications define and standardise the roles of each unit and the interactions required to resolve the remaining interdependencies. These specifications hide all other information about the units. As a result, designing and delivering each unit becomes independent of other units. Changes in one unit do not lead to changes in other unit, as long as the specification of the roles and the interfaces of the units are maintained. Thus, service modules are the result of decomposing a service into a nested hierarchy of loosely coupled units, using abstractions for these units and hiding the local information through standardised roles and interfaces.

For developing (physical) products, however, various authors have underscored the strategic implications of how a product is split into modules. The boundaries that designers and engineers define during product development can have long term implications. Among other things, modular designs create options for recombining modules within or across generations of a product [12, 17, 18]. Ulrich suggests the term product architecture for this scheme that defines modules and their interactions. The architectural perspective underscores the importance an outline view of a product that reflects strategic design decisions and projects them into the development and production of a product. Burr contents that this architectural view also holds important insights for modular services [9].

A service architecture [9] is thus the expression of an agreement about interfaces and boundaries that is enforced throughout service engineering and operations. It comprises those architectural decisions that guide the development and operational efforts within individual modules and that cannot be changed unilaterally. Furthermore, the service architecture establishes an integration framework for service modules through which they can be combined into new or improved services. It is an agreement between those developing and/or delivering the service that needs to be enforced throughout its intended scope and lifetime [10-12]. As a result, architectural decision generally precede other activities in service engineering and operations and guide their execution.

Based on a modular service architecture, designers can define service products for particular market segments and service configurations for individual customers. The architecture enables the designers to mix-and-match modules to specific requirements from the set of modules the architecture comprises.

3.2. Describing Service Modules and Service Architectures

To build IT services from a set of service modules requires information about the service architecture and its service modules. In the following we propose what items need to be specified about the modules and the architecture so that they can be easily leveraged in service engineering processes.

Services: The description of a service module needs to contain a specification of the services that the module provides. Such a specification lists the features of the service and defines options (e.g. different levels of system availability). These options allow for further customisation of the module to match specific customer requirements. Services, however, do not only flow from the service provider to the customer. When the service provider integrates external factors into service operations, the list of services also defines the services that the customer needs to provide for successfully delivering the service. Such a description of the services of a module serves two purposes. It firstly establishes the boundaries of the module within a service architecture. As the defining module boundaries is a critical design decision in modular systems [10], the description documents this decision. Secondly, a detailed description of the scope of services is also an important part of the contract governing the service relationship. Thus the description can be leveraged in designing appropriate contracts for services that comprise the module.

Pricing & Billing: The scope of services also entails pricing and billing models that specify financial flows for these services. Many IT services involve some sort of financial transformation, e.g. converting relatively high payments at the beginning and relatively low payments for ongoing operations into a flat fee paid in regular intervals. Furthermore, pricing and billing models may be dependent on measuring inputs, throughputs and outputs of service operations. The choices for pricing and billing a particular service module therefore are often linked to measurement and billing systems that interface with operational systems. Therefore the scope of pricing and billing may be limited by the capabilities of these systems. To make this information available in service engineering, it becomes part of the description of a service module.

Customer Experience: The customer experience encompasses all instances of communication or cooperation between customers and the service provider in delivering the service, such as co-producing parts of the service, requesting execution of service orders, adapting a service to customer needs, recovering from failure, and communicating service performance. The customer experience has a strong influence on the perception of service quality. It also shapes the visibility of service operations for third parties, which facilitates or inhibits the imitation of the service. Furthermore, defining the instances of customer interaction and assigning them to modules ensures a consistent customer experience. Otherwise the independency of modules may lead to conflicting or insufficient interaction of multiple modules with a customer of a service, leaving the customer to coordinate service delivery across modules. The definition of the instances of the customer experience of a service and the involvement of individual service modules is therefore a critical part of service engineering.

Interfaces: Interfaces define how service processes interact. Like the definition of module boundaries, their specification represents another key area of design decisions of a service architecture [10]. They are in fact internal service level agreements of the service provider, e.g. defining how service processes of different modules connect. How they are specified depends on the type of interdependency that the specification governs. For organisational interfaces, this may be quality management document or even an internal contract. For technical interfaces, a more technical specification is required, e.g. a specification of transactions in ebXML.

3.3. Defining Modules for IT services

How should IT service providers now group elements of their services such as IT systems and service processes into modules? For this paper we focus on defining the scope of services for each module, as treating the implications of payment and pricing models as well that of the customer experience are beyond the scope of a single paper. Defining modules entails five steps: define service functions, identify drivers for variety, assess impact of drivers on functions, define candidate module using module types, and specify module variants and interfaces.

The *first step* is to develop an overview of the functions of the IT service to be developed (cf. 1 in Figure 2). These functions may be identified using a customer-centric design method such as Quality Function Deployment. Alternatively, project documentation of solutions for individual customers or existing service operations can serve as a starting point for identifying those functions that satisfy customer demands. For IT services, the design team needs to look at:

(1) Business functions: the business functions the service should provide (e.g. process pay slips). This category of functions may be the dominant one for a IT-based business service (e.g. e-payment services)
(2) Application functions: the core functions of the application that the service delivers (e.g. human resource management information system) and the auxiliary functions, such as output options (e.g. fax delivery service) and interfaces with other applications (e.g. FTP file import/export). This category may be particularly important for vertical application service providers, but most IT services are likely to have to cover some auxiliary application functions.
(3) Service providing functions: the functions required for delivering business or application functions with the flexibility and quality that customers expect (e.g. availability, speed, recovery time and procedures, etc.). This category is likely to be present for all IT services. For services that provide COTS software to their customers, the service providing functions are particularly important for competitive differentiation (as e.g. in application hosting services).

In our example we focus on service providing functions for a COTS application system. The customisation of this application is regarded as an external factor. The provider operates an application system that the customer or a third party has customized. The functional decomposition is documented for further analysis.

Following Ulrich's concept of a modular product architecture, the functional view of the service helps to define service modules. Ulrich argues that if module boundaries reflect the functional structure then new functional requirements can easily addressed by recombining the modules [12]. Customer choices and external factors, however, can lead to varying demands for performance of individual functions or for collective performance of multiple functions.

As a *second step* the design team needs to establish the drivers for variety in the design of the IT service to understand the implications of customers choices and external factors (cf. 2 in Figure 2). As argued earlier, heterogeneity is increased by external factors, and varying customer preferences that lead to optional service features, all of which need to be identified and mapped to the functional structure as the next step of service development. To do so, the design team has to isolate the relevant drivers for variety. Which external factors will the service involve? What will be the options and variants customers will require? What adaptations will customers need during the lifetime of

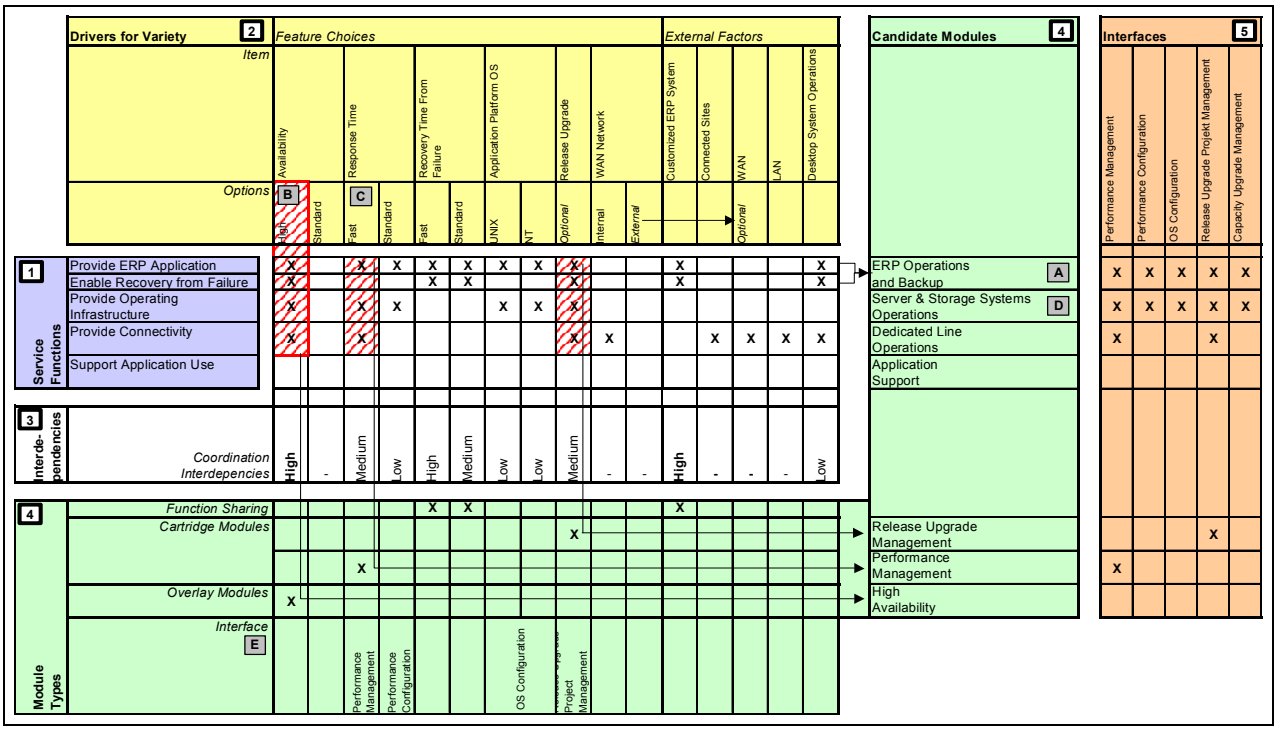

Figure 2: Example Application of the Method

the service? This step is particularly important for defining a modular service architecture as it safeguards that the design of the modules reflects the effect of external factors, the variation of customer requirements, and the flexibility of the service over time that customers demand.

The *third step* is to assess the impact of these drivers on defining modules (cf. 3 in Figure 2). The design teams analyses which functions an individual driver has an effect on. Having an effect means that a customer choice or an external factor creates specific requirements for the function concerned. In our example, the optional feature of fast recovery time from failure generates requirements in terms of backup procedures and processes for the function "provide application". These requirements are distinct from the requirements for standard recovery time from failure. The impact of the drivers depends on the number of modules or service functions they have an effect on. If they affect only a single one, they do not create interdependencies among them. Customization or adaptation requirements may be designed as variants of the module candidate or function. A particular variant can then be selected either when defining a service product or a service configuration. If a requirement affects multiple module candidates or functions, however, the design team needs to evaluate the resulting interdependencies. Depending on the complexity of these interdependencies the team can follow different modular design options to accommodate them.

Coordination theory provides some assistance in identifying interdependencies between activities of service processes. Following coordination theory [19, 20], there are three main types of interdependencies that mechanisms of coordination address. Flow interdependencies occur if one activity of a process consumes a resource that another activity produces. Sharing interdependencies arise if two different activities use or consume the same resource. Fit interdependencies occur if two different activities jointly produce or manipulate the same resource. These interdependencies may occur at different stages of the service life cycle for an individual customer, e.g. when customizing or operating a service. These interdependencies can have different levels of complexity that affect the definition of modules in a different way. The table 2 gives some indication how the design team can gauge the complexity. The assessment is based on the effort required for solving the interdependency.

The *fourth step* is to identify candidate modules from the preceding analysis. To define modules, we suggest four types of modules whose definition the matrix supports: function sharing modules, overlay modules, cartridge modules, and single function modules (cf. 4 in Figure 2).

Functional Sharing Modules: Ulrich suggests to establish a 1:1 mapping between functions and modules while maintain loose coupling. While the functional structure serves as a starting point for identifying

Table 2: Levels of Complexity of Coordination Interdependencies

Level of Complexity	Possible Mechanisms for Coordination
Low	Simple routines or internal service level agreements for defining coordination between the affected modules
Medium	Specific coordination resources and specific interfaces linking the module with the other affected modules
High	Specific resources working in close cooperation required, interaction between activities cannot be standardised through interface specifications

candidate modules, the impact of drivers on functions may require modules that conflict with Ulrich's proposition. Highly complex interdependencies are likely to have the greatest impact on defining module boundaries. A high level complexity cannot be framed through standardised interfaces. If the external factor, variance or change requirements requires close collaboration, possibly even co-location of the activities, these requirements can only be covered through a separate module that provides all the functions that are affected. The impact of highly complex interdependencies depends on whether the cause of the interdependencies is a mandatory or optional characteristic of the service. Typical sources for permanent interdependencies are external factors and features that have the same interdependencies irrespective of the options selected for the feature. As the interdependencies created are present in all configurations of the service, the affected functions should be grouped in a single module. Such function sharing [12] allows to address permanent coordination interdependencies that are not affected by feature choices or adaptations. In our example, the external factor "Customized ERP system" creates highly complex non-optional interdependencies between the first two functions. As a result they are grouped into a single module (cf. A in Figure 2).

Overlay Modules: If, however, the complexity varies among the options for a feature or the external factor may not be present in all configurations, the definition of modules needs a more differentiated approach. For optional interdependencies, the design team can decide to develop an overlaying module. While the standard architecture uses the scope for decomposition that exists for service products and configurations without the optional requirement, they specify a module that supersedes the basic architecture for those cases where the requirements needs to be fulfilled. This overlay module needs to implement all the external interfaces of the basic modules but can be developed with closely coupled interactions within. A good example for IT services is the requirement of a high level of overall availability of a system. Mission critical applications, e.g. settlement systems in the financial service industry, require an extremely high degree of uptime. To provide such systems, the service provider needs to incorporate several technical components across various service components (e.g. application, service & storage, network, etc.) that increase technical reliability of the overall system. Furthermore, service processes (e.g. in application management) also need to reflect the required level of availability. As a result, this non-functional characteristic involves close interdependencies between several service components. If high availability is an optional characteristic of the service, an overlay module may be a suitable solution for the service architecture of the service (cf. B in Figure 2).

Cartridge Modules: Interdependencies of medium complexity may necessitate dedicated resources for coordination, e.g. a specific task for an individual or a team or an information system that is particularly adapted for this purpose. A good example is the feature option "low response time". Maintaining a low response time while data volume and application enhancements increase over time requires a continuous tuning of all the layers of application delivery. Tuning may need specific performance management processes and resources, e.g. additional staff and specific processes to ensure monitoring and improving of the service components in shorter intervals than usual. As these resources are only required if the associated option is selected in configuring the service, it is sensible to separate them from other modules. The design team further needs to define the interface between this new candidate module and others through which the coordination requirements are met. As the option is activated when configuring the service through adding a module to the configuration we call this solution a *cartridge module* that addresses particular non-functional service choices (cf. C in Figure 2).

Single Function Modules: Coordination requirements of low complexity generally only require an interface between the affected modules that defines the flow, sharing or fit of the activities of the service processes involved. An example for this case is the options for selecting the operating system of the application platform. The resulting coordination requirements can be embedded in a standardised workflow interface so that module boundaries are not affected by this driver. In this way functions remain that only have non-optional interdependencies of medium or low complexity. These modules then exhibit a one-to-one mapping to a functional element of a service and are

thus called single function modules (cf. D in Figure 2).

The *fifth step* brings a more detailed specification of the service architecture and the service modules. Based on the module and the module types, the design team can define interfaces between modules that address the coordination requirements identified in the preceding analysis (cf. 5 in Figure 2). Furthermore, the matrix helps to identify module variants and options. These selection define the configuration of an individual module. To identify module options, the design team identifies which feature options, optional external factors and optional adaptations have an effect on the module. For each impact the design team develops a list of choices to be made when selecting a specific module variant. From this list a detailed description of the scope of service can be developed that forms the basis of the description of the module in the modular service kit.

3.4. Related Research

There is little research to date on modular service architectures per se, the noted exception being the work of Burr [9]. He proposes to partition service modules by assigning them a single function, a single partial service (i.e. task) needed to perform the function and an independent organizational unit that provides the resources of performing the partial service. However, he does not discuss how drivers for variety impact the partitioning as our method does.

From our case study it is obvious that IT services are built around IT systems. Research on the architecture of IT systems may therefore be complementary to our work. Even though IT services may predominately have to deal with existing systems (e.g. COTS software), conceptualizations of software architecture design still reveal key elements and critical interdependencies of IT systems. Kruchten [21], for example, differentiates four concurrent views on software architecture that shed lights on a system's elements and their relationships from the perspective of its functions, its development organisation, its run-time processes, and its deployment. The interdependencies captured in these views are likely to create interdependencies in service processes that can then be captured and addressed using the method we proposed.

4. Conclusion

We started our paper with the Gartner Group vision of semi-standardised, mass-customizable solutions for IT service provision. With our analysis of a successful service model of application hosting we identified drivers for variety in IT services. The predominant drivers for variety of IT services are heterogeneous customer requirements requiring the service to be tailored to fit into corporate information management and numerous external factors that need to integrated into service operation. The many drivers of heterogeneity may provide an explanation for the fact that IT services provision is currently dominated by solutions that follow are certain blueprint but are often individually designed and implemented. It also helps to explain why outright standardisation of complex services has often failed, as it has been attempted in first-generation ASP services.

In the second part of our paper we introduced the concept of modular service architectures as a foundation for adaptable, yet partly standardised IT services. Key for developing modular architectures is establishing module boundaries and interfaces between modules. To support this task we introduced a method that maps feature choices, service adaptations and external factors as drivers for heterogeneity onto the functional structure of the service. The resulting analysis enables service designers to identify interdependencies that result from the drivers for heterogeneity and take these interdependencies into account when defining modules. In an example we demonstrate how the method helps to identify those elements and options of an IT service that prohibit function-oriented modularisation because of performance standards that create highly complex interdependencies.

Modular service architectures appear to be a concept well-suited to the challenges of service engineering for providers of IT services. The method systematically takes into account the impact of drivers for heterogeneity that these service providers need to address. Further research, however, is necessary, in several directions. Firstly, the customer experience of a service is a critical part of service design from several perspectives. It influences the perception of service quality while simultaneously defining the visibility of service operations. High visibility may pose a strategic risk as it facilitates imitation and module "cherry-picking". The method proposed should therefore be extended to take into account the effects of modularisation on the customer experience of a service. Secondly, there has been wide-ranging experimentation with various business models for service provision (e.g. transaction-based pricing). Transforming the investment and recurring costs of service provision into new pricing models is a financial service that complements the IT service. Consequently, methods for designing IT services need to take into account the implications of the business models on which the modular service architecture is based.

Thirdly, certain elements of a service are difficult to define before commencing service delivery. They require pooling of knowledge between the client and the service provider. Not all elements, though, share these

characteristics. We thus intend to research the effects in more detail the effects of customer knowledge on defining service modules in more detail and extend the method accordingly.

Traditionally, research on IT services, and particularly IS outsourcing [22-25], has primarily focused on the perspective of the outsourcing client. Our research shifts the focus to the service provider. The objective is to help service providers to identify semi-standardised building blocks for the future corporate information infrastructure. We consider our analysis and method as a first step into this direction.

5. References

[1] N. May, "Worldwide IT Services Industry Forecast and Analysis, 2001-2006," IDC, Report 26937, 22 May 2002 2002.

[2] R. H. Brown and F. Karamouzis, "The Services Value Chain: Forging the Links of Services and Sourcing," *Gartner Research*, pp. 1-5, 2001.

[3] H. Krcmar, *Informationsmanagement*, 3 ed. Heidelberg: Springer, 2002.

[4] R. N. Rapoport, "Three dilemmas in action research," *Human Relations*, vol. 23, pp. 499-513, 1970.

[5] R. Baskerville, "Investigating Information Systems with Action Research," *Communications of the Association of Information Systems*, vol. 2, 1999.

[6] R. K. Yin, *Case Study Research. Design and Methods.*, 2 ed. Thousand Oaks, London, New Delhi: Sage, 1994.

[7] K. M. Eisenhardt, "Building Theories from Case Study Research," *Academy of Management Review*, vol. 14, pp. 532-550, 1989.

[8] M. Kleinaltenkamp, "Begriffsabgrenzungen und Erscheinungsformen von Dienstleistungen," in *Handbuch Dienstleistungsmanagement*, M. Bruhn and H. Meffert, Eds. Wiesbaden: Gabler, 2001, pp. 27-50.

[9] W. Burr, *Service Engineering bei technischen Dienstleistungen: eine ökonomische Analyse der Modularisierung, Leistungstiefengestaltung und Systembündelung*, vol. 286. Wiesbaden: DUV, 2002.

[10] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity*, vol. 1. Cambridge (MA), London: MIT Press, 2000.

[11] R. Sanchez, "Strategic Product Creation: Managing New Interactions of Technology, Markets, and Organizations," *European Management Journal*, vol. 14, pp. 121-138, 1996.

[12] K. Ulrich, "The role of product architecture in the manufacturing firm," *Research Policy*, vol. 24, pp. 419-441, 1995.

[13] H. Balzert, *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg, Berlin: Spektrum, 1998.

[14] H. Simon, "The architecture of complexity," *Proceedings of the American Philosophical Society*, vol. 106, pp. 467-482, 1962.

[15] M. A. Schilling, "Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity," *Academy of Management Review*, vol. 25, pp. 312-333, 2000.

[16] D. L. Parnas, "Information distribution aspects of design methodology," in *Information Processing 71*, vol. 1, C. V. Freiman, J. E. Griffith, and J. L. Rosenfeld, Eds. North-Holland: Elsevier, 1971, pp. 339-344.

[17] J. B. Dahmus, J. P. Gonzalez-Zugasti, and K. N. Otto, "Modular product architecture," *Design Studies*, vol. 22, pp. 409–424, 2001.

[18] A. Ericsson and G. Erixon, *Controlling Design Variants: Modular Product Platforms*. Dearborn, Michigan: Society of Manufacturing Engineers, 1999.

[19] T. W. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," *ACM Computing Surveys*, vol. 26, pp. 87-119, 1994.

[20] T. W. Malone, K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. S. Osbor, A. Bernstein, G. Herman, M. Klein, and E. O'Donnell, "Tools for inventing organizations: Toward a handbook of organizational processes," *Management Science*, vol. 45, pp. 425-443, 1999.

[21] P. B. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, vol. 12, pp. 42-50, 1995.

[22] S. Ang and D. W. Straub, "Production and transaction economies and IS outsourcing: a study of the U.S. banking industry," *MIS Quarterly*, pp. 535-552, 1998.

[23] V. Grover, M. J. Cheon, and J. T. C. Teng, "The Effect of Service Quality and Partnership on the Outsourcing of Information Systems Functions," *Journal of Management Information Systems*, vol. 12, pp. 89-116, 1996.

[24] M. C. Lacity, L. P. Willcocks, and D. E. Feeny, "IT outsourcing: Maximize flexibility and control," *Harvard Business Review*, pp. 85-93, 1995.

[25] J.-N. Lee and Y.-G. Kim, "Effect of partnership quality on IS outsourcing: Conceptual framework and," *Journal of Management Information Systems*, vol. 15, pp. 29-61, 1999.

0-7695-1874-5/03 $17.00 © 2002 **IEEE**