

Regeln, die das Business regeln

Der Business Rules Approach: Anwendung

26. Juli 2002

ÄNDERUNGSGESCHICHTE

Datum	Autor	Änderung	Kapitel
4.1.02	M. Schacher	Erste Version	(alle)
8.1.02	M. Schacher	Begriffsdefinitionen erweitert Beispiele internationalisiert	1.2 3.4, 3.6
26.7.02	M. Schacher	Kleinere Korrekturen nach Review	(einige)

INHALTSVERZEICHNIS

1.	EINFÜHRUNG	1
1.1	Zum Dokument.....	1
1.2	Begriffsdefinitionen	2
2.	EIN BEISPIEL	3
2.1	Use Cases.....	3
2.2	Business Objects.....	4
2.3	Wichtige Funktionalitäten	5
2.4	Neue Anforderungen.....	6
3.	UMSETZUNG MIT DEM BRA	7
3.1	Use Cases.....	7
3.2	Business Objects.....	8
3.3	Deklaration des Business Object Models	9
3.4	Derivations	10
3.5	Constraints	11
3.6	Action Rules	12
3.7	Business Events	13
4.	ZUSAMMENFASSUNG	14
	ANHANG A REFERENZEN	I

ABBILDUNGSVERZEICHNIS

Bild 2-1:	Die Use Cases von PuB	3
Bild 2-2:	Benutzerinteraktion zum Use Case "take order"	4
Bild 2-3:	Die Business Objects von PuB	4
Bild 2-4:	Zustandsdiagramm "Person"	5
Bild 2-5:	Zustandsdiagramm "Order"	5
Bild 3-1:	Die Use Cases von PuB (BRA)	7
Bild 3-2:	Benutzerinteraktion zum Use Case "take order" (BRA)	8
Bild 3-3:	Die Business Objects von PuB (BRA)	8

1. EINFÜHRUNG

1.1 Zum Dokument

Das vorliegende Dokument ist Teil der Serie "Regeln die das Business Regeln" zum "Business Rules Approach" (BRA). Dieser setzt sich zum Ziel ein Unternehmen und seine IT-Systeme ganzheitlich auf die Bedürfnisse des Geschäfts auszurichten. In diesem Dokument wird anhand einer ganz einfachen Fallstudie aufgezeigt, wie sich eine bestehende konventionelle Applikation hin zum Business Rules Approach transformiert lässt und welche Vorteile daraus entstehen. Die anderen Teile der Serie "Regeln die das Business Regeln" behandeln folgende Themen:

- [KG02a] beschreibt die grundlegenden Konzepte und Überlegungen, die hinter dem Business Rules Approach stehen sowie dessen Chancen und Risiken.
- [KG02b] formuliert die Konsequenzen des Business Rules Approaches auf das Business und Software Engineering und skizziert ein systematisches Vorgehen für die Abwicklung von Business Rules Projekten.
- [KG02c] zeigt verschiedene technische Architektur-Varianten für Business Rule Systeme, skizziert einen Kriterienkatalog für die Evaluation von Business Rules Produkten und gibt eine Kurzübersicht über einige gängige Produkte aus diesem Umfeld.

Zielgruppe für das vorliegende Dokument sind in erster Linie IT-Spezialisten und IT-Berater. Zum besseren Verständnis dieses Dokumentes empfiehlt sich das vorgängige Studium von [KG02a]. Ausserdem werden Grundkenntnisse der Modellierung von IT-Systemen mittels der UML (Unified Modeling Language, siehe auch [OMG01]) vorausgesetzt.

Anmerkungen zur verwendeten Terminologie

Da der Business Rules Approach aus dem amerikanischen Sprachraum stammt und sich für dieses Thema bisher noch keine einheitliche Terminologie im deutschsprachigen Raum etabliert hat, werden im folgenden konsequent die originalen englischsprachigen Begriffe verwendet. Dies betrifft insbesondere die Begriffe "Business" (statt "Geschäft"), "Business Object" (statt "Geschäftsobjekt"), "Business Event" (statt "Geschäftsereignis"), "Use Case" (statt "Anwendungsfall"), "Business Rule" (statt "Geschäftsregel") und "Business Rule Approach" (statt "Geschäftsregelansatz").

1.2 Begriffsdefinitionen

Die folgenden spezifischen Begriffe werden in diesem Dokument und werden daher an dieser Stelle definiert:

- Action Rule:** Eine \Rightarrow *Business Rule*, die aufgrund einer Manipulation an einem \Rightarrow *Business Object* oder des Auftretens eines \Rightarrow *Business Events* automatisch eine Aktion auslöst.
- Business Activity:** Eine Aufgabe, deren Ausführung einem fachlichen Ziel (\Rightarrow *Business Motivation*) dient. Ausführender ist entweder ein \Rightarrow *Business Actor* oder ein \Rightarrow *IT-System*.
- Business Actor:** Ein (menschlicher) Sachbearbeiter innerhalb einer Organisationseinheit oder eines Unternehmens mit spezifischen Aufgaben und Verantwortlichkeiten.
- Business Event:** Ein Ereignis (von ausserhalb des betrachteten Kontexts oder ein zeitliches Ereignis), welches die Ausführung einer \Rightarrow *Business Activity* anstösst.
- Business Location:** Ein geographischer Ort, an welchem \Rightarrow *Business Activities* von \Rightarrow *Business Actors* ausgeführt werden. Schliesst auch virtuelle Räume wie z.B. das Internet mit ein.
- Business Motivation:** Ein fachliches motiviertes Ziel, welches als Begründung von \Rightarrow *Business Activities* dient.
- Business Object:** Ein fachlich relevantes, physisches oder abstraktes Konzept welches zur Ausführung von \Rightarrow *Business Activities* benötigt wird bzw. durch diese erzeugt oder bearbeitet wird.
- Business Rule:** Eine prägnante Formulierung eines business-relevanten Zusammenhangs. Typischerweise wird zwischen \Rightarrow *Action Rules*, \Rightarrow *Constraints*, \Rightarrow *Guidelines* und \Rightarrow *Derivations* unterschieden.
- Business Rule Engine:** Eine technische Software-Komponente, die für die effiziente Ausführung und Überwachung von \Rightarrow *Business Rules* verantwortlich ist.
- Constraint:** Eine \Rightarrow *Business Rule*, die eine fachliche Aussage macht, die unter allen Umständen (d.h. was auch immer "gemacht" wird) wahr sein muss.
- Derivation:** Eine \Rightarrow *Business Rule*, die beschreibt, wie eine gewünschte Information aus vorhandenen Informationen abgeleitet wird. Handelt es sich bei diesen Informationen um Zahlenwerte, so reduziert sich eine Derivation oft auf eine simple Berechnung.
- Guideline:** Eine \Rightarrow *Business Rule*, die eine Empfehlung an einen menschlichen \Rightarrow *Business Actor* abgibt, jedoch selber nichts weiter bewirkt.
- IT-System:** Ein Software-System, welches \Rightarrow *Business Actors* bei der Ausführung ihrer \Rightarrow *Business Activities* unter Berücksichtigung der geltenden \Rightarrow *Business Rules* so weit wie möglich unterstützt.
- Use Case:** Ein Dienstleistungspaket, welches ein \Rightarrow *IT-System* einem \Rightarrow *Business Actor* zur Ausführung einer spezifischen \Rightarrow *Business Activity* zur Verfügung stellt.

2. EIN BEISPIEL

Bei der Fallstudie handelt es sich um eine einfache Bestellverwaltung (PuB: Produkte und Bestellungen) einer kleinen Bierhandelsfirma. Im folgenden werden einige wichtige Zusammenhänge ihrer (konventionellen) Implementation mittels Diagrammen der Unified Modeling Language (UML) [OMG01] kurz zusammengefasst.

2.1 Use Cases

Als erstes wird die Benutzerperspektive auf die Applikation "PuB" mittels eines Use Case Diagramms beschrieben.

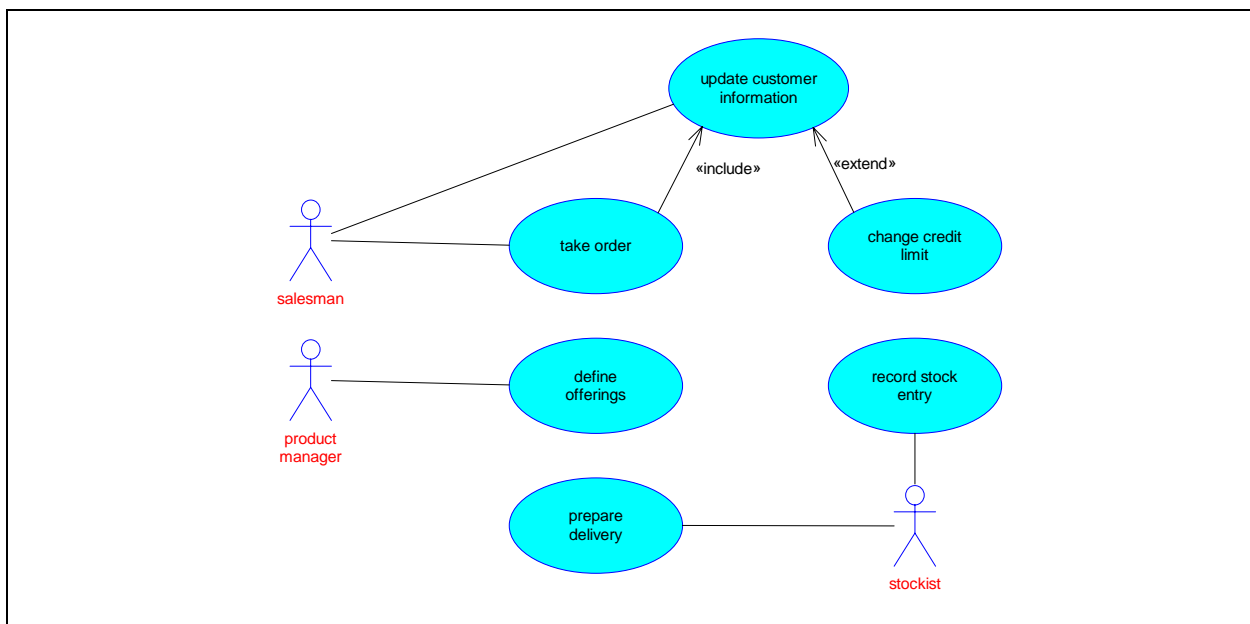


Bild 2-1: Die Use Cases von PuB

Die wichtigsten im Verkauf bereitgestellten Use Cases sind "Bestellung erfassen" ("take order") und "Angebot definieren" ("define offering"). Im Zusammenhang mit der Bestellungserfassung müssen auch Kundendaten im allgemeinen sowie die Kreditlimite im Speziellen bearbeitet werden können. Diese Use Cases werden dem Verkäufer zur Verfügung gestellt. Daneben werden von der Applikation Use Cases für die Lagerbewirtschaftung und Auslieferung zur Verfügung gestellt.

Im folgenden werden wir uns lediglich auf den Use Case ""Bestellung erfassen" ("take order") konzentrieren. Das Sequenzdiagramm Bild 2-2 zeigt die Interaktion zwischen Benutzer und System um eine neue Bestellung zu erfassen.

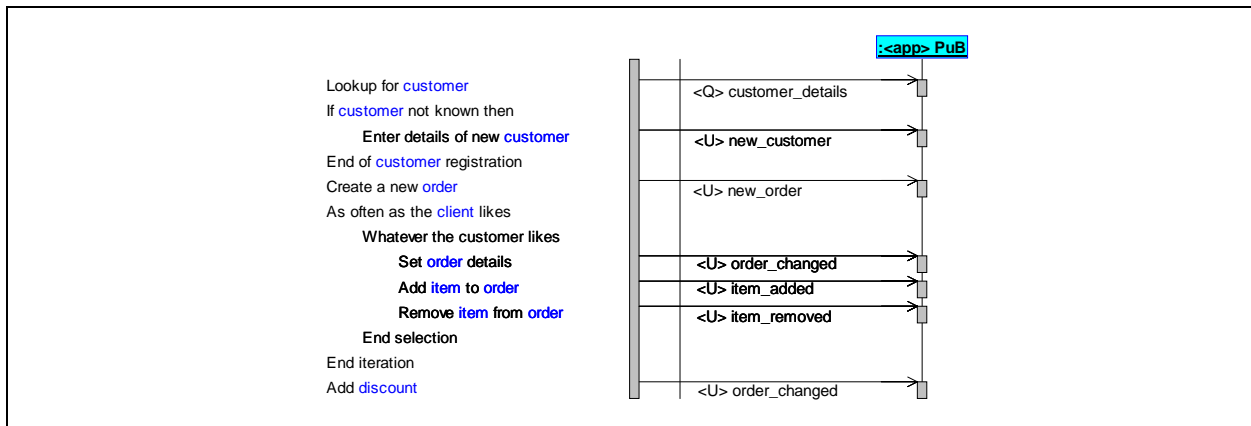


Bild 2-2: Benutzerinteraktion zum Use Case "take order"

2.2 Business Objects

Grundlage der oben beschriebenen Use Cases bilden die Business Objects (d.h. die fachlichen Objekte) von PuB, welche durch folgendes Klassendiagramm zusammengefasst sind:

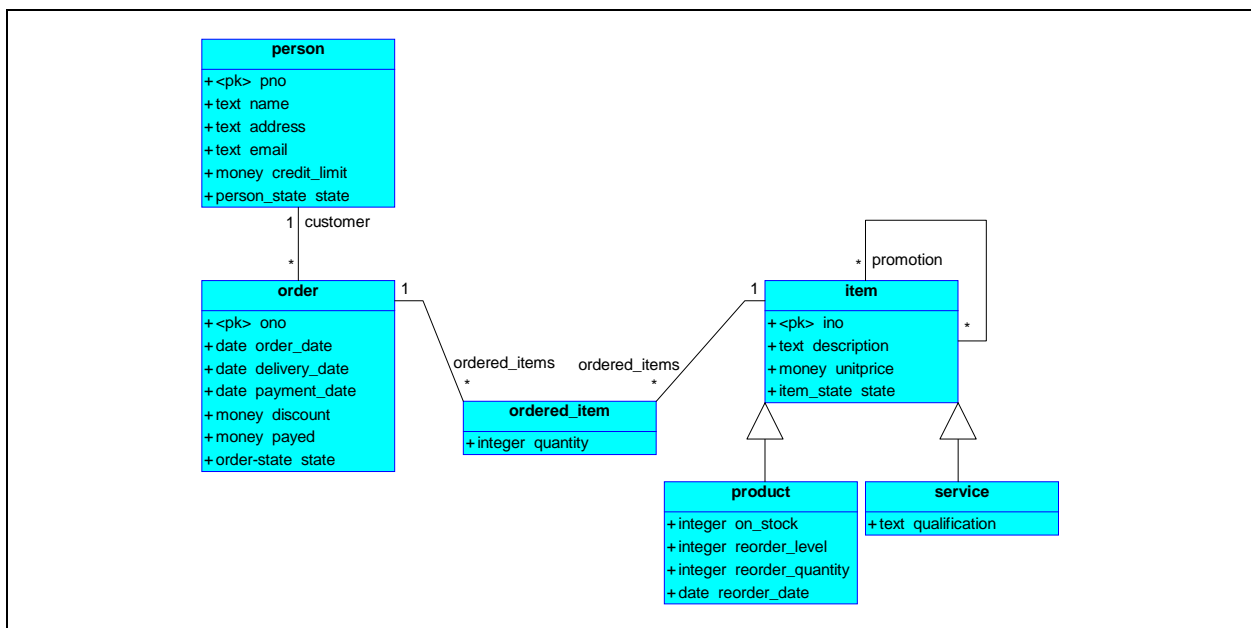


Bild 2-3: Die Business Objects von PuB

Ein "Kunde" ("customer") kann gleichzeitig mehrere "Bestellungen" ("order") aufgegeben haben, wobei immer seine jeweilige Kreditlimite ("credit limit") zu berücksichtigen ist. Eine Bestellung kann mehrere Leistungen ("item") umfassen, wobei eine Leistung jeweils entweder ein Produkt ("product") oder eine Dienstleistung ("service") sein kann.

2.3 Wichtige Funktionalitäten

Ein Kunde ist erst dann "aktiv" ("active"), wenn er mindestens einmal eine Bestellung aufgegeben hat, davor ist er lediglich ein potentieller Kunde ("prospect"). Ein aktiver Kunde ist entweder ein "normaler Kunde" ("standard") oder ein "guter Kunde" ("preferred"): ein "guter Kunde" kann von besseren Konditionen profitieren. Ob ein Kunde "normal" oder "gut" ist, wird durch den jeweiligen Sachbearbeiter festgelegt. Ausserdem kann ein Kunde auf die "schwarze Liste" ("blacklisted") gesetzt werden: in diesem Falle kann er keine neuen Bestellungen mehr aufgeben. Diese Zusammenhänge sind durch das Zustandsdiagramm in Bild 2-4 dargestellt.

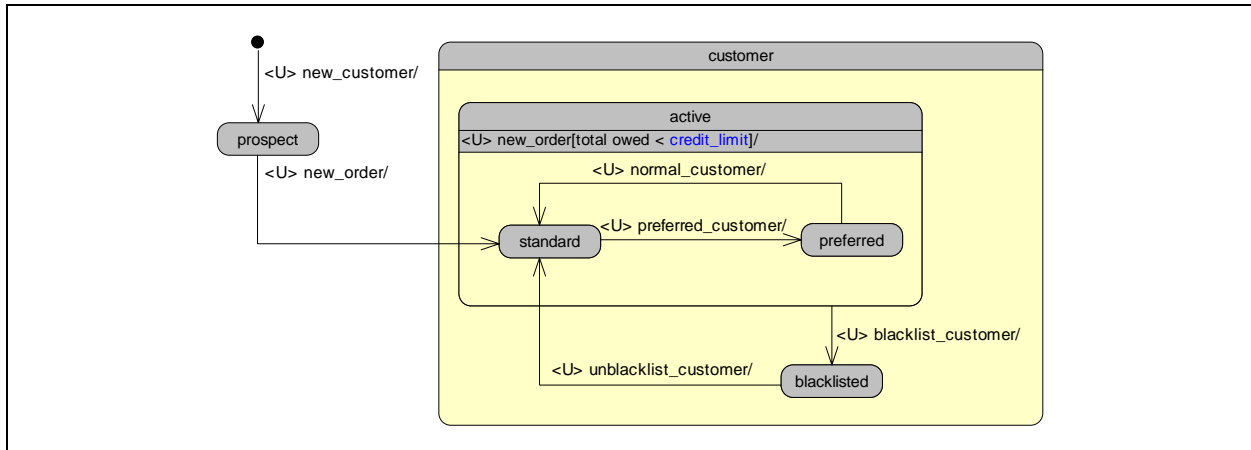


Bild 2-4: Zustandsdiagramm "Person"

Das Zustandsdiagramm Bild 2-5 zeigt einige wichtige Zusammenhänge für eine Bestellung. Durch den Sachbearbeiter wird festgelegt, ob es sich um eine kleine ("small") oder grosse ("big") Bestellung handelt. Im Falle einer grossen Bestellung ist eine Anzahlung notwendig ("initial payment") bevor die Auslieferung erfolgen kann. Eine Bestellung kann nur so lange verändert werden ("order changed"), bis sie ausgeliefert wurde bzw. bis eine Anzahlung erfolgt ist.

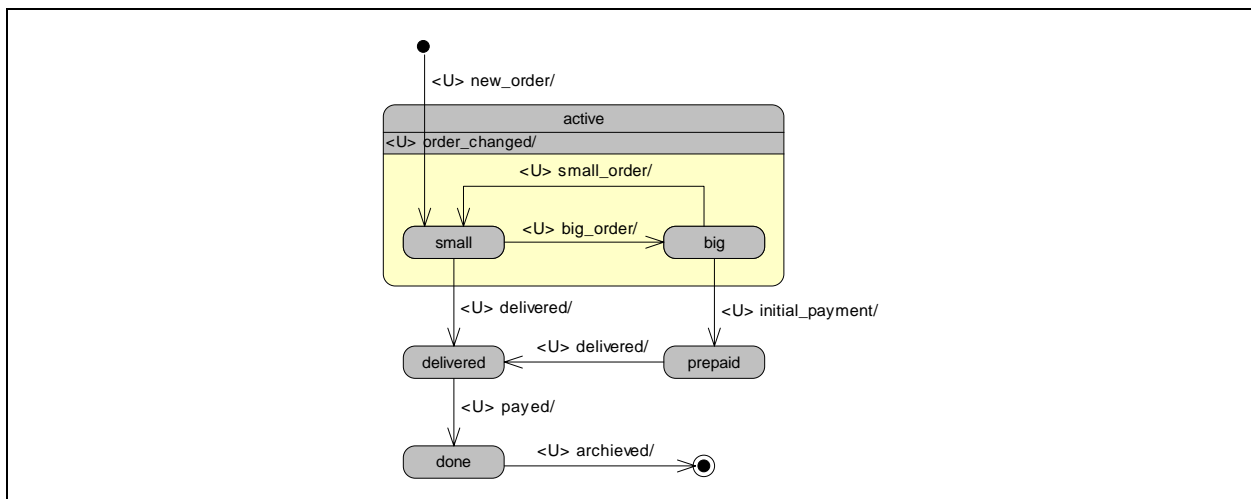


Bild 2-5: Zustandsdiagramm "Order"

2.4 Neue Anforderungen

Basierend auf der oben beschriebenen konventionellen Implementation drängen sich zur Sicherstellung des Geschäfts heute folgende Unsicherheiten und neuen Anforderungen auf:

- **Selbstbedienung über Web**

Da in Zukunft nicht nur der Schweizer Heimmarkt, sondern auch das Ausland beliefert werden soll, muss eine Web-fähige Selbstbedienungslösung angeboten werden. Nur so kann verhindert werden, dass gleichzeitig der Personalaufwand im Verkauf "explodiert".

- **Wann ist ein Kunde "gut"?**

Bisher wurde von jeweiligen zuständigen Sachbearbeiter festgelegt, ob ein Kunden auch ein "guter Kunde" ist (mit entsprechenden Sonderkonditionen). Zwar existieren Richtlinien auf Papier, wann ein Kunde zu welchen Sonderkonditionen berechtigt ist, diese wurden allerdings von verschiedenen Verkäufern sehr unterschiedlich interpretiert, falls sie überhaupt bekannt waren. In Zukunft soll daher eine unternehmensweit einheitliche Kundenklassifikation möglich sein.

- **Wann ist eine Bestellung "gross"?**

Auch ob eine Bestellung gross ist und damit eine Anzahlung notwendig wird wurde bisher durch den zuständigen Sachbearbeiter festgelegt. Hier gelten die analogen Probleme wie bei der Fragestellung "Wann ist ein Kunde gut?" und auch hier soll eine unternehmensweit einheitliche Regelung garantiert werden.

- **Flexible und einheitliche Rabattierung**

In Zukunft soll nicht nur die Kundenklassifikation vereinheitlicht werden, sondern auch das Rabattsystem. Es soll festgelegt werden können, wann welche Produkte unter welchen Bedingungen in welchen Niederlassungen wie stark rabattiert werden können/sollen. Diese Zusammenhang sollen in kürzester Zeit (Tage, wenn nicht sogar Stunden) unternehmensweit der aktuellen Marktsituation angepasst werden können (z.B. für die Lancierung neuer Produkte).

3. UMSETZUNG MIT DEM BRA

Im folgenden werden vor allem die Unterschiede aufgezeigt, die der Business Rules Approach in der Fallstudie "PuB" bewirkt. Zur Illustration der Beispiele wird die experimentelle Business Rules Plattform von KnowGravity Inc. verwendet. Sie basiert auf der Layer-Architektur, wie sie in [KG02c], aber auch in [DATE00] beschrieben ist. Andere Business Rules Produkte können andere Eigenschaften aufweisen und/oder eine andere Syntax verwenden.

3.1 Use Cases

Das folgende Use Case Diagramm zeigt die wichtigsten Einflüsse des Business Rules Approach aus der Benutzerperspektive:

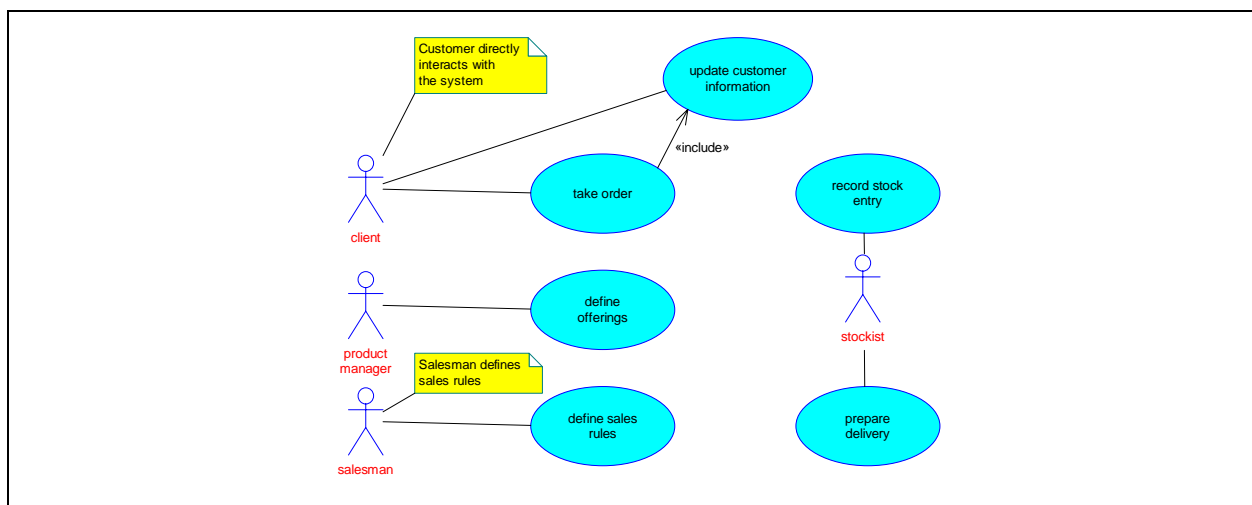


Bild 3-1: Die Use Cases von PuB (BRA)

Vielleicht die wesentlichste Änderung ist, dass der Verkäufer praktisch entfällt: wir haben nun einen "Selbstbedienungsladen", indem der Kunde (z.B. via Internet) direkt mit der als Verkäufer agierenden Applikation interagiert (Use Case "take order"). Dazu muss die Applikation allerdings wissen, "wie man verkauft" bzw. welche Regeln dabei zu beachten sind. Glücklicherweise gibt's da doch noch etwas für einen Verkäufer zu tun: er legt die aktuellen Regeln fest, wie z.B. Produkt-Promotionen und andere Marketing-Aktionen oder die Vergabe von Kreditlimiten an verschiedene Kundentypen.

Das Sequenzdiagramm in Bild 3-2 für den Use Case "take order" zeigt nun deutlich, wie die neue Applikation nicht nur passiv auf Befehle des Benutzers (des Kunden) wartet, sondern bei Gelegenheit auch einmal selber die Initiative ergreift. Insbesondere werden folgende Aktionen und Entscheidungen von der Applikation und nicht mehr durch den Verkäufer getätigt:

- Sobald ein interessierter Kunde beginnt Informationen über sich zu erfassen, wird er (falls er bereits bekannt ist) von der Applikation begrüßt und seine Daten werden automatisch vervollständigt.
- Je nach bestellten Produkten werden dem Kunden automatisch komplementäre Produkte oder Sonderaktionen vorgeschlagen.
- Bei Abschluss der Bestellung entscheidet die Applikation aufgrund der History des Kunden über die Höhe eines allfälligen Zusatzrabattes und präsentiert diesen dem Kunden als "nette Zugabe".

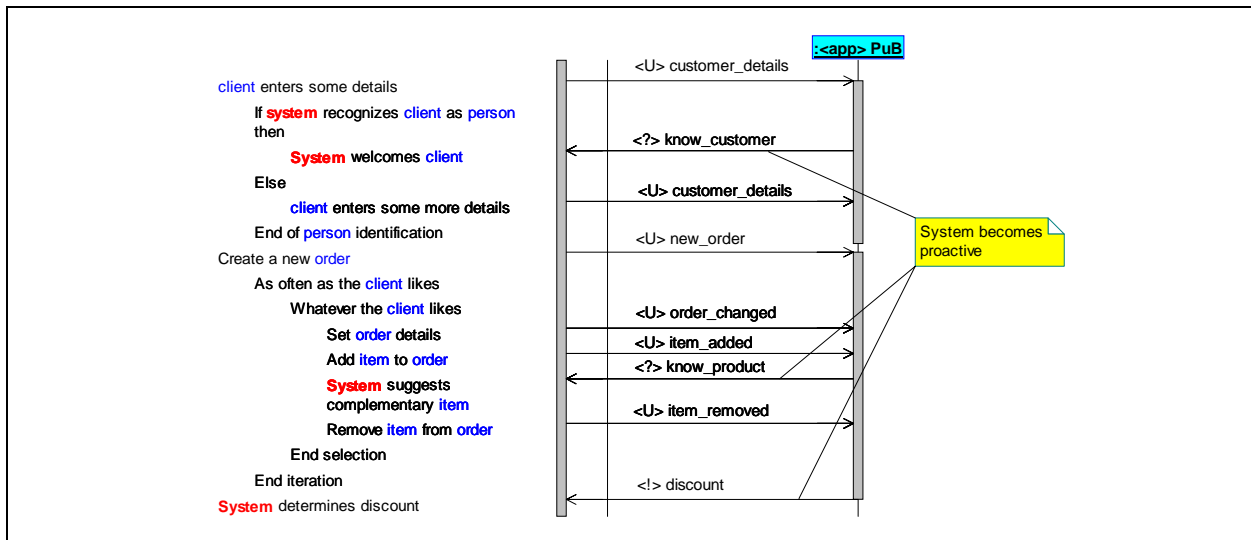


Bild 3-2: Benutzerinteraktion zum Use Case "take order" (BRA)

3.2 Business Objects

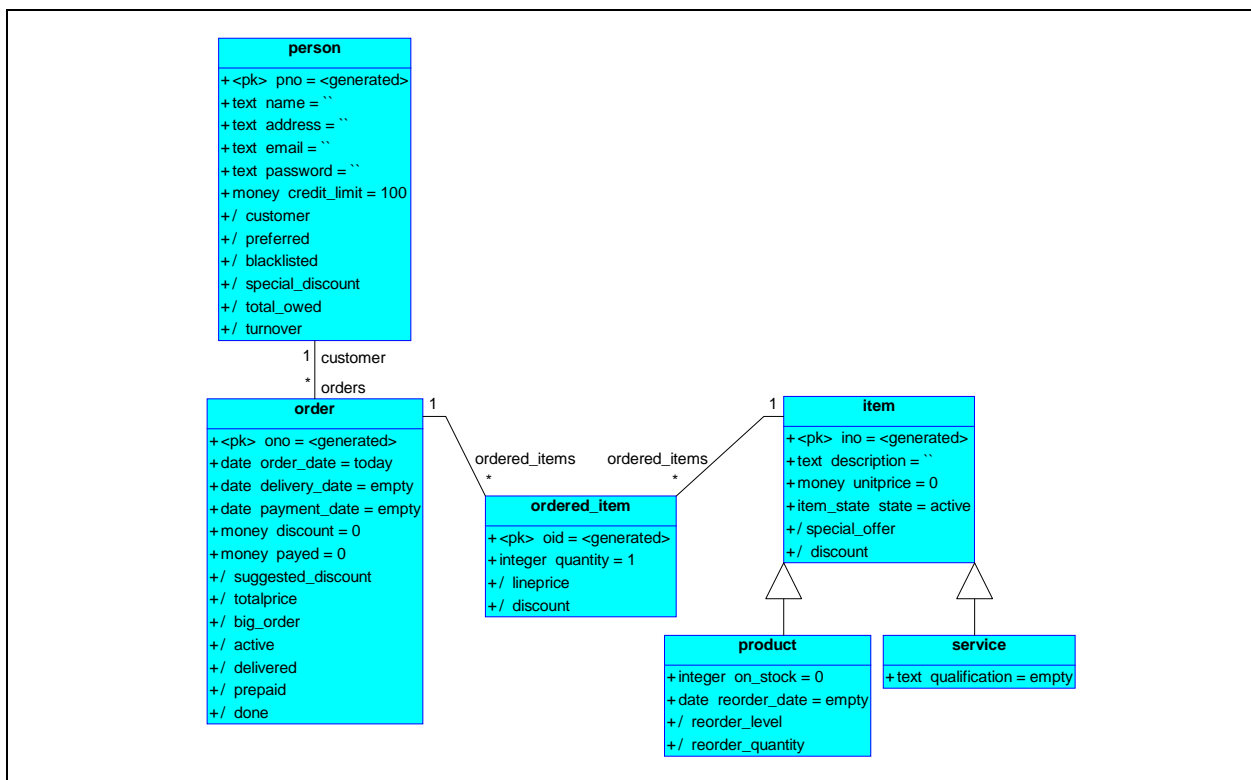


Bild 3-3: Die Business Objects von Pub (BRA)

Am offensichtlichsten im Vergleich zur konventionellen Implementation fällt in der BRA-Implementation die Menge von abgeleiteten Attributen (in der UML durch einen dem Attributnamen vorangestellten Schrägstrich gekennzeichnet) auf. Diese Attributwerte (bzw. diese Informationen) werden nun durch Business Rules automatisch bestimmt und müssen nicht mehr durch den Benutzer festgelegt werden.

Ausserdem sind für alle anderen Attribute Initialisierungswerte angegeben und gegenüber der konventionellen Implementation ist bei der "person" das Attribut "password" hinzugekommen. Dieses wird zusammen mit der "email"-Adresse benötigt, damit sich ein Kunde via Internet eindeutig identifizieren kann.

Bevor dies allerdings funktioniert, muss dem Business Rules Engine das Business Object Model "bekannt" gemacht werden.

3.3 Deklaration des Business Object Models

Das Business Object Model wird üblicherweise mittels einer konventionellen, relationalen Datenbank persistent gemacht. Dazu müssen die konzeptionellen Business Objects wie üblich in physische Tabellen abgebildet werden, wobei gegebenenfalls auch komplexere Transformationen notwendig werden können, etwa zur Auflösung von M:M Beziehungen oder Generalisierungen/Spezialisierungen.

Bevor nun Business Rules einem Business Rules Engine "beigebracht" werden können, muss diesem erst das Business Object Model bekannt gemacht, d.h. deklariert werden. Die folgenden Beispiele zeigen, wie die sechs Business Objects unseres Fallbeispiels deklariert werden¹:

```
declare_object(person, [], tbl_person, pno,
  [name is ``, address is ``, credit_limit is 100]).
declare_object(order, [], tbl_order, ono,
  [pno is 0, order_date is today, delivery_date is empty, payment_date is empty,
  discount is 0, payed is 0]).
declare_object(ordered_item, [], tbl_ordered_item, oid,
  [ono is 0, ino is 0, quantity is 1]).
declare_object(item, [], tbl_item, ino,
  [description is ``, unitprice is 0, state is `active`]).
declare_object(product, [item], tbl_product, ino,
  [on_stock is 0, reorder_date is empty]).
declare_object(service, [item], tbl_item, ino,
  [qualification is ``]).
```

Wie obige Beispiele zeigen, muss pro Business Object dessen Name, dessen allfällige Generalisierungen (z.B. "item" für Business Object "product"), der Name der Datenbank-Tabelle mit Angabe des Primärschlüssels sowie die Initialisierungswerte der Attribute definiert werden.

Neben den Objekten müssen auch die Beziehungen zwischen den Objekten dem Business Rules Engine deklariert werden. Die folgenden drei Deklarationen definieren jeweils die an der Beziehung beteiligten Objekte (z.B. "person" und "order"), deren Rollennamen (z.B. "customer" und "orders") sowie das Fremdschlüssel-Attribut (z.B. "pno"):

```
declare_assoc(person, customer, orders, order, pno).
declare_assoc(order, order, ordered_items, ordered_item, ono).
declare_assoc(item, item, ordered_items, ordered_item, ino).
```

Insbesondere die Definition der Rollennamen ist hier wichtig, damit diese in den Navigations-Ausdrücken der Business Rules (siehe folgende Kapitel) verwendet werden können. In relationalen Datenbanksystemen werden solche Rollennamen üblicherweise nicht bzw. nicht vollständig unterstützt und müssen daher hier explizit eingeführt werden.

¹ Die hier verwendete Syntax ist diejenige der experimentellen Business Rules Plattform von KnowGravity Inc. Andere Business Rules Produkte können eine andere Syntax verwenden oder bieten eine Import-Schnittstelle für Datenbanken an.

3.4 Derivations

Derivations sind sogenannte "State Inference Rules", d.h. Business Rules, die Informationen über ein Objekt (d.h. Teile seines Zustands) herleiten. Besonders deutlich wird dies für die Attribute "/customer", "/preferred", "/active" und "/blacklisted" der Business Objects "customer" sowie die Attribute "/active", "/big_order", "/delivered", "/prepaid" und "done" von "order": Sie machen die Zustandsdiagramme der konventionellen Lösung einerseits überflüssig und können andererseits flexibel neuen Bedürfnissen angepasst werden.

Die folgende Regel¹ definiert beispielsweise, wann eine Person als ein Kunde betrachtet wird: sobald mindestens eine Bestellung existiert (das führende "d1" dient zur deren eindeutigen Benennung).

```
d1 :: derive person.customer is true if person.orders=>exists
      else person.customer is false.
```

Etwas komplizierter ist die Entscheidung, wann ein Kunde ein bevorzugter ("preferred") Kunde ist:

```
d2 :: derive person.preferred is true if person.turnover > 200 and not person.blacklisted
      else person.preferred is false.
```

Die obige Regel basiert auf Attributen einer Person, die ihrerseits abgeleitet sind, d.h. von weiteren Business Rules "bestimmt" werden (siehe unten).

Die folgende Regel zeigt schliesslich die Entscheidung, wann ein Kunde auf der schwarzen Liste steht: nämlich immer dann, wenn mindestens eine seiner Bestellungen vor mehr als 30 Tagen ausgeliefert wurde, jedoch immer noch keine Zahlung eingegangen ist.

```
d3 :: derive person.blacklisted is true
      if person.orders=>exists(order.delivery_date + 30days < now
                               and order.payment_date = empty)
      else person.blacklisted is false.
```

Was nun noch fehlt ist die Berechnung des Umsatzes einer Person, welche durch die folgenden bedingungslosen Regeln beschrieben wird:

```
d4 :: derive person.turnover is person.orders.totalprice=>sum.
d5 :: derive order.totalprice is order.ordered_items.lineprice=>sum - order.discount.
d6 :: derive ordered_item.lineprice is ordered_item.quantity * ordered_item.item.unitprice *
      (100 - ordered_item.discount)/100.
```

Diese schliesslich basieren wiederum auf Regeln, welche die verschiedenen Rabattsätze bestimmen:

```
d7 :: derive ordered_item.discount is 5 + ordered_item.item.discount
      if ordered_item.quantity > 9 and ordered_item.item.description = `Heineken`
      else ordered_item.discount is 0 + ordered_item.item.discount.
d8 :: derive item.discount is 7 if item.description = `San Miguel`
      else item.discount is 0.
```

Da diese Regeln deklarativ formuliert sind und durch eine Business Rules Engine ausgeführt werden, ergeben sich im wesentlichen folgende Vorteile:

¹ Die hier verwendete Syntax ist diejenige der experimentellen Business Rules Plattform von KnowGravity Inc. und orientiert sich an der Object Constraint Language der UML [WK99]. Die Syntax anderer Business Rules Produkte kann von der hier verwendeten mehr oder weniger stark abweichen.

- Ohne Änderungen von Datenbank und/oder Applikation können diese Regeln verändert werden, um z.B. eine Marketingaktion zu lancieren.
- Es können jederzeit neue Regeln hinzugefügt oder bestehende Regeln entfernt werden.
- Die Regeln können in beliebiger Reihenfolge formuliert werden. In welcher Reihenfolge sie ausgeführt werden, bestimmt die Business Rules Engine automatisch.

Aus obigen Regelbeispielen wird ersichtlich, dass gewisse Regeln statischer sind (z.B. zur Bestimmung des Umsatzes) als andere (die verschiedenen Rabattregeln). Aus diesem Grund können Regeln in BRA-Produkten typischerweise nicht von jedermann definiert bzw. verändert werden, sondern nur von denjenigen Benutzern, die auch dazu berechtigt sind.

3.5 Constraints

Als nächstes werden Regeln formuliert, die Einschränkungen oder Rahmenbedingungen zur Ausführung des Business' darstellen. Als erstes Beispiel eine Regel, die garantiert, dass die Summe der Preise der offenen Bestellungen einer Person ihre Kreditlimite nicht überschreiten darf:

```
c1 :: ensure person.total_owed =< person.credit_limit.
```

Dabei ist "total_owed" ein abgeleitetes Attribut, welches durch eine Derivation bestimmt wird und "credit_limit" ein Attribut, welches (zumindest in der momentanen Lösung) durch einen menschlichen Sachbearbeiter festgelegt wird.

Als weiteres Beispiel begrenzt die folgende Regel die Anzahl der gleichzeitig aktiven, d.h. gültigen Angebote auf maximal 20:

```
c2 :: ensure item.allinstances=>select(state = `active`)=>size =< 20.
```

Demgegenüber werden durch die folgenden Regeln verwaiste Bestellzeilen verhindert:

```
c3 :: ensure ordered_item.order=>exists.  
c4 :: ensure ordered_item.item=>exists.
```

Diese beiden Regeln entsprechen eigentlich weitgehend der referentiellen Integrität, wie sie heute von den meisten Datenbank Management Systemen unterstützt wird. Hingegen zeigt die folgende Regel ein wesentlich komplexeres Beispiel einer referentiellen Integrität, die sich nur noch sehr schwer in einer herkömmlichen Datenbank formulieren lässt:

```
c5 :: ensure ordered_item=>select(order.active)=>forall(item.state = `active`).
```

Sie besagt, dass in jeder Bestellzeile einer "aktiven" (eine abgeleitete Information) Bestellung eine Leistung existieren muss, welche im Zustand "aktiv" ist. Durch diese Regel wird einerseits verhindert, dass nicht aktive Leistungen bestellt werden, aber auch, dass eine Leistung aus dem Verkehr gezogen werden kann (state = "phased_out"), wenn noch aktive Bestellungen für diese Leistung existieren.

Da diese Regeln wiederum deklarativ formuliert sind und durch eine Business Rules Engine überprüft werden, ergeben sich die folgende Vorteile:

- Bei der (programmtechnischen) Manipulation der Business Objects müssen diese Regeln nicht berücksichtigt werden; die Business Rules Engine „wehrt sich schon“, wenn daraus eine Verletzung einer dieser Regeln resultieren würde.
- Es können jederzeit neue Regeln hinzugefügt oder bestehende Regeln verändert bzw. entfernt werden.
- Die Regeln können in beliebiger Reihenfolge formuliert werden. Wann und in welcher Reihenfolge sie überprüft werden, bestimmt die Business Rules Engine automatisch.

3.6 Action Rules

Schliesslich werden Regeln formuliert, die Aktivitäten auslösen oder auch verhindern wenn gewisse Ereignisse eintreten. Beispielsweise sorgt die folgende Regel dafür, dass neue Kunden aus der Region "Paris" speziell begrüsst werden:

```
a1 :: when person is created if person.address = `Paris` then say `Hello in Paris!`.
```

Die folgenden drei Regeln sorgen dafür, dass wenn eine Person gelöscht wird, auch sämtliche ihre Bestellungen, inklusive ihrer Bestellzeilen gelöscht werden. Dies ist selbstverständlich nur dann erlaubt, wenn keine der Bestellungen noch nicht erledigt ist (Aktion "reject" in Regel a3):

```
a2 :: when person is deleted do delete person.orders.  
a3 :: when person is deleted if orders=>exists(done=false) then reject.  
a4 :: when order is deleted do delete order.ordered_items.
```

In ähnlicher Weise verhindert die folgende Regel, dass eine Bestellung verändert werden kann, die zwar ausgeliefert wurde aber noch nicht bezahlt worden ist. Dabei wurde auch gleich die Meldung spezifiziert, die im Falle einer Zuwiderhandlung dem Benutzer angezeigt werden soll:

```
a5 :: when ordered_item is created or ordered_item is changed or ordered_item is deleted  
      if order.delivered and not order.done  
      then reject(`Order cannot be changed since it is delivered but not yet payed!`).
```

Durch die folgende Regel ist gewährleistet, dass eine neu kreierte Bestellung auch immer gleich mindestens ein Produkt enthält und dass bei der Bestellung eines Produkts automatisch ein anderes empfohlen wird (zwei gemeine, aber durchaus übliche Marketing-Tricks):

```
a6 :: when order is created  
      do create order.ordered_items with [quantity is 1, ino is 1].  
  
a7 :: when ordered_item is created or ordered_item is changed  
      if ordered_item.item.description = `Heineken` then say `Hey, I have a hot tip!` and  
                                             say `Do you also know Guinness?`.
```

Schliesslich sorgt die folgende Regel dafür, dass bei der Bezahlung einer Bestellung auch gleich das aktuelle Datum als Zahlungsdatum festgehalten wird:

```
a8 :: when order.payed is changed do change order.payment_date is today.
```

Auch diese Regeln sind wieder deklarativ formuliert, woraus sich folgende Vorteile ergeben:

- Bei der (programmtechnischen) Manipulation der Business Objects müssen diese Regeln nicht berücksichtigt werden; die Business Rules Engine sorgt dafür, dass automatisch alle notwendigen Aktionen ausgeführt werden.
- Es können jederzeit neue Regeln hinzugefügt oder bestehende Regeln verändert bzw. entfernt werden.
- Die Regeln können in beliebiger Reihenfolge formuliert werden. Wann und in welcher Reihenfolge sie ausgeführt werden, bestimmt die Business Rules Engine automatisch.

3.7 Business Events

Die Action Rules im vorherigen Kapitel definieren, was passieren soll, wenn ein Business Object kreiert, verändert oder gelöscht wird. Dies sind Ereignisse, welche sich auf einem relativ feingranularen, technischem Niveau befinden. Demgegenüber sind Business Events Ereignisse, die eine fachliche Bedeutung haben und üblicherweise Operationen auf mehreren Objekten bewirken.

Mittels einer spezielle Variante von Action Rules lassen sich Reaktionen auf Business Events definieren. Tritt ein solcher Business Event auf, so werden alle dadurch ausgeführten Aktionen ganz oder gar nicht ausgeführt: sie bilden somit eine logische Transaktion. Innerhalb einer solchen logischen Transaktion kommen sämtliche bisher besprochenen Business Rules zum Zug:

- Bei lesenden Objekt-Operationen kommen ggf. Derivations zur Anwendung.
- Bei schreibenden Objekt-Operationen werden ggf. Action Rules angestossen.
- Beim Abschluss der Transaktion werden sämtliche Constraints überprüft und ggf. ein Rollback ausgelöst.

Das folgende Beispiel zeigt die Definition des Business Events "order delivered" in Form einer Action Rule:

```
e01 :: when order_delivered occurs with [ono is ONO]
      do order.select(ono = ONO).delivery_date is today and
         order.select(ono = ONO).ordered_items=>
            forall(item.type = `product` implies item.on_stock is item.on_stock - quantity).
```

Wird der Business Event nun z.B. mittels

```
do_event(order_delivered with [ono is 3]).
```

dem IT-System mitgeteilt, so wird das Lieferdatum der Bestellung "3" auf "heute" gesetzt und bei sämtlichen bestellten Produkten (und nur bei diesen) der Lagerbestand entsprechend korrigiert. Wird dabei auch nur eine Constraint verletzt (z.B. Lagerbestand < 0), so werden alle bisherigen Aktionen rückgängig gemacht, da ein Business Event wie bereits gesagt, als Transaktion betrachtet wird.

Ein anderes Beispiel stellt der Business Event "customer enters shop" dar. Dabei gibt ein Kunde via Internet seine EMail-Adresse ein. Falls er bereits bekannt ist, soll er begrüßt, andernfalls sollen seine weiteren Details abgefragt werden. Folgende Action Rule erledigt dies für uns:

```
e02 :: when customer_enters_shop occurs with [email is EMail]
      if customer=>exists(email = EMail) then
        if ask password_form = customer.password then
          say `This is not you (or your password is incorrect)!`
        else
          say `Hello my friend!`
      else
        show customer_details_form [email is EMail].
```

Mittels den Aktionen "say", "ask" und "show" werden jeweils Elemente des User Interfaces aktiviert, welche in ähnlicher Weise wie das Business Object Model dem Business Rules Engine bekannt gemacht werden müssen (hier nicht gezeigt). Ein anderes Beispiel einer solcher User Interfaces Aktion könnte der Versand einer EMail sein, wenn ein bestimmtes Ereignis eingetroffen ist.

4. ZUSAMMENFASSUNG

Im vorliegenden Dokument wurde aufgezeigt, wie eine nach dem Business Rules Approach entwickelte Applikation konkret aussehen könnte. Business Rules sind fachliche Zusammenhänge, die einem Business Rules Engine "beigebracht" werden, damit dieses innerhalb eines IT-Systems automatisch diese Regeln befolgt bzw. überwacht.

Business Rules beziehen sich immer auf Business Objects, den für das Business fundamentalen Konzepten. Grundsätzlich kann daher zwischen folgenden Typen von Business Rules unterschieden werden:

- **Derivations** verleihen den Business Objects **Wissen**
- **Constraints** bringen den Business Objects **Selbstverteidigung** bei
- **Action Rules** verleihen Business Objects **Eigeninitiative**

Der Business Rules Approach verleiht den Business Objects eine sehr grosse Selbständigkeit, da durch die Business Rules die gesamte fachliche Logik eines IT-Systems formuliert werden kann. Eine entsprechend leistungsfähige Business Rule Engine vorausgesetzt, reduziert sich die Applikationsentwicklung nach dem Business Rules Approach auf folgende Gleichung:

$$\text{Application} = \text{Business Objects} + \text{Business Rules (+ User Interface)}$$

Mit anderen Worten: um eine Applikation zu bauen, müssen lediglich die Business Objects definiert (und in einer gängigen Datenbank implementiert werden), die Business Rules formuliert (und in eine leistungsfähige Business Rules Engine geladen werden) und schliesslich noch (in mehr oder weniger konventioneller Weise) ein User Interface implementiert werden.

Sobald sich eine solche Applikation im operationellen Betrieb befindet, lassen sich typischerweise die Business Rules "im Flug", d.h. ohne weitere Involvierung von IT-Spezialisten direkt durch die verantwortlichen (fachlichen) Sachbearbeiter den sich ständig ändernden Bedürfnissen des Geschäfts anpassen.

Selbst in der einfachen Fallstudie dieses Dokumentes werden folgende fachlichen Themen durch Business Rules abgedeckt:

- automatische Preisgestaltung
- automatische Markt-Segmentierung von Kunden
- automatisches Marketing und Promotionen
- automatische Produktbündelungen und -konfiguration.

Dies sind neben weiteren auch die als "typische" bekannten Anwendungsgebiete des Business Rules Approaches. Herausragende Merkmale aller dieser Beispiele sind der hohe Automatisierungsgrad sowie die flexible und sehr rasche Anpassbarkeit an die aktuellen Bedürfnisse des Business.

ANHANG A REFERENZEN

- [DATE00] C. J. Date: *WHAT Not HOW*, Addison-Wesley, 2000
- [KG02a] KnowGravity Inc.: *Rules that Rule the Business - The Business Rules Approach: Concepts*, 2002, verfügbar unter www.knowgravity.com
- [KG02b] KnowGravity Inc.: *Rules that Rule the Business - The Business Rules Approach: Methodology*, 2002, verfügbar unter www.knowgravity.com
- [KG02c] KnowGravity Inc.: *Rules that Rule the Business - The Business Rules Approach: Technology*, 2002, verfügbar unter www.knowgravity.com
- [OMG01] Object Management Group: *OMG Unified Modeling Language - Version 1.4*, OMG, September 2001
- [WK99] Jos Warmer, Anneke Kleppe: *The Object Constraint Language - Precise Modeling with UML*, Addison Wesley, 1999