FROM AN E-BUSINESS REVENUE MODEL TO ITS SOFTWARE REFERENCE ARCHITECTURE

Volker Gruhn, Thorsten Weber
Chair of Applied Telematics/e-Business, Department of Computer Science, University of Leipzig

Abstract: Revenue models define *how* a company creates their revenues and hence they are an integral part of business models. While a lot of research on business models and revenue models of the e-Business already exists, there is a shortfall of a concept to derive appropriate software architectures for the underlying software system directly from these models. This is interesting since a software system for companies of the e-Business is the fundamental basis to operate this business in practice. In this paper a concept is introduced to derive an important part of the overall software architecture for business models based on a characterization of its revenue model. For that purpose, a 'classification cycle' is defined including a set of criteria which enables you to conclude technical decisions for the design of a software architecture. Using this classification cycle, a variety of revenue models can be identified. Within this paper we focus on the specific revenue model subscription of services as one example. The derived architecture serves as a software reference architecture for all business models which are based on this type of revenue model. It means that in case a software system has to be developed for a business model using this revenue model, the software architecture presented in this paper can be used as a sample solution to refer to. Thus, it helps architects to derive a fundamental part of the overall software architecture in an easy and efficient way.

Key words: business model, revenue model, software reference architecture, e-Business

2. RELATED WORK

Zerdick et al. (1999) define the revenue model as the 'determination of the sources of revenue'. They investigate revenue models for the e-Business isolated from business models. This approach is also pursued by Skiera and Lambrecht (2000). However, they classify possible revenue models within e-Business but do not analyze the relation of these models to the underlying software architecture.

Other authors use a different approach by defining revenue models as a part of the encompassing business model. One of the first definitions of a business model in e-Business is given by Timmers (1998, 1999). According to him, 'a business model is defined as the organization (or 'architecture') of product, service and information flows, and the sources of revenues and benefits for supplier and customer'. It becomes obvious that the revenue model that Timmers calls 'sources of revenue' is an integral part of a business model. Buchholz (2001) defines a business model as a 'brace of four constitutive components', where the revenue model is one of them. Also Wirtz (2001) and Doubosson-Torbay, Osterwalder and Pigneur (2001) see revenue models as a part of business models.

However, we could not identify any research that analyzes the dependency of business models and its related software architecture in depth. Only a few authors deal with this relation at all. One approach in this area is delivered by Bartelt and Lamersdorf (2000). They define four methods for the designing of business models. One of these methods are so called 'modules of functionality' like product-catalogues and search engines. They are identified within business models and can be used independently from a precise case and thus can be reused in other business models. However, this approach does not deliver a set of such functionality modules and identifies only some examples which cannot be used in general. Furthermore, they do not investigate these modules of functionality for the domain of the revenue generation.

Due to this shortfall we chose an alternative approach that is described in the next chapters to derive a software reference architecture on the basis of a characterization of a revenue model. A software reference architecture can be interpreted as 'a collection of computational components [...] together with a description of the interactions between these components – the connectors' (Gerlan and Shaw, 1993). Thereby, multiple descriptions can be given depending on different aspects one wants to focus on. We describe in this article our software reference architecture using UML class diagrams (OMG, 2005) to place emphasis on static aspects, and sequence diagrams to place emphasis on dynamic aspects of the system.

1. MOTIVATION

e-Business is characterized and influenced both by the usage of information technology for business purposes and by the occurrence of adapted or completely new generated business models. To lead a company successfully, there has to be a clear understanding of how the revenues can be generated. The offered value a customer is willing to pay for as well as the related processes of revenue generation are defined in the revenue model. Thus, a revenue model is one of the core elements in planning and realizing a company. Analyzing the literature, it becomes obvious that there is a common understanding that a revenue model is an integral part of a business model.

To realize a business model in the e-Business, there is a special focus on the software architecture of its underlying software system, because, like the 'e' suggests already, the processes are supposed to be performed mostly electronically. Thus, the business model and its underlying software architecture are tightly connected. Since the revenue model is a very important element of the business model, the software architecture also has to be designed according to the requirements which have to be satisfied to generate revenues. There are other elements of a business model like its procurement or distribution model that influence the overall software architecture of the complete software system, but we focus on this fundamental aspect.

The purpose of this article is to develop a software reference architecture to support the application of one revenue model within its encompassing business model. Thus, the reference architecture helps architects to derive an important part of the overall software architecture which is necessary to realize a business model. Within this article it will be focused on the subscription revenue model as one possible revenue model in e-Business. It can be applied to different business models like Internet-Service-Provider, online magazines or even e-shops which offer an ongoing claim of their products using a subscription.

Therefore, the revenue model has to be characterized in a way which enables a company to derive technical decisions for the design of this software architecture. Because of the mutual dependency between the revenue and the business model, the criteria for this characterization have to be based on the business model itself. This article shows a way of deriving a software architecture on this background. Using the resulting software reference architecture for the subscription revenue model in practice, the planning, (re-)designing and implementing of new applications for business models becomes more reliable and effective.

Since we claim to derive a reference architecture, this term yet has to be defined. More generally than an architecture, a reference model is according to Bass, Clements and Kazman (1997) 'a standard decomposition of a known problem into parts that cooperatively solve the problem'. Then, a (software) reference architecture is 'a reference model mapped onto software components [...] and the data flow between these components'. Related to our article, the known problem is represented by the revenue models. This domain is decomposed into functional parts which can be implemented as components and their relations. Thus, a sample solution is given that can be used as a reference during the development of a software architecture related to this domain.

3. CLASSIFICATION OF THE SUBSCRIPTION REVENUE MODEL

Subscription is taken as the relevant e-Business revenue model for this article. Other examples of revenue models are transaction fees, advertising or profiling, but here we focus on the subscription revenue model. In general, a subscription represents a contract between a supplier of an offer and its customer (the subscriber) about the claim of a specific amount of a specific offer within a specific period for a specific price.

The agreed amount represents the maximum the customer is allowed to obtain during the period. In the case that this amount is exceeded, additional entities are charged separately. In the majority of the cases, the subscriptions are extended automatically and the price is paid per period.

In this section, the revenue model subscription and its variations are characterized. Therefore, some key criteria are identified with regard to the purpose of reasoning design decisions of an appropriate software architecture. Using these criteria the classification of the revenue model enables two things:

- to identify different variations (or subtypes) of the revenue model and
- to be able to derive decisions for the software architecture of the underlying software system which enables the application of this revenue model.

For the classification of the revenue model, the 'classification cycle' is used. The following paragraph introduces this tool.

3.1 The classification cycle

Related to the elements of the definition of a business model the classification criteria to characterize a revenue model are grouped into four sections: actor-related, offer-related, benefit-related and revenue-related criteria. The close dependencies between the revenue model and its business model are considered via this approach. The characterization of the revenue model is based on criteria that are linked to its encompassing business model. Nine criteria were collected as a result of related work research and own considerations. In the following, the criteria are introduced in brief.

3.1.1 Actor-related criteria

The actor-related criteria are the customer role and the customer relation. The customer role may obtain one of the parameters informant, buyer, seller, or value integrator (actors within communities where the customer itself generates the value). This criterion limits the possible ways of receiving money from the customer. It is not limited that a customer only acts in one role. For example, a customer on a market platform may sell his own products and buy products from other participants. The important question from a software architecture point of view is whether a customer role has to be supported or not. The customer relation might be anonymous, identified or identified and authenticated. An online bank for example has to be sure that the customer is the person he claims to be. Therefore, special authentication procedures have to be supported in this case. In general, depending on this criteria the design of customer profile management and access control is influenced.

3.1.2 Offer-related criteria

Offer-related criteria are consistency, pricelevel and origin of the products or services. The consistency is one of the most important criteria since it provides information about the ways the company has to manage their products internally and how the shipment can be handled. Possible values are services, digital goods, physical goods, or information. The pricelevel ranges from nano via micro and medium to macro[1] and gives information about possible payment methods. The origin of the offer may either be self-determined or over-directed. This differentiation is important

[1] According to Reif (2001), the ranges are (in €) nano (0,001 to 0,1), micro (>0,1 to 5), medium (>5 to 1000) and macro (>1000).

for devices which the product management component has to maintain and to control.

3.1.3 Revenue-related criteria

The revenue-related criteria comprise the payment method and the revenue origin. The origin of the revenues can be direct or indirect and is addressing either the end-consumer (direct) or third parties like other companies (indirect). The latter parameter occurs for example in advertisement-based revenue models. As we mentioned earlier already, payment methods are influenced among others by the pricelevel of the offering. A company has to decide whether its customers are claimed using invoices, direct debits, credit cards, or external payment providers which offer for example a method to transfer prices also on a nano or micro level.

3.1.4 Benefit-related criteria

The last section is the benefit related criteria. Two criteria can be found here: the primary benefit and the additional benefit for a customer. The primary benefit does not have static values but answers the question 'what is the customer willing to pay for?' This criterion can thus be interpreted as the revenue model in a nutshell and contains the most important value from a customer point of view. The additional benefit can be e-Business-inherent, personalization or anonymity. In the case that one of the last two values is selected, the design of the software architecture has to consider this fact strongly. Otherwise, the e-Business-inherent means nothing but benefits that are encompassed anyway by using the e-Business technology like time savings due to independency of physical distances or convenience by ordering products simply using a PC. In this case, nothing has to be considered in particular.

This total of nine criteria, grouped in their sections and arranged in the classification cycle are used to characterize a revenue model. This classification cycle, which can be seen in Figure 1, is a general tool to describe revenue models of the e-Business. However, within this article, it is only applied to classify the subscription revenue model. Thus, only a limited set of all parameter values will be used.

As it will be classified in the following, we concentrate only on one variation of the subscription revenue model: subscription of services. Other variations like subscription of digital products or subscription of physical products are not within the scope of this article. Nevertheless, it becomes

obvious that the product consistency is the most relevant criterion of subtypes of the subscription revenue model.

3.2 Classifying the subscription of services revenue model

The classification cycle in figure 1 shows the characteristics of the subscription of services revenue model.

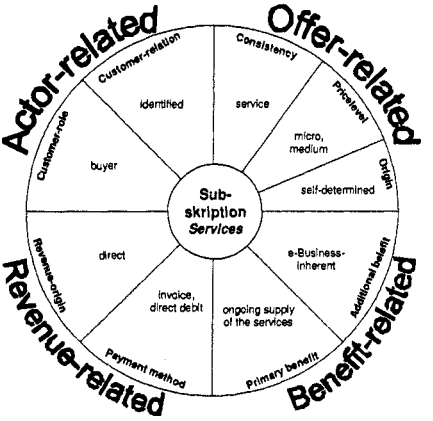

Figure 1. Classification cycle of the revenue model subscription of services

A typical example for a business model which offers services to subscribe is an Internet Service Provider who offers web-access to its customer. Based on this example, the following considerations are explained.

4. FROM REVENUE MODEL TO SOFTWARE REFERENCE ARCHITECTURE

Based on this characterization of the subscription of services revenue model by using the classification cycle, technical decisions for a software reference architecture are derived within this section.

The company offers services (criteria consistency). The primary benefit for the customer is the ongoing supply of these services. Therefore, he subscribes to this service by concluding a contract. These criteria obviously affect the way how the offers have to be maintained by a software system. In

addition, the origin of the offer is self-determined. This leads to the requirement that the company has to create, change and maintain its services by itself. For the given example of an Internet Service Provider, the company has to offer a variety of different tariffs for its customers to meet a wide range of different whishes depending on the personal behavior of each customer. The tariff determines the kind of service the customer subscribes and is volume or time related. The maximum available amount of units for each kind has to be fixed by the tariff plus the time period in which these units have to be consumed. Furthermore, the tariff defines the price for this package and the costs of additionally claimed units within one period. All these parameters have to be fixed within a tariff. While the contract itself contains personal information about the customer and maybe determines general issues like e.g. the payment method, the tariffs contain the detailed description of the service. Therefore, contract and tariffs belong to one component but will be realized in separate classes. A selected tariff is then aligned to a contract.

To be able to distinguish various statuses of a customer this is a relevant information that has also to be stored in the contract. By providing a status, it is possible to lock out users from using any services if it is required, for example if they did not pay their last bill correctly. Thus, the risk of betrayal can be reduced for the company.

It also became obvious, that depending on the tariff different reference values (volume or time) have to be charged and therefore have to be logged. Since the customer relationship is identified, each customer session has to be investigated separately regarding the activities of the user. At the end of a session, a logging mechanism must determine the relevant reference value and the consumed entities and save it persistently. For revision purposes and to create itemized bills, each session has to be saved separately. Therefore, an appropriate usage account has to be realized in a separate component. This component performs the central processes to charge the customer in accordance to the use of the subscription revenue model.

Obviously, customer profiles have to be supported as well. That the origin of the revenue is direct can be seen at the criteria. This means that the end-customer transfers the revenues. They act as identified buyer who pay their bills by using invoices or direct debits. In addition, there is no need for any anonymity assumed to be an additional benefit[2]. Thus, the customer disposes of a customer profile. Access data like login and password and private address data have to be saved there. Furthermore, a debit account, saving all financial transactions, is necessary within the customer profile.

[2] In case the customer relation is classified as identified, it is hardly possible in practice that there the additional benefit might be anonymity.

This account will be periodically charged with the amount of the invoice (therefore debit account) and in return will be credited on the event of incoming financial transactions. Since a history of all transactions has to be available, each transaction has to be stored in an own data record. Thus, the account will be realized by two classes where one class (better: the instances of one class) contains the data records. The complete customer profile will be realized by a further component which contains cooperative classes.

The invoice was mentioned already. The invoice has to determine and sum up the consumed entities and compare this value with the customer's chosen tariff at the end of a period. In case the summed up value extends the maximum count of entities, the difference has to be calculated separately based on the defined price. Otherwise, the invoice will show the defined price for the subscription. The final amount will be charged and added as a new data record of the customer's debit account and a physical version of the invoice has to be created and send to the customer. An own component will implement the invoice. This component will gather the relevant information from the customer profile component, the contract and tariff component, and the usage account component. Therefore, references between these components have to be considered in the reference architecture.

To summarize: the following components were derived based on the classification of the revenue model:

- a contract and tariff component which separates both elements in own classes
- a usage account component which is responsible for the logging of the user activities depending on the tariff; this is the central component to charge the customer using this revenue model
- a customer profile component which comprises classes for personal data and for debit account data
- a invoice component

These considerations are picked up again in the next chapter where a detailed appropriate software reference architecture is represented.

5. A SOFTWARE REFERENCE ARCHITECTURE FOR THE SUBSCRIPTION REVENUE MODEL

According to the views defined by Gruhn and Thiel (2000), the software architectures are described on a software-technical level in the following. Since it is a goal of the represented reference architectures to identify relevant components which realize necessary functionality to implement the

revenue model, these components are identified within the software-technical architecture by encompassing their belonging classes.

The following figure 2 shows the class diagram of the reference architecture. The grey boxes encompass the classes to their related components.

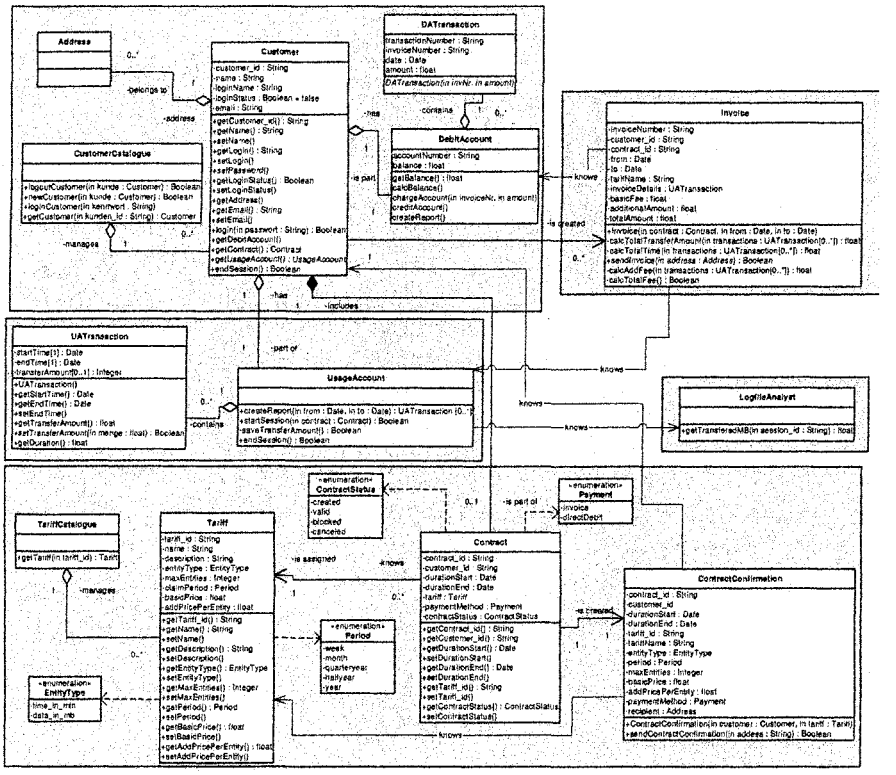

Figure 2. Class diagram of the software reference architecture

As it can be seen, five components are defined, while in the previous section only four main components were derived from the classification cycle of the revenue model. The reason for this is that the UsageAccount component needs additional services from another component called *LogfileAnalyst*. Before we analyze its role in more detail, we look at the other components.

The *CustomerProfile* component contains five classes which should be largely self-explaining. A user can have multiple addresses that lead to the creation of an own class. The *DebitAccount* contains header information like the current balance. The transactions are stored as instances of *DATransactions*. Therefore, the constructor of this class is activated by the

DebitAccount, handing over the relevant information which it received earlier by the invoking component, e.g. the *Invoice*. In addition, a *CustomerCatalogue* is necessary to create new instances of *Customer* or to deliver a reference of a customer object to an invoking class.

The *ContractAndTariff* component contains the separated classes *Contract* and *Tariff* as required, while an instance of *Tariff* is assigned to an instance of *Contract*. A class *TarifCatalogue* is similar to a *CustomerCatalogue* and delivers a reference to a *Tariff* object. A further class of this component is the *ContractConfirmation* which will be generated by the time the customer places the contract. It contains contract-relevant data and will be sent to the customer.

As mentioned earlier, the central component related to the core process of this revenue model is the *UsageAccount* component. With the input of the *LogfileAnalyst* component this component reports the activities within a session and stores them persistently into a separate data record. Figure 3 shows this component.

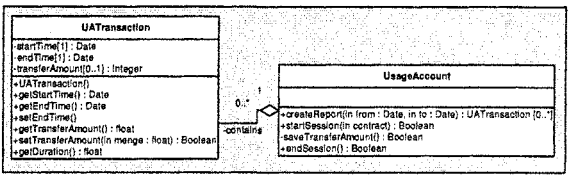

Figure 3. UsageAccount component

The process of the logging and the roles of the participating classes is shown in detail in the sequence diagram in figure 4. By the time a customer logs in and a new session is started, the method *startSession* of the class *UsageAccount* is called. It controls at first the status of the contract to ensure the customer is allowed to consume the services. Assuming a positive response, the *UsageAccount* initiates a new instance of the class *UATransactions*.

The data which have to be logged depend on the tariff the related customer had chosen. Either the transferred data volume or the connected time is relevant. In either case, by the time the customer logs out and ends the session, the relevant information has to be determined and saved within the *UATransactions* data record. Therefore, the method *endSession* of *UsageAccount* is invoked. In case the transferred data volume has to be recorded, this information has to be determined at first. Therefore, the component *LogfileAnalyst* can be used, which gathers the information by parsing the log file. This information can be saved afterwards within the

created instance of *UATransaction*. The duration of the session is always saved in the data record independently of the tariff for reporting reasons.

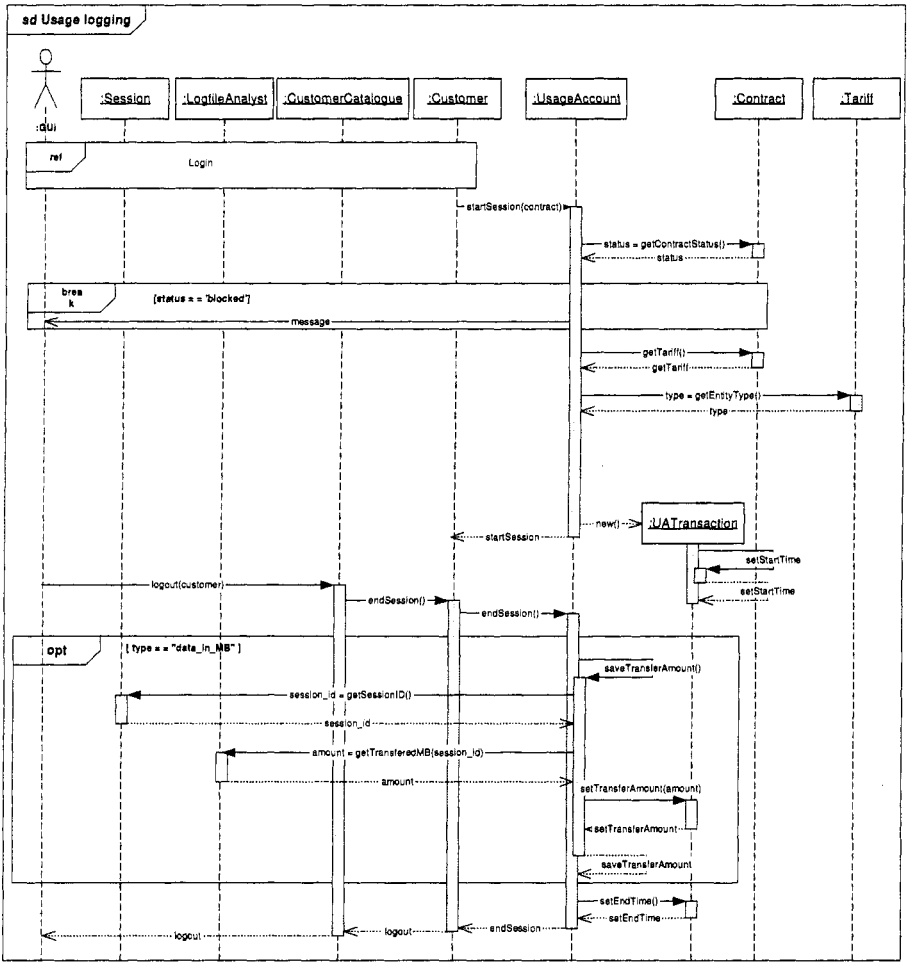

Figure 4. Sequence diagram of the usage logging

The last component to be discussed is the *Invoice* which comprises one equally named class. The constructor of this class is called at the end of each subscription period from a controller of the complete system which is not described here in any detail. The constructor receives the relevant parameters to calculate the chargeable amount which are a reference to the relevant customer, its contract, and the invoice period. Figure 5 shows the interaction of the participating components during the invoicing.

For the calculation, the tariff chosen by the customer has to be compared with the used entities during this expired period.

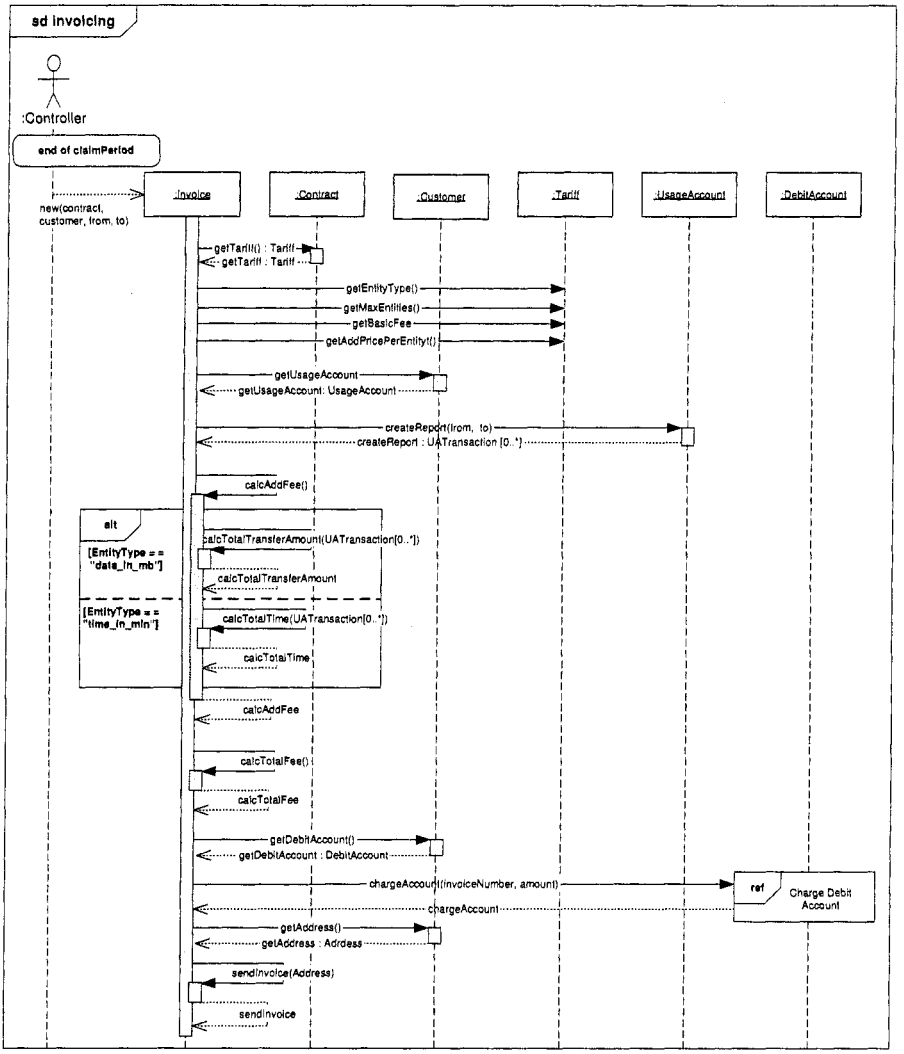

Figure 5. Sequence diagram of the invoicing

The tariff is received from the *ContractAndTariff* component, the used entities from the *UsageAccount* component. If the amount of consumed units is not higher than the maximum amount according to the tariff, only the agreed subscription fee has to be charged to the *DebitAccount* of the customer. In case the used entities exceeded this maximum, an additional price has to be calculated by the invoicing and charged to the account. At the end, a physical invoice has to be sent to the customer.

6. VALIDATION

A German Lottery company was rebuilding its internet platform recently. One of the offered games on the platform can be played in a subscription mode, so that the customer automatically takes part on the draws according to his configured parameters like draw days, predictions, and bet amounts. To ensure the participation, the stake has to be available on the customers account in advance. This virtual account is part of the customers profile. By the time the bets are placed into the system, the stake is withdrawn automatically from this account. To agree on the conditions of the contract, the customer has to sign on to the platform and configure his subscription.

The design of the platforms software architecture was derived from the reference architecture presented in the previous section. Because of the divergent specification of bets, some components had to be adapted slightly. The *ContractAndTariff* component had to be changed in a way to treat the individually configured bets as tariffs. Thus, a bet was assigned to a contract rather than a tariff. Furthermore, the *UsageAccount* and the *Invoice* components were adapted, because each participation on a draw had to be logged and the regarding bet amount had to be withdrawn from the customers account immediately. Nevertheless, the usage of the software reference architecture was helpful to design the required architecture with reduced effort and in reduced time.

7. CONCLUSIONS

This paper focused on the creation of a software reference architecture for a revenue model of the e-Business based on a business model related characterization of this revenue model. Therefore, we introduced a classification cycle including relevant classification criteria. The parameters of these criteria, selected in accordance with a specific revenue model, enabled us to derive requirements and conclusions for the design of an appropriate software architecture. Because we focused on the domain of the subscription of services revenue model, the software architecture can be considered as a reference architecture. The benefit of this software reference architecture was proven already during its application within a practical software development project.

At this point, wee see the need of further research in order to extend the usage of the classification cycle to further revenue models and to derive more appropriate software reference architectures. It is our aim to get a comprehensive set of reference architectures which can be used to a large variety of business models within the e-Business.

ACKNOWLEDGEMENT

The Chair of Applied Telematics/e-Business is endowed by Deutsche Telekom AG.

The authors can be contacted at {gruhn, weber}@ebus.informatik.uni-leipzig.de

REFERENCES

Bass, L., Clements P.C., and Kazman, R., 1997, *Software Architecture in Practice*, Addison Wesley

Bartelt, A., and Lamersdorf, W., 2000, Business Models of Electronic Commerce: Modeling and Classification (German: Geschäftsmodelle des Electronic Commerce: Modellbildung und Klassifikation), in: *Verbundtagung Wirtschaftsinformatik 2000*, pp. 17-29, Shaker

Buchholz, W., 2001, Netsourcing Business Models – Business Models for purchasing platforms (German: Netsourcing Business Models - Geschäftsmodelle für Einkaufsplattformen), in: Dangelmaier, W., Pape, U., and Rüther, M., ed., *Die Supply Chain im Zeitalter von E-Business und Global Sourcing*, pp. 37-52

Dubosson-Torbay, M., Osterwalter, A., and Pigneur, Y., 2001, eBusiness model desing, classification and mesurement, (October 5, 2004) http://citeseer.ist.psu.edu/dubosson.torbay01ebusiness.html

Garlan, D., Shaw, M., 1993, An Introduction to Software Architecture, in: *Advances in Software Engineering and Knowledge Engineering*, pp. 1-39, World Scientic Publishing Company

Gruhn, V., and Thiel, A., 2000, *Componentmodels - DCOM, JavaBeans, EnterpriseJavaBeans, CORBA* (German: *Komponentenmodelle - DCOM, JavaBeans, EnterpriseJavaBeans, CORBA*), Addison-Wesley

Object Management Group: OMG Unified Modeling Language Specification, March 2005, Version 2.0.

Reif, W., 2001, What is E-Commerce? (German: Was ist E-Commerce?), University of Augsburg, (May 5, 2003) http://www.uni-augsburg.de/lehrstuehle/info1/lehre/ss01/e-ommerce/folien/Definition.pdf

Skiera, B., and Lambrecht, A., 2000, Revenue Models for the Internet (German: Erlösmodelle im Internet), (December 13, 2003) http://www.ecommerce.wiwi.uni-frankfurt.de/skiera/publications/Erloesmodell.pdf

Timmers, P., 1998, Business Models for Electronic Markets, in: Gadient, Y., Schmidt, B., Selz, D., ed., *EM – Electronic Commerce in Europe. EM – Electronic Markets*, Vol.8. No. 2, 07/98, (May 10, 2005) http://www.electronicmarkets.org/modules/pub/view.php/electronicmarkets-183

Timmers, P., 1999, *Electronic Commerce – Strategies and Models for Business-to-Business Trading*, John Wiley & Sons Ltd. England

Wirtz, B.W., 2001, *Electronic Business*, 2. Auflage, Gabler

Zerdick, A., et al., 1999, The Internet Economy – Strategies for Digital Business (German: Die Internet-Ökonomie - Strategien für die digitale Wirtschaft), Springer