Specification Proposals for Customizable Business Components

Jörg Ackermann

University of Augsburg, Universitätsstraße 16, 86135 Augsburg, Germany, E-Mail: joerg.ackermann.hd@t-online.de

Abstract. Compositional plug-and-play-like reuse of black box components requires sophisticated techniques to specify components. A solution how to specify business components formally was proposed in the memorandum „Standardized Specification of Business Components". So far not considered was the situation when a component can be tailored to user requirements by setting parameters (customizing). In this paper we discuss how the ideas of the memorandum can be extended to formally specify customizing options of business components.

Keywords: Component Based Software Engineering; Business Component; Business Application System; Formal Specification; Adaptation; Customizing

1 Introduction

Combining off-the-shelf software components offered by different vendors to customer-individual business application systems is a goal that is followed-up for a long time. One of the crucial prerequisites towards this goal is an appropriate specification of business components, since the specification might be the only available support for a composer who combines components to an application system. The memorandum „Standardized Specification of Business Components" proposes how to specify business components formally, cf. [Turo2002]. (For details see section 2).

The construction of business applications from components promises solutions that are flexible and well tailored to user requirements. Nevertheless there will still be a need for adaptation of single components. One prominent technique for planned adaptation is customizing, that means setting of predefined parameters (for details see section 3). Is a business component customizable, its customizing options and consequences must be apparent for a component consumer and therefore need to be specified.

The memorandum did so far not consider the situation when a business component allows adaptation by customizing. In this paper we address the problem how customizable business components can be specified.

We identify which aspects of customizing need to be specified and then propose how they can be specified. We will transform customizing data to an interface information model and enable to specify customizing options within the frame of the memorandum. By this approach the basic structure and the notation mix of the memorandum will be preserved and only some enhancements will be necessary.

2 Memorandum „Standardized Specification of Business Components"

To enable a common understanding component specifications need to be standardized. This is the main goal of the memorandum „Standardized Specification of Business Components". The content of the memorandum is a recommendation made by the business components working group of the German Informatics Society (GI). The English version of the memorandum is accessible via Internet, cf. [Turo2002]. Documentation about practical experiences can also be found there ([Acke2001] and [FeLT2001]).

The aim of the memorandum is to set a *methodical* standard for the specification of business components. This is achieved by identifying the objects to be specified and by defining a notation mix that is standardized, accepted and agreed upon by all participating parties. The term *specification* of a business component is defined as a complete, unequivocal and precise description of its *external* view, that is, it describes which services a business component provides under which conditions.

Based on the ideas of [BJP+1999], [Turo1999] and [Turo2001b], the memorandum defines different contract levels for the specification of components. Besides arranging the specifications' contents according to contract levels, for all of these levels a specific notation language is proposed (see fig. 1).

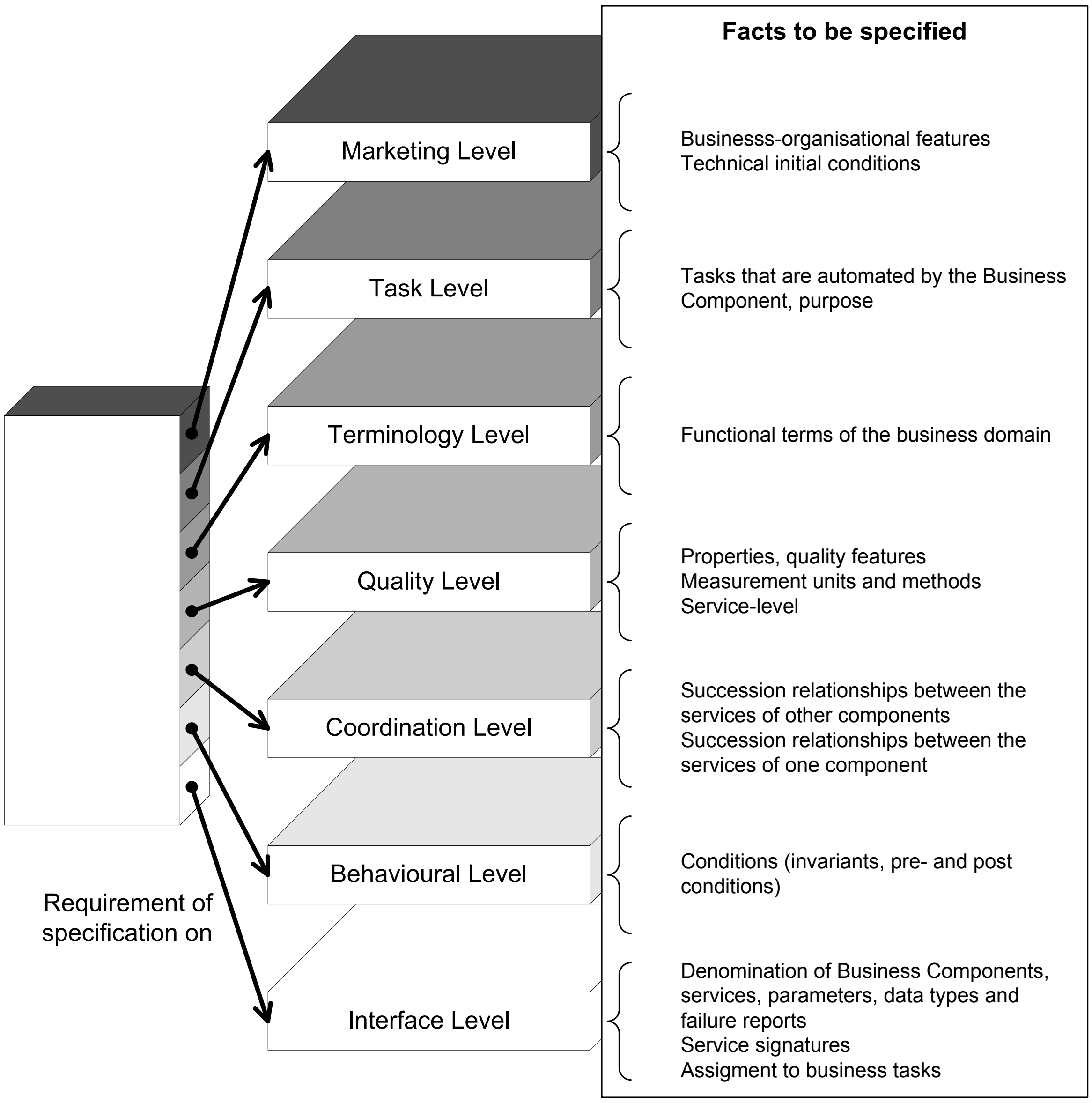
Facts to be specified
Marketing Level
Businesss-organisational features
Technical initial conditions
Task Level
Tasks that are automated by the Business Component, purpose
Terminology Level
Functional terms of the business domain
Quality Level
Properties, quality features
Measurement units and methods
Service-level
Coordination Level
Succession relationships between the services of other components
Succession relationships between the services of one component
Behavioural Level
Conditions (invariants, pre- and post conditions)
Interface Level
Denomination of Business Components, services, parameters, data types and failure reports
Service signatures
Assigment to business tasks
Requirement of specification on

Figure 1: Software contract levels and facts to be specified

For a wide-spread acceptance in practice the memorandum recommends well-known and well-accepted formal notations as OMG IDL or UML. (Note that UML 2.0 details were not available at the time of discussion in 2001 and will be considered in the next version of the memorandum.) The goal of the memorandum is not to invent new notations but to combine existing ones in order to enable a complete and standardized specification. Suitability in practice received special attention.

Below, the different contract levels are going to be characterized and supplemented with the proposed specification method. Note that we start with the more technical levels, that is in reversed order of fig. 1.

The *interface* (or syntactic) *level* contains basic agreements as names of services and data types, signatures of services, as well as the declaration of error messages. The OMG IDL [OMG2001a] has gained a broad acceptance as a standardized notation for the interface level and is therefore our notation of choice. The resulting contract guarantees that service client and service donator can technically communicate with each other. Semantic aspects remain unconsidered.

Agreements at *behavioral level* serve as a closer description of the behavior and describe how a given component acts in general or in borderline cases. As an example, we could define an invariant condition for a business component *stock keeping*, which says that the reordering quantity for each (stock) account has to be higher than the minimum inventory level. The Object Constraint Language (OCL), which complements the UML, is an example for a widespread notation to specify behavioral facts ([OMG2001b]). The UML (together with the OCL) is especially recommended to specify components, cf. [AlFr1998], [DSWi1999] and [ChDa2001]. Therefore we use the OCL as notation on the behavioral level.

Agreements at *coordination* (or synchronization) *level* regulate the sequence in which services of a business component may be invoked, and synchronization demand between its services. Here, e.g., we may lay down that a minimum inventory level has to be set before it is allowed to book on a (stock) account for the first time, or that it is not allowed to carry through more than one bookkeeping entry at the same time for the same account. The coordination level is also used to regulate the sequence in which services of *different* business components may be invoked. To restrict the number of used notations the OCL shall be used again. The OCL, however, is only partially suited to specify coordination issues. Therefore the OCL was enhanced by temporal operators, cf. [CoTu2001]. By using temporal operators one can specify requirements about the order in which different services can be performed.

As an extension to functional characteristics, we have to describe non-functional characteristics of business components. These are specified at the *quality level*. Examples for such characteristics are the distribution of the response time of a service or its availability.In all levels mentioned so far, the specification uses technical terms, which have a domain specific functional meaning (semantic), e.g. stock, inventory level, or account. Often, these terms do not have an unequivocal meaning or definition and, hence, have to be specified to guarantee their unequivocal use. The *terminology level* serves as central registry and keeps all used terms and their definitions in a dictionary. To achieve a high quality of specification, we use norm language reconstruction (cf. [Ortn1997]) to specify the respective issues. Norm language reconstruction is characterized by a dictionary of unequivocally defined functional terms and by using a rational grammar (reconstructed grammar of natural, colloquial language) to form the statements. A rational grammar consists of patterns and stencils to compose sentences.

We use the same technique to specify issues at the *task level*. There, we explain which business tasks are supported or automatically done through services offered by a business component.

At the *marketing level* we finally specify features of the business component that are important from a business-organizational point of view, e.g., (legal) contract terms, version, coarse business domain, or vendor contact persons. As notation predefined tables are used.

To summarize, the main contributions of the memorandum are given by the facts that it

- considers all specification relevant objects in one unified proposal,
- supports the connection to business domain models (business tasks, business terms),
- identifies suitable contract levels,
- proposes appropriate notation languages for each contract level and provides guidelines how to use them
- and extends the OCL by temporal operators to satisfy specification needs on the coordination level.

3 Customizable Business Components

The construction of business applications from components promises solutions that are flexible and well tailored to user requirements. Nevertheless there will be a need for adaptation in component based systems, cf. e.g. [Turo2001a] or [Reus2001].

The topics reuse, adaptation and variability for software in general are broadly covered in [JaGJ1997]. The authors define so called variation points, where the adaptation takes place, and identify the following adaptation mechanisms: inheritance, extensions, uses (use case reuse), configuration, parameters, template instantiation and generation. All of these mechanisms work (in accordance with the author's aim) at the implementation level.

Adaptation in component based systems is discussed by many authors, cf. [BRS+2000] or [Reus2001]. One can identify two general adaptation strategies: adaptation of component composition and architecture, and adaptation of single components. [BRS+2000] identifies the following adaptation mechanisms: wrapper, composition with adaptor, adaptation interface, inheritance and reimplementation. [Reus2001] additionally identifies superimposition and parameterized contracts.

For the adaptation of single components we distinguish between planned and unplanned adaptation. *Planned adaptation* means that adaptation opportunities are predefined by the component developer. This requires that the techniques to perform the adaptation are provided by the component developer as well. Known techniques for planned adaptation include maintenance of initializing files, maintenance of parameters in data base tables and programming of user exits. *Unplanned adaptation* means that adaptation opportunities are not predefined by the developer. Possible techniques are e.g. inheritance and reimplementation. Unplanned adaptation does not correspond to easy, plug-and-play like reuse of black box components and hence will not be considered further.

One technique for planned adaptation is customizing, that means setting of predefined parameters. Adaptation by customizing does not require knowledge about implementation details and fits well to our model of easy black box reuse. Moreover, customizing is a widely used technique for adaptation of standardized off-the-shelf business software like SAP R/3. From this we conclude that customizing is well suited for adaptation of business components and will play an important role.

Typical customizing settings are e.g. definition of organizational units (like plants), definition of control parameters (like working hours of a plant for production planning) and choices between different variants of business processes.

In the rest of the paper we focus on customizing and will now define some terms more precisely.

Under *Customizing* we understand the planned adaptation of a business component, which is based on

- assigning (at configuration time) values to predefined data fields
- that influence structure and behavior of the business component and
- have an impact on many business transactions.

Data fields provided for customizing are called *customizing fields*. We use the term *customizing data* when we want to address the sum of all customizing fields as a whole. A business component is called *customizable* if it allows adaptation by customizing.

It is sometimes not evident how to distinguish between customizing data and master data of an application, cf. e.g. [ApRi2000]. To help with the distinction we propose the collection of characteristics in Table 1. Note that the given characteristics in both columns shall not be considered as black-and-white, but rather as the opposite ends of a continuous spectrum. Table 1 is supposed to help in a classification, but in praxis not each of the characteristics needs to apply completely.

		Customizing data	**Master data**
Time of definition		Configuration time (updates at run time possible)	Run time
Impact on business transactions		Impact many business transactions (of the same type, possibly of different types)	Impact only business transactions involving the specific master data instance
Frequency of entry updates	**Insert (New)**	Seldom	Regularly
	Change	Seldom	Occasionally
	Delete	Very Seldom	Seldom
Authorized for changes		Power user	Normal user
Scope of data		More general	More specific

Table 1: Characteristics of customizing data and master data

Customizing settings influence behavior and structure of a business component - that is they change its external view. Therefore customizing options and its consequences must be apparent for a component consumer and hence need to be included in the components specification. Moreover, customizing options often impact the functioning of a business component *substantially* and have to be set at configuration time. Thus it is also necessary, that the customizing options of a business component can be clearly recognized as such.

4 Extending the Memorandum to Specify Customizable Business Components

Before we can propose *how* to specify customizing options we first need to establish *what* shall be specified. There are only limited experiences for the customizing of business components [Acke2002]. There are, however, manifold experiences for the customizing of standardized off-the-shelf business software like SAP R/3. We assume that the basic customizing principles of such standardized business applications are a good starting point to analyze the customizing of business components.

Customizing standardized business software is a complex task and is discussed by many authors (for SAP R/3 cf. e.g. [KeTe1998] and [ApRi2000]). Main topics being discussed are general approach, configuration of business processes and project management. For the area of production planning and control (PPC) several investigations about management of parameters are available, e.g. [DiMH1999]. The PPC parameters of standardized business applications were determined and classified. Based on these findings tools for configuration support were developed.

The coupling between business process models and business application systems were extensively investigated by the WEGA project, cf. [FSH+1998]. Based on the results today's approach of customizing of SAP R/3 was developed, cf. [SAP1997]. Main focus of the approach was to reduce the complexity in the large. Not in the focus was, however, the question how to describe (specify) single customizing options. The so called customizing activities of SAP R/3 are documented only at a glance, but are not formally specified. Constraints and dependencies can often only be found by trial-and-error in the system itself.

In [Acke2002] we analyzed a subset of the customizing of SAP R/3. We determined what aspects of customizing need to be considered in a specification and we discussed if these findings will also be valid for business components. The results of this investigation serve as starting point for our specification proposals.

Before developing specification proposals we formulate a general goal: Specification of customizing options shall be included in the frame provided by the memorandum. The memorandums basic structure and specification techniques shall be preserved and extensions should be limited.

The specification of customizing options can be subdivided into two parts:

- specification of customizing data; that is specification of a) the customizing fields provided by a component including possible values, b) the means to manipulate the customizing fields and c) given restrictions,
- specification of the impact of customizing data; that is specification of consequences that customizing settings have on the structure and behavior of the component.

These two parts will be discussed in sections 4.1 and 4.2, respectively. In section 4.3 we apply our specification proposals to a simple example.

4.1 Specification of customizing data

To describe the behavior of business services (that is operations of a business component) it is often necessary to refer to the state of the component. A mechanism to do so is given by Interface Information Models (IIM), cf. [ChDa2001] or [Andr2003]. Specification of behavior according to our memorandum also uses this technique.

We want to specify customizing aspects without changing the basic structure of the memorandum. To do so we use the idea of IIMs also for customizing. This means we must define a transformation from the customizing facts to be specified (cf. [Acke2002]) to an appropriate interface information model.

1. Adaptation by customizing works by setting values for predefined parameters (customizing data). The structure of the customizing data and its static dependencies must be described in a specification. To do so we make the following specification proposals:

- As a general rule, customizing data is represented by types in a UML class diagram that augments the behavioral level of the specification. This diagram acts as Interface Information Model.
- Customizing fields can be combined to logical units that will typically be maintained together. These units we call *customizing groups*. Each customizing group is represented by one type in the UML class diagram. The name of the type is given by the name of the corresponding customizing group.

- At one time customizing groups can have several instances. By structure they obey to the customizing group but can differ in their values for the customizing fields. These instances are naturally represented by the instances of the corresponding type.
- There are two possibilities for the value range of customizing fields:
  - The value range is fixed and not dependent from other customizing settings. Such a field is represented by an attribute of the respective type. The attribute will be furnished with the tagged value {C} and so labeled as relevant for customizing. The value range of the customizing field is expressed by the data type of the attribute. Additional restrictions to the value range can be expressed by an OCL condition.
  - The value range is customizing dependent, that is the value range of a field consists of all instances defined for some other customizing group. Such a field is represented by a binary association. Involved in the association are the type to which the customizing field logically belongs (association begin) and the type which defines the value range of the field (association end). The association is furnished with the tagged value {C} at the association end. A rolename can be used at the association end to represent the name of the customizing field. Otherwise the type name at the association end equals the name of the customizing field.
- There might be dependencies between customizing fields, that is, possible values for a field depend on the values of other customizing fields. Such dependencies are described by invariants in OCL. They will be part of the behavioral level and logically augment the class diagram.

2. There must exist means to manipulate the customizing data. We model them conceptually as services and call them *customizing services*. The properties of such customizing services must be specified. To do so we make the following specification proposals:

- Customizing services are represented as operations. All the operations manipulating one customizing group should be collected in one interface. Typically such an interface will include operations to create, change and delete instances of the customizing group.
- Customizing groups and their services will be listed at the task level. Their purpose and their effect on the business tasks are explained.
- The interfaces are shown in an UML diagram representing the component and its interfaces. The customizing services are shown as operations. These operations will be furnished with the tagged value {C} and so labeled as relevant for customizing.
- In accordance with our memorandum, the signature of customizing services are described on the interface level using the OMG IDL.
- When manipulating a customizing field there might be conditions, e.g. restriction of its allowed values or mandatory maintenance of this field. Such conditions are expressed as OCL pre and post conditions on the behavioral level.
- There might also be restrictions regarding the order in which customizing services can be performed. Such restrictions are specified at the coordination level. Again we use here the OCL supplemented with temporal operators.

To summarize, we identified specification relevant aspects of customizing data and proposed a way to specify them. To do so we did not need to define new notations but used the ones already suggested in the memorandum. The basic structure and the notation mix of the memorandum were so preserved and only some enhancements were necessary: mandatory use of a UML diagram to represent the customizing data, use of the tagged value {C} to identify customizing relevant properties and concrete rules how to use UML (to represent customizing services and different types of customizing fields) in a standardized way.

4.2 Specification of customizing impact

The purpose of customizing is to influence structure and behavior of a business component by assigning values to predefined data fields. Correspondingly the setting of values to customizing data can impact the following aspects of the business component:

- the structure of its business entities and the relation between different entity types,
- the behavior of the business services, the required input parameters and the result of the service,
- the order in which different business services can be performed.

This means the pre- and postconditions of the business services might change depending on the customizing data. The impact of customizing settings is therefore concentrated on the behavioral and the coordination level of our memorandum. Customizing dependent changes on other levels (especially interface and quality level) seem possible, but will not be considered here.

In order to specify these issues, we need to consider the business data and interfaces together with the customizing data and interfaces. We use one integrated conceptual scheme to represent business and customizing data of the component (see fig. 2). This scheme will serve as interface information model for all the services belonging to the business component. By doing so, the above identified consequences of customizing settings can naturally be expressed in the pre- and postconditions of the business services. The specification is analogously to other constraints regarding these services.

Note that this approach seems only practical if the complexity of the customizing options is not exceedingly high. Although the approach is technically always possible, the constraints might become too complex to be understandable. How to enhance the approach for complex cases is still an open issue and direction of further research.

4.3 Specification example

As our specification example we discuss a business component *OrderProcessing* that supports business tasks from the area of production planning and control (PPC). For the sake of simplicity we only consider a very restricted set of functionality.

The component *OrderProcessing* has an interface *IOrder* that provides business related services for orders (see fig. 4). Additionally there is an interface information model (see fig. 2) where the orders managed by the components are represented by the type *Order*.

Suppose that the component *OrderProcessing* offers three customizing groups: *Plant*, *CustomerType* and *CustomerTypeProcessControl*. With the customizing group *Plant* one can define different plants (as organizational units). For the sake of simplicity we only consider two customizing fields belonging to plants: an *ID* and a *Name*. The customizing group can have an arbitrary number of instances, e.g. the plant with ID “0001” and the name “Plant Amsterdam”. The customizing group *CustomerType* serves to define different customer types whose orders shall be handled uniformly by the component *OrderProcessing*. Again we only consider two customizing fields: an *ID* and a *Name*. There can be an arbitrary number of *CustomerType* instances. Possible examples are “Business Customer” (ID = “BUS”), “Private Customer” (ID = “PRIV”) and “Anonymous Private Customer (Internet)” (ID = “ANON”).

The customizing group *CustomerTypeProcessControl* allows to define how certain processes shall be handled for customers of different customer type. To keep our example simple we consider only one control parameter: The customizing field *PaymentFirst* is of type Boolean and indicates if an order must be paid before it will be delivered. As this choice might be different for various plants, the decision can be made for each combination of plant and customer type. The customizing group *CustomerTypeProcessControl* therefore offers three customizing fields: *Plant*, *CustomerType* and *PaymentFirst*. A possible instance with the values *Plant* = "0001", *CustomerType* = "ANON" and *PaymentFirst* = true indicates, that anonymous customers buying from plant "Amsterdam" must pay their orders before they will be delivered to them. (Note that in praxis there will be more control parameters making the customizing group more useful than in our shortened example.)

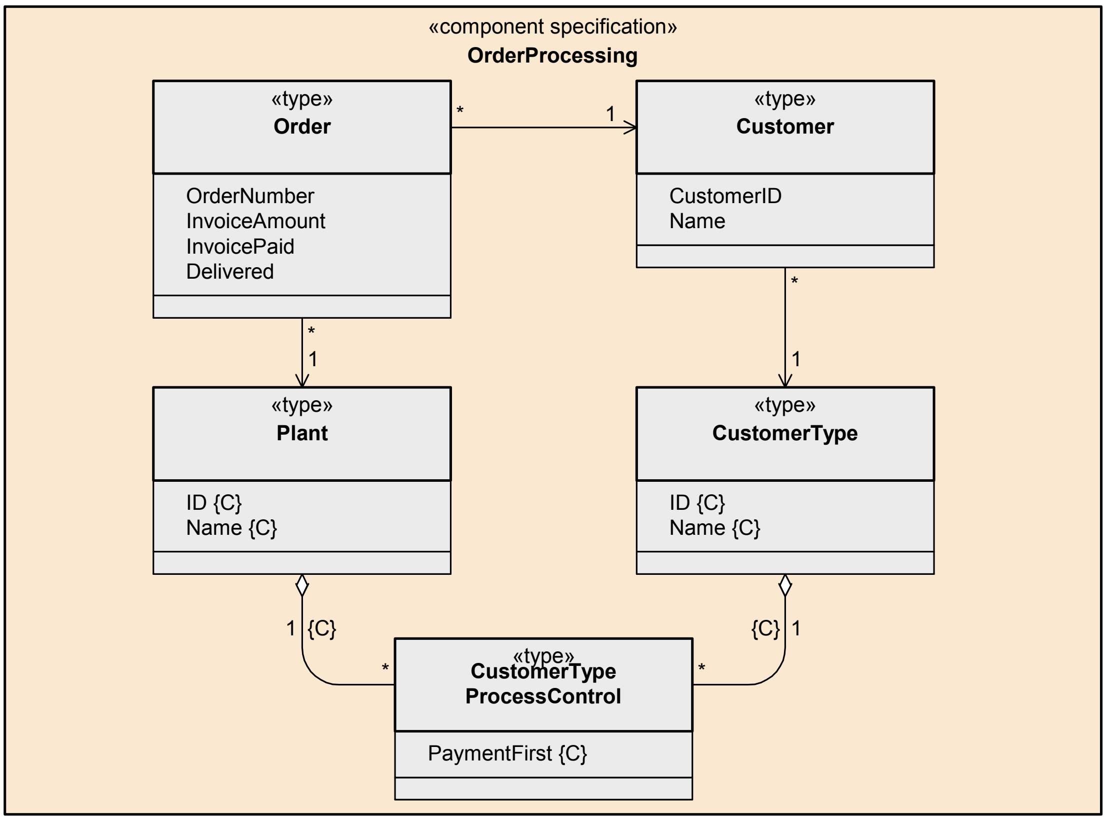

Figure 2: UML diagram acting as interface information model

The interface information model in fig. 2 shows the structure of the customizing data. The customizing groups *Plant*, *CustomerType* and *CustomerTypeProcessControl* are represented by types (having the same denominator). The customizing fields *ID* and *Name* of *Plant* are of type `string<4>` and `string<30>`, respectively. That is, they have a fixed value range. Therefore they are represented as attributes and they are tagged by {C} to indicate their relevance for customizing. The customizing fields *ID* and *Name* of *CustomerType* are represented in the same way.

The customizing group *CustomerTypeProcessControl* has one field with fixed value range: *PaymentFirst* is of type Boolean and also represented as attribute. The customizing field *CustomerType* of *CustomerTypeProcessControl*, however, does not have a fixed value range. Its value range is given by the customer types defined in the customizing group *CustomerType*. Thus this customizing field is represented by an association between the types *CustomerTypeProcessControl* and *CustomerType*. The association is tagged by {C} at the association end at *CustomerType*. The customizing field *Plant* is represented in the same way.

Conditions not shown in the diagram can be expressed by OCL on the behavioral level. The condition in figure 3, for example, declares that all plants defined in *Plant* differ in their ID.

```
OrderProcessing
  Plant->forAll(p1, p2: Plant | p1 <> p2 implies p1.ID <> p2.ID)
```

Figure 3: Example for the specification of customizing restrictions at behavioral level

Available customizing services are grouped to interfaces (according their customizing group) as shown in figure 4. There is an interface for each of the three customizing groups having the operations Create, Change and Delete. The operations are tagged by {C} to express their customizing relevance. The detailed interface specification is done on the interface level (using the OMG IDL) and will be omitted here. Restrictions regarding these services can be declared by OCL expressions.

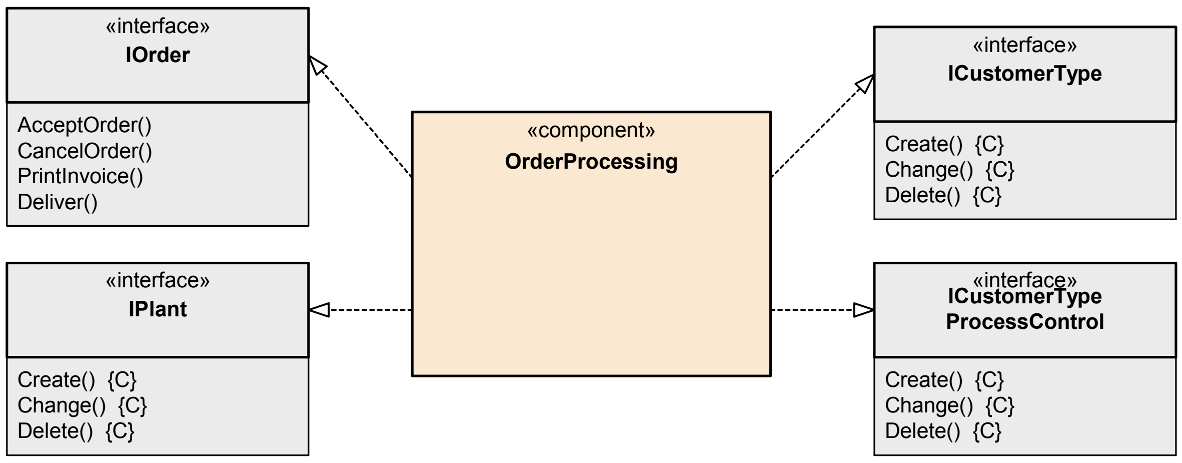

Figure 4: UML diagram for the business component *OrderProcessing* and its interfaces

By using one integrated interface information model for all interfaces we can readily specify the impact customizing settings have. In our example the business service *Deliver* can only be performed, if either the invoice is paid or the customer does not need to pay the invoice first. Figure 5 shows, how this is expressed on the behavioral level as a pre condition for the service *Deliver*.

```
OrderProcessing::Deliver(at:Order)
 pre: at.InvoicePaid = true or
   CustomerTypeProcessControl->any(pc | pc.Plant = at.Plant and
                    pc.CustomerType = at.Customer.CustomerType).PaymentFirst = false
```

Figure 5: Example for the specification of customizing impact on services

Practical experiences with the specification proposals were gained by a case study that demonstrated the feasibility of the approach, cf. [Acke2003].

5 Conclusions and Outlook

The use of business components can enable the building of business applications that combine the advantages of custom-made and standard software. To comply to user requirements, business components often need to be adapted. One prominent technique for planned adaptation is the setting of predefined parameters (called *customizing*). The availability and consequences of such parameters must be specified. As this aspect was so far not considered in the memorandum „Standardized Specification of Business Components" ([Turo2002]), we discussed how the memorandum can be enhanced accordingly.

We identified which aspects of customizing need to be specified and then showed how they can be specified. We transformed customizing data to an interface information model and enabled so to specify customizing options within the frame of the memorandum. By this approach the basic structure and the notation mix of the memorandum was preserved and only some enhancements were necessary: mandatory use of an integrated conceptual model (as UML class diagram) to represent business and customizing data, use of the tagged value {C} to identify customizing relevant properties and concrete rules how to use UML to represent customizing services and different types of customizing fields. Additionally, we discussed how the impact of customizing settings can be specified. How to do this appropriately in complex cases is still an open issue and direction of further research.

References

[Acke2001] *Ackermann, J.*: Fallstudie zur Spezifikation von Fachkomponenten. In: *K. Turowski (ed.):* 2. Workshop Modellierung und Spezifikation von Fachkomponenten. Bamberg 2001, pp. 1-66.

[Acke2002] *Ackermann, J.*: Spezifikation des Parametrisierungsspielraums von Fachkomponenten – Erste Überlegungen. In: *K. Turowski (ed.)*: 3. Workshop Modellierung und Spezifikation von Fachkomponenten. Nürnberg 2002, pp. 17 – 68.

[Acke2003] *Ackermann, J.*: Zur Spezifikation der Parameter von Fachkomponenten. In: *K. Turowski (ed.)*: 5. Workshop Komponentenorientierte betriebliche Anwendungssysteme (WKBA 5). Augsburg 2003, pp. 47 – 154.

[AlFr1998] *Allen, P.; Frost, S.*: Component-Based Development for Enterprise Systems: Applying The Select Perspective. Cambridge University Press, Cambridge 1998.

[Andr2003] *Andresen, A.:* Komponentenbasierte Softwareentwicklung mit MDA, UML und XML. Hanser Verlag, München 2003.

[ApRi2000] *Appelrath, H.-J.; Ritter, J.*: SAP R/3 Implementation: Methods and Tools. Springer, Berlin, Heidelberg 2000.

[BJP+1999] *Beugnard, A.; Jézéquel, J.-M.; Plouzeau, N.; Watkins, D.*: Making Components Contract Aware. In: IEEE Computer 32 (1999) 7, pp. 38-44.

[BRS+2000] *Bergner, K.; Rausch, A.; Sihling, M.; Vilbig, A.:* Adaptation Strategies in Componentware. In: Proceedings 2000 Australian Software Engineering Conference. IEEE Computer Society 2000, pp. 87 – 95.

[ChDa2001] *Cheesman, J.; Daniels, J.:* UML Components. Addison-Wesley, Boston 2001.

[CoTu2001] *Conrad, S.; Turowski, K.*: Temporal OCL: Meeting Specification Demands for Business Components. In: *K. Siau; T. Halpin (eds)*: Unified Modeling Language: Systems Analysis, Design and Development Issues. Idea Group, Hershey 2001, pp. 151-165.

[DiMH1999] *Dittrich, J.; Mertens, P.; Hau, M.:* Dispositionsparameter von SAP R/3-PP : Einstellungshinweise, Wirkungen, Nebenwirkungen. Vieweg, Wiesbaden 1999.

[DSWi1999] *D'Souza, D. F.; Wills, A. C.*: Objects, Components, and Frameworks with UML: The Catalysis Approach. Addison-Wesley, Reading 1999.

[FeLT2001] *Fettke, P.; Loos, P.; Tann, M. v. d.*: Eine Fallstudie zur Spezifikation von Fachkomponenten eines Informationssystems für Virtuelle Finanzdienstleister – Beschreibung und Schlussfolgerungen. In: *K. Turowski (ed.):* 2. Workshop Modellierung und Spezifikation von Fachkomponenten. Bamberg 2001, pp. 75-94.

[FSH+1998] *Ferstl, O.K.; Sinz, E.J.; Hammel, C.; Schlitt, M.; Wolf, S.; Popp, K.; Kehlenbeck, R.; Pfister, A.; Kniep, H.; Nielsen, N.; Seitz, A.:* WEGA – Wiederverwendbare und erweiterbare Geschäftsprozeß- und Anwendungssystemarchitekturen. Abschlussbericht des Verbundprojektes. Walldorf 1998.

[JaGJ1997] *Jacobson, I.; Griss, M.; Jonsson, P.:* Software Reuse. ACM Press/Addison Wesley Longman, New York 1997.

[KeTe1998] *Keller, G.; Teufel, T.:* SAP R/3 Process-oriented Implementation : Iterative Process Prototyping. Addison Wesley Longman, Harlow 1998.

[OMG2001a] *OMG (ed.):* The Common Object Request Broker: Architecture and Specification, Version 2.5, September 2001. Framingham 2001.

[OMG2001b] *OMG (ed.):* OMG Unified Modeling Language Specification, Version 1.4, September 2001. Needham 2001.

[Ortn1997] *Ortner, E.*: Methodenneutraler Fachentwurf: Zu den Grundlagen einer anwendungsorientierten Informatik. Teubner, Stuttgart 1997.

[Reus2001] *Reussner, R.*: Parametrisierte Verträge zur Protokolladaption bei Software-Komponenten. Logos Verlag, Berlin 2001.

[SAP1997] *SAP (Hrsg.*): R/3-Referenz(prozeß)modell 4.0 im R/3 Business Engineer – Zielsetzung, Inhalte, Vorgehensweise. Walldorf 1997.

[Turo1999] *Turowski, K.*: Standardisierung von Fachkomponenten: Spezifikation und Objekte der Standardisierung. In: *A. Heinzl (ed.):* 3. Meistersingertreffen. Schloss Thurnau 1999.

[Turo2001a] *Turowski, K.*: Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme. Habilitationsschrift, Otto-von-Guericke-Universität Magdeburg, Magdeburg 2001.

[Turo2001b] *Turowski, K.*: Spezifikation und Standardisierung von Fachkomponenten. In: Wirtschaftsinformatik 43 (2001b) 3, pp. 269-281.

[Turo2002] *Turowski, K. (ed.):* Standardized Specification of Business Components: Memorandum of the working group 5.10.3 Component Oriented Business Application System. Augsburg, February 2002. URL: http://www.fachkomponenten.de. Date of Call: 2003-03-07.