Managing Hybrid IT-Products

Adding IT Support to the SCORE Method

Philipp Langer[1], Thomas Winkler[2], Tilo Böhmann[3], Helmut Krcmar[1]

*[1]Lehrstuhl für Wirtschaftsinformatik,
Technische Universität München,
Boltzmannstr. 3, 85748 Garching*

*[2]avaso GmbH, München
Lichtenbergstr. 8, 85748 Garching*

*[3]ISS - International Business School of Service Management Hamburg,
Hans-Henny-Jahnn-Weg 9, 22085 Hamburg*

1 Introduction

While the economic impact of pure in-kind transfers tends to decrease, because of the declining differentiation possibilities, Hybrid Products (solutions) gain relevance. Hybrid Products are aligned products and services, which are being offered as integrated performance bundles (Johansson et al. 2003, S. 116-125). Therefore, their special trait is the added value of the integration of both, products and services, which helps to solve customer specific problems (Foote et al. 2001, S. 84). Consequently the value of the Hybrid Product is supposed to exceed the sum of its parts.

Besides the possibilities of differentiation, utilizing Hybrid Products leads to a better integration into the customer's value creation chain (C1)[1]. The Hybrid Product provider takes responsibility for the downstream activities in the value chain of products, e.g. on the integration of products to complex systems, on the provision and operation of such systems, or even on the conduction of customer business processes. At the same time, this results in a change from a transactional to a

[1] Cx = Challenge no.x

more relational relationship (Galbraith 2002, S. 194-207). However, the shape of these product-service bundles may vary a lot. On the one hand, there are simple maintenance contracts that are being sold with machines and on the other hand, Hybrid Products take the form of performance outcomes, which are connected to Service-Level-Agreement (SLA) guarantees (Burianek et al. 2007; Kersten et al. 2006). Providers are aiming for higher turn-over by taking over downstream value creation activities as well as higher profits.

Hybrid Products bear an especially high relevance in the IT sector. The offering of customer specific performance bundles, consisting of hardware, software and services, is a widely used concept in this industry (Böhmann et al. 2006). An example for such IT-services is the operation, maintenance and support of a large Webhosting solution. With the providers facing a growing complexity in providing such Hybrid Products, customers demand an even richer product variety and individualization with reduced prices at the same time (Kratochvil und Carson 2005). A resort to overcome these contradictory demands is the concept of standardization, which is the key to realize the economy of large scale reuse (Kratochvil und Carson 2005; Miller et al. 2002; Sawhney 2006) (C2). A method of standardization known in the literature is productizing, which means that reusable components are transferred into bricks and added to the knowledge base (Sawhney 2006) (C3).

In the next section, we will introduce the SCORE method, which addresses these three mentioned challenges providing Hybrid IT-Products. Additionally, we will present a short summary of a case study, which we will use to motivate the need to add IT support to the SCORE method. In the main part, we will describe the data model for the electronic Hybrid Product Catalogue (eHypCat) prototype. The Hybrid Product Catalogue is used as fundamental information database for the subsequent bid process.

2 The SCORE Method: A short summary

The SCORE method is a multi-level process enabling IT providers to identify standardizable, elementary elements in existing projects, aggregating these elements to modules (standardized bricks) and structuring these modules in a Hybrid IT-Product Catalogue (solution portfolio). Using this Hybrid IT-Product Catalogue the provider is able to recombine existing modules into individual, customer configured Hybrid IT-Products (Böhmann et al. 2008).

2.1 Case study at ACME

ACME is a medium-sized provider of web solutions. Typically, the firm designs, operates, and improves web-based information systems for its customers, primarily hosting customer systems in the firm's own data centers. In 2006, ACME employed about 70 people with an annual turnover of approx. 7m Euro. ACME's main customer base is dominated by medium-sized firms that run highly successful e-shops or online communities. These customers choose ACME because of the firm's substantial technical expertise and track record for superior quality of service. Thus, the ability to offer customer-specific web solutions is ACME's unique selling proposition to differentiate from standard internet systems providers that compete on price leadership. In the following sections, we will refer to the case study at ACME, introducing certain examples to illustrate the respective purposes.

2.2 The phases of the SCORE method

The SCORE method consists of the five phases "Objectives", "Component Analysis", "Modularization", "Customer individual Configuration" and "Learning and Development".

The phase "Objectives" is used to identify relevant categories, markets and target groups of the provider as well as strategic goals that should be resolved by building a Hybrid IT-Product Catalogue, e.g. improving component quality, reducing costs or keeping flexibility.

As soon as these objectives are determined, the "Component Analysis" phase begins. There, successful projects are being analyzed in order to identify reusable service, product components, and procedures as well as dependencies among them. Reusable components are stored in the DeliveryElement catalogue. In the case study at ACME, different complex e-commerce solutions were analyzed and it was found amongst others that ACME was adjusting an existing RedHat product (Linux operating system) in order to facilitate the processes of installation and update distribution in all appropriate projects. Formerly, reused elements were not stored and consequently they were just utilized by accident or if the responsible staff had experience with a similar project realized before. It is very important to note that not only adjusted products were stored in the catalogue, but also reusable processes, like "ACME OS installation", procedures like "Change Management" and customer integration cases like "Customer Requirements Engineering".

The subsequent modularization phase is used to create modules (solution bricks) by structuring identified DeliveryElements of the Hybrid IT-Product Catalogue. Creating modules is crucial in order to regroup DeliveryElements with strong dependencies into one module, whereas the dependency coupling between modules is loose (Baldwin und Clark 2000; Ulrich 1995). The SCORE method proposes four different module types resolving different DeliveryElement characteristics and a different grade of external factor integration (cf. (Böhmann et al. 2008). These modules are Process Performance Modules (PPM), System Providing Modules 1 and 2 (SPM1, SPM2), and Integration Modules (IM). The four module types are grouped into backend- (PPM, SPM1) and frontend-modules (SPM2, IM). While backend-modules are used customer independently in a Hybrid Product Factory, frontend-modules are characterized through the need of customer integration in the solution delivery.

First of all, procedures, which are neither customer-specific, nor system specific, are created as PPMs. These modules describe provider internal management procedures like Incident Management, Change Management, Project Management (PPM at ACME).

Then, SPMs are created. The SCORE method differentiates two types of SPMs: SPM1 and SPM2. SPM1 describes a module variant comprising the whole lifecycle of a Hybrid IT-Products, leaving little space for individual adjustments. SPM2 is a frontend module, which means that it's not highly specified and will be adjusted to the customer's individual requirements. At ACME SPM2 modules are used to define interfaces to yet unknown, but eventually needed systems that are required for certain customer projects. At ACME, this module keeps the information who is able to handle unknown database systems, for example.

IMs are used as interfaces to integrate unknown customer requirements, the always existent project part of customer individual projects. The usage of IM enables the provider to encapsulate services delivered from an external factor, e.g. the customer or a third party provider. Encapsulating delivery by a third party provider is very important for the results view, because the provider should be able to specify the quality of the Hybrid IT-Product as Service Level Agreements in the contract. One of ACME's projects, for example, required the usage of a third party webshop that was coded exclusively for the customer. Here, ACME only offered Service Level Agreements below the webshop application layer, e.g. the availability of the network connection of the server.

The case study at ACME showed that about 80% of customer requirements could be mapped on backend modules, whereas the rest of the requirements had to be specified in the project. The result of the Modularization phase is a module-based Hybrid IT-Product Catalogue. In the customer configuration phase, identified customer requirements are mapped onto these modules, which allow the provider to fulfill these requirements with a maximum of standardized components. This phase is not well elaborated yet. The SCORE method only provides guidance in terms of using the four modules, but it doesn't give a hint on how to identify customer requirements and to how map them on the Hybrid IT-Catalogue.

The last phase of the SCORE method is the "Learning and Development" phase. The SCORE method only succeeds if finished projects are analyzed in the end of a project and gained know-how is transferred to the knowledge base of the catalogue. For example, if a SPM2 module is used successfully in a project, it should be analyzed for reusability in future projects At ACME, the ORACLE database lifecycle might be defined in detail after a successful project as it is a candidate to be transferred into a SPM1 module.

3 A data model to add IT-support to the SCORE method

In the following section we will describe the application and outcomes of the SCORE method in the case study at ACME. The results are used to discuss the general need of IT-support for the SCORE method. As a next step, we will introduce the core data model used in the eHypCat prototype.

3.1 Utilizing the SCORE method in practice at ACME

The utilization of the SCORE method at ACME led to the definition of 111 modules comprising 287 DeliveryElements. At the beginning Microsoft Word was used as tool in the Component Analysis phase. We used separate Word templates to summarize analyzed projects, to identify and to describe DeliveryElements. Furthermore, dependencies between these elements were documented in Word and in Microsoft Excel (redundantly). Afterwards we started to utilize Word and Excel templates for creating modules, but we soon found out that the growing number of DeliveryElements led to an exponential growth in managing these elements (cf. Figure 1 – Quality). One reason for this is that changing information in one element entails a series of concurrent slow and error-prone changes in all dependant elements, e.g. changing the name of a process led to a change in all Excel templates holding dependencies to this process (cf. Figure 1 – Speed). Furthermore the usage of non-database dependant tools offers only a very small group of people to work on the project at the same time, because working on the same document at the same time is impossible and therefore teamwork is impossible.

Even if it was possible to create the Hybrid IT-Product Catalogue, the management of real world data exceeds the limits of Word and Excel. The continuous change of the IT DeliveryElements during the catalog's lifecycle requires the possibility of versioning DeliveryElements and Modules, e.g. if a Module is implemented at a customer site and later changed because of technology updates, the provider still needs to hold information on the deprecated version to assure the module delivery and management at the customer site. So, there even exist dependencies between different versions of one module (cf. Figure 1 – Manageability).

Additionally, there is a great demand for qualified dependencies with the possibility to apply fine grained rules. For example, a customer requiring a high availability solution should fire a rule changing the quantity model of the proposed solution. Even more complex rules could arise which leads to inclusions or exclusions of DeliveryElements or modules, hence such rules impact the customer's solution setup and architecture.

Table 1: Advantages utilizing the proposed IT prototype

Prototype Advantages	Comparison	
	Without developed prototype	**With developed prototype**
Quality	• Management is difficult for a large number of documents, which leads to error- prone synonyms and homonyms. • Missing Overview of the sum of all existing elements leads to failing module development. • Complex dependencies are not manageable. Only one-hop dependencies are stored in one document, therefore dependency checking leads to a multi-document traversal.	• Centralized database concept avoids redundant elements. • Dependency checking is made easy through visualization. • Application and verification of fine grained rules along different layers of the architecture becomes possible.
Speed	• Distribution of Word and Excel files as well as a missing versioning system impedes teamwork. • The effort on element management rises exponentially, e.g. an element name change entails the change of all dependant documents.	• The idea of database concept enables teamwork on the Hybrid Product catalogue in large teams. • Changes on elements are conducted only once and updated automatically through the catalog.
Manageability	• IT Architectures and therefore the Hybrid Product catalog underlie continuous changes in a lifecycle view: Versioning as a subsequent requirement is impossible with Word and Excel. • The selectable granularity for the elements is limited.	• Versioning is implemented. • XML export of the elements enables their exchange and facilitates manageability.

3.2 The Data Model Overview

Before introducing the core data model, we will give an insight on our general idea of modeling Hybrid IT-Products (cf. Figure). The main idea is to implement a four-layer architecture to support the idea of modularization, which is an extension to the original three-layer architecture (Böhmann 2004). The first layer represents the architecture layer, which is used to manage the DeliveryElements (elementary elements) of the catalogue, e.g. managing dependencies between DeliveryElements and Properties. Here, DeliveryElements are combined to modules. Modules form the configuration layer and assure different functions. On one hand, they form the smallest hybrid unit fulfilling a set of customer requirements over the lifecycle of a contract. On the other hand, different modules are combined together to meet all customer requirements in a customer individual product, the third layer of the architecture. Modules, as well as DeliveryElements are invisible to the customer to prevent unbundling and cherry picking. The key to map customer requirements on modules are FeatureTypes. Each of them describes a set of customer relevant properties of DeliveryElements, e.g. product properties like "VPN" or quality properties like "AVAILABILITY". The required properties are identified in the requirements engineering processes using e.g. rule based questionnaires during the bid process of Hybrid Products. Therefore the questionnaire as a sample method represents the requirements engineering layer. As a main contribution in the extension of the three-layer architecture, the utilization of this architecture considers the relationship between customer and provider: The customer expresses his requirements to the provider and is able to control if these requirements are met without knowing how the provider delivers these requirements (cf. Figure 1).

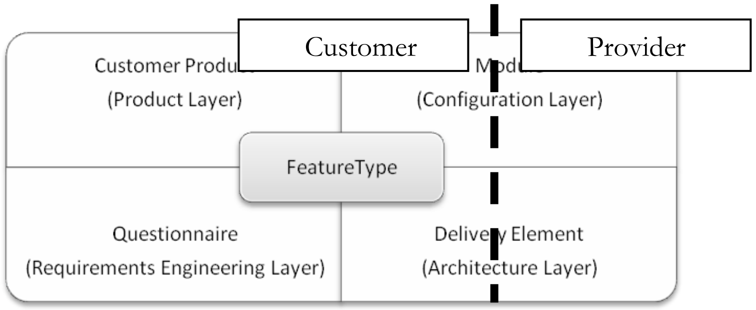

Figure 1: The data model overview of Hybrid Products

3.3 The Data Model in Detail

In the following subsection, we will describe the data model in detail and we will give some examples to illustrate the suitability for daily use (cf. Figure 2).

As stated above, the FeatureType is a key element of the data model. It describes a number of properties, whereas each property belongs to a single FeatureType and is described throughout a UnitType. Each Property is either a product property indicating the need for a product or a quality property indicating the need for services. For example, if the FeatureType is RAM, there are technical properties describing the amount of RAM as RAM-size like 512 MB RAM, 1024 MB RAM and properties describing the speed as RAM-speed like PC-2100. Furthermore all measurable Properties indicate the possibility to add quality warranties as Service Levels. If the FeatureType is network switch, a property like "Port" would be measurable in terms of availability: The ServiceLevelMeasureType used would be "Push", the ServiceLevelPlace would be inside the subnet and the ServiceLevelPeriod would be "One month". These attributes of ServiceLevelMeasure may require additional properties and subsequently DeliveryElements assuring the promised Service Level. This means that FeatureTypes are defined by Properties that hold dependencies to DeliveryElements, but there are also FeatureTypes that do not hold any properties. In this case FeatureTypes are categories summarizing a FeatureType subset, which represents the same family of functions, like network security.

A FeatureType category is named Offering Element. The sum of all Offering Elements defines the catalogue structure we call the Hybrid Product Catalogue. Offering Elements are important because each of them contains an amount of questions (QuestionBranch) with associated configuration rules in Object Constraint Language (OCL) needed for the identification of suitable properties configuring customer individual needs during the bid process. Through utilizing the RulePropertyRequirement association, rules can be applied to sort out unfitting properties to customer requirements. An Offering Element would be for example "PC Collaboration". This means that different people work together using the same data. One question in this context could be if people work together in one place or if they work together spread over the globe, e.g.at home. If people work together from their respective home offices, a rule has to be applied, which creates a requirement "need of a Virtual Private Network (VPN)", resulting in a subset of further questions (RuleQuestionRequirement) defining the exact properties.

As soon as all needed properties are defined throughout the questionnaire, an automated process of DeliveryElement resolution is executed. The aim is to identify all DeliveryElements that are needed to fulfill the specified Properties before. If, for example, a VPN for 100 people was needed, all VPN Firewalls (DeliveryElement) which are able to manage at least 100 VPN tunnels at the same time (featured in) would be selected. The result of this process is the FeaturePropertyDefinition covering all selected DeliveryElements. Dependencies between DeliveryElements are also represented through properties. The Association FeaturePropertyRequirement is used to describe these dependencies, e.g. the property operating system depends on the property server, which leads to the dependency between all DeliveryElements that incorporate operating system and server. The resulting DeliveryElements of the resolution form the collectivity of deliverables matching the standardizable requirements formulated by the customer. In this phase, the selection still holds competitive DeliveryElements (elements that fulfill the same requirement) as well as complemental, dependant and precluding DeliveryElements. Sticking to the example above, there exist different VPN Firewalls that meet the requirement of at least 100 VPN tunnels. Since another requirement might require high network availability and therefore specific services like remote management, these services might be applicable to only a subset of the identified VPN Firewalls reducing the amount of comparative DeliveryElements. The next process step is to exclude precluding DeliveryElements from the selection, which results in a selection covering compatible but yet partly comparative DeliveryElements which might be sorted in additional filters, e.g. pricing or quality filters. DeliveryElements are furthermore divided into Products, Software and Services. The design of Services is more demanding than the design of products or software, which are only described through Properties. In contrast to them, Services comprise the description of a process as a series of activities. The additional design of these activities is important, because operationalResponsibilities are stored there as well as input- and output documents and working plans: These additional items are important for both, the operations phase and the contracting phase providing e.g. cost relevant information. For example, the need to mandate a senior technician to fulfill activities is more expensive than mandating a young trainee.

Furthermore a big problem in delivering solutions is the missing integration in the phase of DeliveryElement development. Additionally, promising quality aspects like Service Levels requires the embedding of the "to be monitored Elements" in a lifecycle concept, incorporating their scope and state. These requirements are met with the introduction of Modules. Modules are the smallest hybrid elements of the catalogue from the viewpoint of the provider, but they are invisible to the customer to avoid unbundling. Every module is of a certain module type. Possible ModuleTypes are CANDIDATE, ABSTRACT, and MODULE. The ModuleType ABSTRACT is a template for a function family of modules. It defines a minimum set of required FeatureTypes. The set of these FeatureTypes is not necessarily an Offering Element: An ABSTRACT Module may require FeatureTypes of different Offering Elements, e.g. the ABSTRACT Module "SafeNetwork" may contain FeatureTypes like "VPN" from the Offering Element "PC Collaboration" as well as FeatureTypes like "Hotline" from the Offering Element "Customer Support". Utilizing ModuleTypes as templates for implementing Modules of the same kind helps to preserve the quality of the module variants belonging to the same ModuleType as its children, e.g. all Modules implementing SafeNetwork supply "Hotline Support". A Module CANDIDATE is used to describe a module, which uses an ABSTRACT Module as a template, but is not fully engineered yet. The CANDIDATE Module therefore represents a top-down approach in the module development. In this case, the product management decides to create a module with certain characteristics, e.g. a "Highly Available Firewall" which requires properties that are not met yet by existing Firewalls (DeliveryElements), like redundancy. This means that needed DeliveryElements must be developed before the CANDIDATE Module may be transferred to a Module and used in the catalogue.

Being the smallest unit for customer individual configuration, Modules incorporate a set of DeliveryElements delivering Properties. Following the propositions of the SCORE method, Modules exist in different levels of maturity. They can be either fully (SPM1, PPM) or partly (SPM2) specified. If they are partly specified, neither products nor service processes are explicated; there exists only a mapping between the ModuleType and the SPM2 Module as well as the definition of organizational roles and process outcomes (cf. Table 1). Modules are also used to integrate the external factor: Since interfaces between modules are documented as property-dependencies between them, the operationalResponsible OrgaUnit may be the customer or a third party supplier. Defining the role operationalResponsible is very important for Hybrid IT-Product contracts, because it helps to clarify rights and liabilities as well as consequences of unbundling and Service Levels in the contract. If the customer insists on self-delivery of a FeatureType like "VPN", the concept of unbundling forces him to self-delivery of all associated DeliveryElements in the same module as "VPN", so cherry picking on the customer side is made nearly impossible. The SCORE method proposes a fourth Module Type, the Integration Module (IM). Integration Modules are used to add unstandardizable customer requirements.. The analysis of projects at ACME has shown that in several complex IT projects there has been a highly individual part, which could not be anticipated before (about 20% of the projects). The existence of the Integration Module helps to combine both, the standardizable und the unstandardizable part of the project.

4 Conclusion

Managing data of Hybrid Products is relevant for companies that increasingly compete on the base of customer individualization. In this paper, we introduce IT support for the SCORE method in order to develop a Hybrid Product Catalogue from former projects in the IT industry. As main contribution of this paper, we extend the already existing Three-Layer Architecture invented by Böhmann. Through utilizing examples from the validation study, we explain the structure of the proposed data model enabling the creation of Hybrid Product Catalogues for the IT industry in detail.

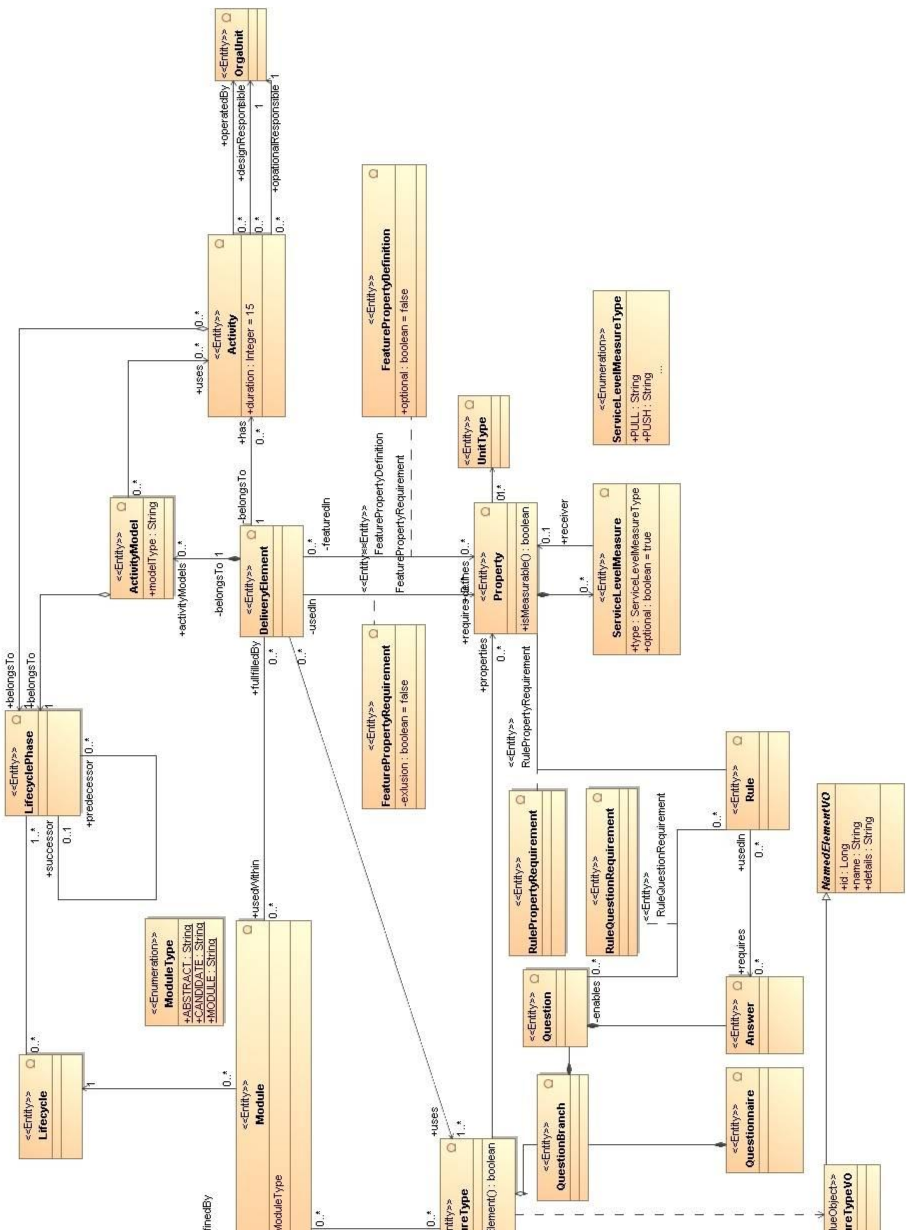

Figure 2: The data model for Hybrid Products in detail

References

Baldwin, C.Y.; Clark, K.B. (2000) The power of modularity, MIT Press, Cambridge, Mass.

Böhmann, T. (2004) Modularisierung von IT-Dienstleistungen - Eine Methode für das Service Engineering, Deutscher Universitäts-Verlag, Wiesbaden.

Böhmann, T.; Langer, P.; Schermann, M. (2008) Systematische Überführung von kundenspezifischen IT-Lösungen in integrierte Produkt-Dienstleistungsbausteine mit der SCORE Methode. Wirtschaftsinformatik, Vol. 50(Nr. 3).

Böhmann, T.; Taurel, W.; Krcmar, H. (2006) Paketierung von IT-Dienstleistungen: Chancen, Erfolgsfaktoren, Umsetzungsformen. Technische Universität München, Lehrstuhl für Wirtschaftsinformatik.

Burianek, F.; C., I.; Bonnemaier, S.; Reichwald, R. (2007) Typologisierung hybrider Produkte. München: Lst. Für Betriebswirtschaftslehre - Information und Management der TU München.

Foote, N.W.; Galbraith, J.R.; Hope, Q.; Miller, D. (2001) Making solutions the answer. McKinsey Quaterly, (Nr. 3), S. 84.

Galbraith, J.R. (2002) Organizing to deliver solutions. Organizational Dynamics, Vol. 31(Nr. 2), S. 194-207.

Johansson, J.E.; Krishnamurthy, C.; Schlissberg, H.E. (2003) Solving the solutions problem. McKinsey Quarterly, (Nr. 3), S. 116-125.

Kersten, W.; Zink, T.; Kern, E.-M. (2006) Wertschöpfungsnetzwerke zur Entwicklung und Produktion hybrider Produkte: Ansatzpunkte und Forschungsbedarf. In: Wertschöpfungsnetzwerke. Hrsg.: Blecker, T.; Gemünden, H.G. Berliin 2006, S. 189-202.

Kratochvil, M.; Carson, C. (2005) Growing Modular: Mass Customization of Complex Products, Services And Software, Springer, Berlin.

Miller, D.; Hope, Q.; Eisenstat, R.; Foote, N.W.; Galbraith, J.R. (2002) The problem of solutions: Balancing clients and capabilities. Business Horizons, Vol. 45(Nr. 2), S. 3-12.

Sawhney, M. (2006) Going Beyond the Product: Defining, Designing and Delivering Customer Solutions. In: Going Beyond the Product: Defining, Designing and Delivering Customer Solutions. Hrsg.: Lusch, R.; Vargo, S. M.E. Sharpe, Armonk, NY 2006.

Ulrich, H. (1995) The role of product architecture in the manufacturing firm. Research Policy, Vol. 24(Nr. 3).