Überführung einer Spezifikation in EXPRESS in die Entwicklungsumgebung KAPPA

Ort, A.

Dieser Artikel beschreibt eine automatische Umsetzung einer formalen Spezifikation in eine Datenstruktur für eine Entwicklungsumgebung. Dabei dient die international standardisierte Modellierungssprache EXPRESS als Ausgangsbasis. Das Zielsystem ist die objektorientierte Umgebung Kappa.Die am Institut für Maschinenwesen entwickelte Anwendung hat besondere Relevanz bei der Erstellung von Prototypen, wie sie häufig in praxisbezogenen Projekten gefordert sind.

This publication describes a method to transform a formal specification written in EXPRESS (an internatinal standardised modelling language) onto the object oriented software developement environment Kappa.The developed application supports the creation of prototypes. This has a significant relevance in industrial oriented projects.

1 Motivation

Softwareentwicklung bedeutet nicht allein das Erstellen von Programmzeilen, sondern beschreibt den gesamten Prozeß von der Problemanalyse bis hin zur Validierung des enstandenen Softwareproduktes. Grundlage für die eigentliche Software ist aber immer eine (mehr oder minder) formale Spezifikation, in der sich natürlich auch die aus der Problemanalyse resultierenden Anforderungen niedergeschlagen ha-ben. Diese Spezifikation umfaßt unter anderem eine Festlegung eines Datenmodells (unabhängig von der Implementierung) und der erforderlichen Funktionalität /1/. Je formaler die Spezifikation, desto besser kann die Softwareentstehung (d.h. das Erzeugen von Programmtext) mit Hilfe von anderen Softwarewerkzeugen unterstützt werden.

Viele der am Institut für Maschinenwesen laufenden Projekte erfordern eigene, mittlere Softwareprojekte, die möglichst effizient abgewickelt werden müssen (vergleiche auch /2/). Aufgrund der vielfältigen Tätigkeiten im Bereich der Produktmodellierung und den damit erworbenen Kenntnissen hat sich die für diesen Bereich entwickelte und standardisierte Modellierungssprache EXPRESS /3/ als Spezifikationssprache für Datenstrukturen ergeben. Als unterstützendes System für die Software- und Benutzerschnittstellenerstellung wird die Objektorientiertheit emulierende Entwicklungsumgebung Kappa /4/ eingesetzt.

Dies legt den Schritt nahe, eine weitestgehende automatische Umsetzung von der EXPRESS Spezifikation in die Umgebung von Kappa zu entwerfen.

2 EXPRESS

Das zentrale Element der Modellierungssprache EXPRESS ist das `ENTITY`. Es definiert eine Klasse im Sinne der Objektorientierung. Zur Charakterisierung einer Klasse werden unter anderem auch Attribute herangezogen. Die Attribute selbst werden durch den ihnen zugewiesenen Typ genauer beschrieben. Neben diesen sogenannten expliziten Attributen gibt es noch abgeleitete Attribute (`DERIVE`) , für die eine Herleitung aus beliebig anderen expliziten Attributen existiert. Attribute können `OPTIONAL` sein, d.h. dem Attribut muß bei der Instanziierung nicht notwendigerweise ein Wert zugewiesen werden. Darüberhinaus können Attribute als Schlüsselattribute (`UNIQUE`) oder als invers zu einem anderen Attribut (`INVERSE`) deklariert werden.

Für eine Klasse lassen sich Einschränkungen angeben. Diese können lokal für Attribute der Klasse gültig sein (`WHERE`-Regeln) oder aber global für alle Ausprägungen einer Klasse definiert werden (`RULE FOR`-Regel).

Mit den Schlüsselworten `SUBTYPE OF` und `SUPERTYPE OF` lassen sich Klassenhierarchien bilden. Ein weiteres strukturelles Element sind die Relationen, d.h. der Typ eines Attributs ist als eine Klasse angegeben.

Weitere Sprachelemente sind die `TYPE` Definitionen, mit denen eigene Attributtypen kreiert werden können und Konstanten (`CONSTANT`), die eine modellweite Belegung eines Namens mit einem festen Wert erlauben.

Die alles umfassende Klammer für ein in EXPRESS denotiertes Modell ist ein SCHEMA. Es faßt alle Klassen, die zugehörigen Regeln und gegebenenfalls Abhängigkeiten zu anderen Modelle unter einem Namen zusammen.

Ausführlichere Sprachbeschreibungen sind in /3, 5/ enthalten. Für das Verständnis der Transformation von EXPRESS in Kappa reichen die bisherigen Angaben aber aus.

3 Kappa

Kappa läßt ebenfalls die Definition von Klassen (im Sinne der Objektorientierung) zu, die hier aber *Objects* genannt werden. Ein *Object* ist durch *Slots* charakterisiert. Typdefinitionen für *Slots* im strengen Sinne gibt es in Kappa nicht. Es wird nur angegeben, ob ein *Slot* genau einen Wert (*Single Value*) oder aber mehrere Werte (*Multiple Value*) oder aber eine Methodenbezeichnung (*Method*) aufnehmen soll. Es besteht die Möglichkeit, eine Formel für einen *Slot* anzugeben, die in dem Moment ausgewertet wird, wenn alle angegebenen Parameter tatsächlich zur Verfügung stehen. *Facettes* lassen weitere Attributcharakterisierungen zu ("Attribute von Attributen"), während *Monitors* die Aufnahme von Formeln und Regeln für *Slots* ermöglichen.

Kappa-Objekte können in einer Hierarchie angeordnet werden. Weitere strukturelle Fähigkeiten fehlen jedoch, ebenso eine vordefinierte Schablone für Konstanten.

Eine Struktur in Kappa, bestehend aus *Objects*, deren *Slots* und den *Subtype-Supertype*-Beziehungen, ist vollständig mit Hilfe der graphischen Benutzeroberfläche von Kappa erstellbar. Auf diese Art und Weise ist ein schneller Aufbau einer Datenstruktur gewährleistet. Prinzipiell sind alle Aktionen der graphischen Schnittstelle auch über einen Funktionsaufruf zu erreichen. Dies ist für die Entwicklung von Programmen mit dynamischen Strukturen essentiell. Diese in Kappa angebotene Schnittstelle wird auch für die hier vorgestellte Implementierung genutzt.

4 Ansatz

Der "Umsetzer" LaCuca soll eine Transformation der EXPRESS Strukturen in die Kappa Umgebung vornehmen. Für die Entwicklung von LaCuca sollten möglichst viele Aufgaben von bereits vorhandenen Werkzeugen übernommen werden. So stand zum Beispiel die Implementierung eines weiteren Compilers für EXPRESS außer Frage. Zum Zeitpunkt des Beginns von LaCuca stand ein kommerzielles Softwarepaket von StepTools, Inc. zur Verfügung. Das Paket ST-Developer /6/ enthält verschiedene Komponenten, die für LaCuca verwendet werden konnten:

– Einen Compiler, der EXPRESS Dateien in ein Datenbankschema für die objekt-orientierte Datenbank Rose übersetzt.

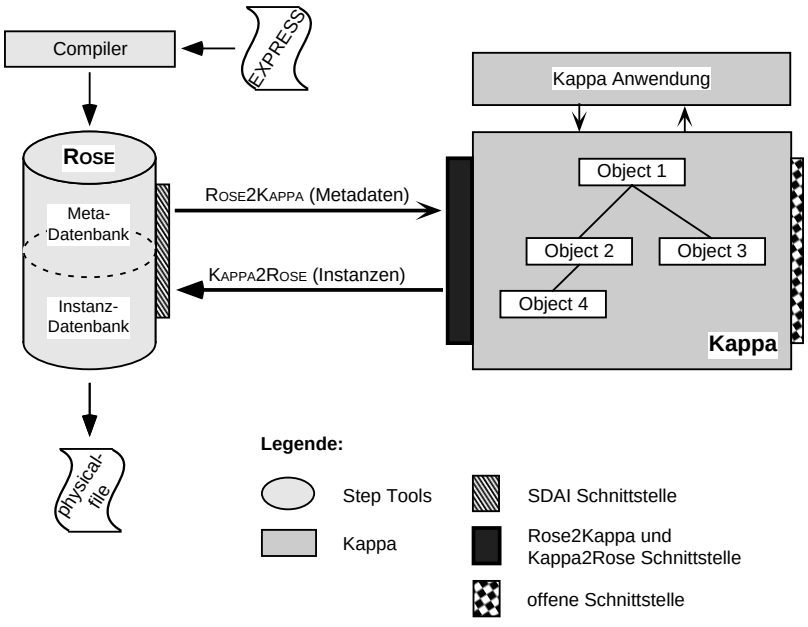

Bild 1: Gesamtkonfiguration von LaCuca

– Zwei Schnittstellen zu dieser Datenbank; eine C++ gebundene und die standardisierte, C gebundene SDAI Schnittstelle /7/.

– Einen Generator für das standardisierten Austauschformat nach ISO 10303-21 /8/.

Der erwähnte Compiler erzeugt zweierlei in der Rose Datenbank. Zum einen wird ein Metaschema angelegt, welches die Informationen über das gerade übersetzte EXPRESS Modell enthält. Diese Informationen können über die SDAI Schnittstelle abgefragt werden. Zum zweiten wird eine Instanzdatenbank vorbereitet, die Instanzen des EXPRESS Modells aufnehmen kann. Diese Instanzdatenbank kann auch über die SDAI Schnittstelle gelesen und beschrieben werden. Abgelegte Instanzen können über ein weiteres Modul als ISO 10303 Austauschformat ("physical file") geschrieben werden.

Der Einsatz dieses Programmpaketes macht nun nur noch eine Anbindung von Kappa an die Rose Datenbank notwendig. Da C als Implemtierungssprache in Kappa vorgesehen ist, wurde die SDAI Schnittstelle gewählt. Dies bewirkte auch eine saubere Trennung von den verschiedenen Softwareprodukten. **Bild 1** zeigt die sich ergebene Konfiguration der eingesetzten Software. `LaCuca` besteht aus den Schnittstellen, die mit Rose2Kappa und Kappa2Rose bezeichnet sind.

5 Abbildung

Sprachkonstrukte in EXPRESS sind auf Konstrukte abzubilden, die Kappa anbietet. Einige der Abbildungen sind kanonisch, weil es Eins zu Eins Entsprechungen der Strukturen sind. Andere erfordern mehr Aufwand, weil direkte Konstrukte in Kappa für EXPRESS Strukturen fehlen oder aber keine direkten Platzhalter hierfür gefunden werden können.

i) `SCHEMA` → *Application*

Die zusammenfassende Klammer für ein Modell in EXPRESS ist das `SCHEMA`. Ein direktes Pendant hierzu in Kappa ist die *Application*. Alle weiteren Konstrukte des Express Modells werden in diese *Application* abbgebildet. Die Applikation erhält den Namen des Schemas in EXPRESS.

ii) `ENTITY` → *Object*

Für jedes `Entity` wird ein *Object* in der Kappa Applikation angelegt. Diese *Objects* werden in der Objekthierarchie der *Application* als direkte oder indirekte Unterobjekte eines neu und einmalig definierten Kappa-Objektes mit Namen "EntityDefs" angeordnet. Alle *Objects*, die unterhalb von "EntityDefs" zu finden sind, entsprechen genau einem `ENTITY`. Die Einführung eines künstlichen Oberobjektes führt zu einer klareren Strukturierung, da auch andere EXPRESS Konstrukte als `ENTITY`s auf *Objects* abgebildet werden.

iii) Attribut → *Slot*

Wird für ein `ENTITY` ein *Object* angelegt, so wird für jedes Attribut des `ENTITY`s ein *Slot* im *Object* erzeugt, der den Namen des jeweiligen Attributes trägt. Für die Typinformationen eines Attributs gibt es keinen adäquaten Platzhalter in Kappa, da *Slots* (wie oben beschrieben) mit wesentlich weniger Informationen charakterisiert werden. Für spätere Implementierungen, die auf dem transferierten EXPRESS Modell aufbauen sollen, können diese spezifischen Typdefinitionen aber durchaus notwendig und nützlich sein. Daher wird auf die *Facettes* von *Slots* zurückgegriffen. Diese Kappa Konstrukte können weitere Charakteristika, die der Benutzer selbst definieren kann, für *Slots* aufnehmen.

Für die Typinformation wird eine *Facette* "type_of_attribute" definiert. Ist der Typ des Attributs ein Basistyp in EXPRESS (`Integer`, `Real`, `Boolean`, `Logical`, `String`), so wird der Name dieses Typs als Zeichenkette in der *Facette* abgelegt. Diese Namen sind für `LaCuca` fest definiert worden und entsprechen den Typbezeichnungen in EXPRESS.

Der Inhalt des *Slots* bleibt bei dieser Abbildung leer. Werden zu einem späteren Zeitpunkt Instanzen dieses *Objects* angelegt, dann nimmt der Inhalt des *Slots* den Wert der Instanz auf.

Schwieriger ist die Abbildung von Relationen zwischen `ENTITY`s, d.h. wenn ein Attribut ein anderes `ENTITY` als Typ besitzt. Für die Abbildung in Kappa bedeutet dies: Zum einen muß festgehalten werden, daß das Attribut als eine Referenz auf ein anderes `ENTITY` zu verstehen ist. Zweitens muß die Referenz selbst abgelegt werden. Das Problem wird gelöst, indem die *Facette* "type_of_attribute" die fest definierte Zeichenkette "REF" zugewiesen bekommt. Sie deutet darauf hin, daß der Inhalt des *Slots* eine Referenz auf ein *Object* enthält; eben genau jenes *Object*, wel-

ches das referenzierte ENTITY repräsentiert.

Bei der Instanziierung eines solchen Falls wird die Referenz auf das *Object* durch eine Referenz auf eine Instanz dieses *Objects* ersetzt, da der Wert des Attributes nicht die Klasse selbst, sondern eben eine Instanz einer Klasse betrifft.

iv) OPTIONAL, DERIVE, UNIQUE, INVERSE

Die möglichen Charakteristika von Attributen in EXPRESS werden nach der gleichen Methode behandelt, wie bei den oben beschriebenen expliziten Attributen. Für den betreffenden *Slot* wird die *Facette* um die Charakteristika "is_optional", "is_derive", "is_unique" und "is_inverse" erweitert. Die Kappa Werte "true" oder "false" werden entsprechend zugeordnet.

v) SUBTYPE und SUPERTYPE

Kappa bietet die Möglichkeit, *Objects* in einer Hierarchie anzuordnen. Diese Hierarchie kann direkt gemäß Hierarchie des EXPRESS Modells übernommen werden. Ist etwa ENTITY A ein SUPERTYPE von ENTITY B, dann ist auch *Object A* ein *Supertype* von *Object B*.

vi) TYPE → *Object*

Ein SCHEMA kann vom Modellierer eingeführte Typdefinitionen enthalten, die mit dem Schlüsselwort TYPE eingeleitet werden. Die Typedefinitionen treten innerhalb des SCHEMAs als Typbezeichnungen von Attributen auf. Die logische Weiterentwicklung der Abbildung für explizite Attribute und Relationen (siehe oben) ergibt, daß für jeden TYPE ein *Object* mit dem Namen der Typdefinition angelegt wird. Referenziert ein Attribut einen solchen Typ, dann enthält die *Facette* "type_of_attribute" des entsprechenden *Slots* die Zeichenkette "REF" und der Inhalt des *Slots* eine Referenz auf das den Typ repräsentierenden *Object*.

Da frei definierte Typen auch als *Objects* abgelegt werden, diese aber keine ENTITYs repräsentieren, werden sie unter einem *Object* mit dem Namen "TypeDefs" angeordnet. "TypeDefs" ist wie "EntityDefs" (für ENTITYs) ein für die LaCuca Applikation neu eingeführtes und vordefiniertes *Object*.

vii) CONSTANT → *Object*

Konstanten sind fest vorgegebene, unveränderliche Größen, die im weiteren Gebrauch mit ihrem Namen referenziert werden. Für sie wird ein fest vorgegebenes *Object* "ConstantDefs" angelegt. Für jede Konstante wird eine Instanz dieses *Objects* mit dem Namen der Konstante erzeugt. Der einzige *Slot* der Instanz nimmt den Wert der Konstante auf.

viii) Problemfälle

Obwohl mit der oben beschriebenen Abbildung wesentliche Teile eines EXPRESS Modells in Kappa übertragen werden können, bleiben noch zwei wesentliche Problemfälle zu lösen.

– Der Mechanismus für die Abbildung von selbstdefinierten Typen läßt sich beliebig fortsetzen, d.h.

Konstrukte in EXPRESS		**Konstrukte in Kappa**
SCHEMA	→	*Application*
ENTITY	→	*Object* unter "EntityDefs"
– Attribut	→	– *Slot*
– Attribut Charakteristika	→	– *Facette* eines *Slots*
Typen		
– einfache Typen	→	– *Facette* "type_of_attribute" (Zeichenkette)
– zusammengesetzte Typen	→	– Referenzmarkierung und *Object*
– selbstdefinierte Typen (TYPE)	→	– *Object* unter "TypeDefs"
CONSTANT	→	Instanz unter "ConstantDefs"
SUBTYPE / SUPERTYPE	→	*Subtype* / *Supertype*

Tab. 1: Abbildung der Begriffe

er ist auch für zusammengesetzte Typen (Aggregate) anwendbar. Die vollständige Abbildung in Kappa scheitert indes an der Unvollständigkeit der Metadatenbank von Rose. Für einen zusammengestzten Typ (etwa ARRAY[1:3] OF INTEGER) läßt sich zwar die Zusammensetzung (hier: ARRAY) aus der Metadatenbank ermitteln, nicht jedoch der Basistyp (hier: INTEGER). Diese Information ist schlichtweg nicht vorhanden.

– Gerade die für die Bewertung von Instanzen wichtigen Informationen, nämlich die durch WHERE-Regeln und RULE definierten Einschränkungen, sind ebenfalls nicht in der Metadatenbank vorhanden. Auch diese Regeln lassen sich nicht, auch nicht als einfache Zeichenkette, über die SDAI Schnittstelle ermitteln.

Während das erste Problem sicherlich mit einer neueren Version des Step Tools-Pakets gelöst werden wird, wird das zweite Problem derzeit aktiv in einer Erweiterung von LaCuca angegangen (siehe unten).

Tab. 1 faßt nocheinmal alle Einzelabbildungen zusammen.

6 Ablauf und Verwendung

Das Ziel der Entwicklung von LaCuca war es, eine Spezifikation in EXPRESS möglichst detailiert in Kappa abzubilden. Die so generierte Kappastruktur kann dann als Grundlage für beliebige Applikationen dienen, die auf diesem Schema basieren und operieren. Aus diesem Grund stand eine möglichst originalgetreue Abbildung im Vordergrund, während auf die trickreiche und vertrackte Nutzung von Kappaoptionen verzichtet wurde. Als Beispiel für diese Philosophie sei die Abbildung der Charakteristika und Typen von Attributen angeführt. Da Kappa in dieser Beziehung sehr unspezifisch ist, wurde keine automatische Typprüfung implementiert. Es bleibt dem Programmierer der jeweiligen Applikation überlassen, wie und ob er Typprüfungen durchführen möchte. Denkbar wären etwa einfache Konverterfunktionalitäten, bei denen von einer korrekten Dateneingabe ausgegangen und auf Typprüfung verzichtet werden kann.

Die momentan implementierten Module führen zunächst nur die Konvertierung von EXPRESS nach Kappa durch. Als vorbereitenden Schritt ist es notwendig, den von Step Tools bereitgestellten Compiler aufzurufen. Dieser verwendet ein EXPRESS Schema, um ein zu diesem Modell passendes Datenbankschema in Rose zu erzeugen. Ist dies geschehen, kann die Konvertierung in Kappa erfolgen. Sie selbst wird von Kappa aus gestartet. Die Angabe des gewünschten Datenbankschemas erfolgt interaktiv während des Programmlaufes. Anschließend wird die Datenbank mit den SDAI Funktionen geöffnet und gelesen. Gemäß der SDAI Spezifikation wandert ein Zeiger über alle abgelegten (Meta-)Daten der Datenbank. Die gelesenen Daten werden in einer einfachen Zwischendarstellung abgelegt, weil die Reihenfolge der Datenbankausgaben für den Aufbau einer Kappastruktur ungünstig ist. Erst wenn die Datenbank wieder geschlossen ist, greifen Kappafunktionen auf die Daten der Zwischendarstellung zu und generieren die entsprechende Struktur in Kappa.

Die Verwendung einer weiteren Darstellungsform unterstützt zwei wesentliche positive Aspekte:

– Das ineffiziente Ausleseverhalten der SDAI Schnittstelle und die damit erzwungene Reihenfolge der gelesenen Daten schlagen nicht auf die Kappa Anwendung durch. Die Zwischendarstellung erlaubt ein stetiges Ablegen der aus der Datenbank gelesenen Daten ebenso wie ein sinnvolles Zugreifen der Kappafunktionen auf diese Daten.

– Es ist eine strikte Trennung von SDAI- und Kappafunktionen und Datentypen möglich. Sowohl die SDAI- als auch die Kappa-Zugriffe sind aus Sicht der Zwischendarstellung vollkommen verborgen. Theoretisch wäre es somit möglich, eine andere Datenbank mit anderen Zugriffsfunktionen zu verwenden.

Ist die EXPRESS-Struktur erst einmal in Kappa verfügbar, steht jede weitere Verwendung offen. Innerhalb der Entwicklungsumgebung können nun die gewünschten Funktionen, die auf den Datenstrukturen arbeiten sollen, implementiert werden.

Dabei ist zu beachten, daß die Transformation mit Klassen gearbeitet hat. Es sind, bis auf die Ablage on Konstanten, keine Instanzen angelegt worden. Dies kann innerhalb der zu entwickelnden Anwendung geschehen. Kappa bietet eine sehr bequeme und schnelle Möglichkeit, Benutzeroberflächen graphisch zu erzeugen. Einer Visualisierung der Instanzen steht somit ebenfalls nichts mehr im Wege.

8 Ausblick

Eine Spezifiation in EXPRESS bedeutet auch die Option, Instanzen des Modells in einem definierten Format austauschen zu können. Dabei fließen die Definitionen des SCHEMAs direkt in das Format mit ein. Die Abbildung von einem Schema zum Austauschformat ist in der ISO Norm 10303-21 /8/ definiert.

Das verwendete Werkzeug von Step Tools generiert nicht nur ein Datenbankschema aus einem EXPRESS Modell, sondern es kann auch Daten, die in der Instanzdatenbank abgelegt worden sind, in dem Austauschformat schreiben. Um also die Fähigkeiten von Step Tools und die Optionen von EXPRESS voll nutzen zu können, ist es notwendig, eine Schnittstelle zwischen der Kappa Anwendung und der Instanzdatenbank zu schaffen. Auch dieses Modul wird SDAI Funktionen verwenden. Dabei sollen alle in der Kappa Anwendung erzeugten Instanzen gemäß der oben beschriebenen Abbildung in die Instanzdatenbank von Rose zurückgeschrieben werden.

Um das Konzept abzurunden, ist eine weitere Schnittstelle zu der Kappa Anwendung erforderlich. Sie ist in Bild 1 als offene Schnittstelle bezeichnet worden, da sie von der jeweiligen Anwendung und deren gewünschten Funktionalität abhängig. Auf alle Fälle aber können an dieser Stelle weitere unterstützende Module von Kappa verwendet werden, etwa die Möglichkeit, eine Abbildung eines externen Datenformates auf vorhandene Objects vorzunehmen.

Weiterhin ist das Problem der nicht in der Rose Datenbank vorhandenen Regeln (WHERE, DERIVE, RULE) zu lösen. Hier wird auf einen externen Compiler zurückgegriffen, der jene Regeln aus dem EXPRESS SCHEMA liest und für Kappa als C Quellcode zur Verfügung stellt.

9 Zusammenfassung

Das hier vorgestelle Werkzeug LaCuca fungiert als Hilfsmittel, um aus in der Modellierungssprache EXPRESS formulierten Modellen möglichst schnell und vollständig Datenstrukturen in der im Institut für Maschinenwesen eingesetzten Entwicklungsumgebung Kappa zu erzeugen. Prototypen, die auf den Modellen operieren sollen, lassen sich somit zügiger erstellen als in der herkömmlichen Art und Weise. Außerdem wird eine gewisse Einheitlichkeit der verschiedenen Prototypen gefördert.

Die Vorteile dieses Ansatzes kommen auch dann zum Tragen, wenn der Datenaustausch bei den Prototypen einer weitere Rolle spielt. Da LaCuca auch eine Schnittstelle zu einer Instanzdatenbank erhalten wird, die mit einem Generator für "physical files" nach ISO 10303-21 gekoppelt ist, lassen sich die in dem Prototypen erzeugten Instanzen sehr leicht in dem standardisierten Format schreiben.

Literatur

/1/ Frick, A.: Der Software-Entwicklungsprozeß - Ganzheitliche Sicht; Carl Hanser Verlag, München 1995

/2/ Heimannsfeld, K.; Teichert, Th.: Informationssysteme des Internets in Forschung und Lehre; Institutsmitteilung des IMW, Clausthal-Zellerfeld 1995

/3/ ISO/TC184/SC4: ISO 10303: Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual; IS, ISO 1994

/4/ IntelliCorp: Kappa User Manual, 1994

/5/ Schenk, D.; Wilson, P.: Information Modeling the EXPRESS Way, Oxford University Press 1994

/6/ Step Tools Inc.: STEP Utilities Reference Manual, 1994

/7/ ISO/TC184/SC4: ISO 10303: Product data representation and exchange - Part 22: Implementation methods : Standard Data Access Interface; CD, ISO 1994

/8/ ISO/TC184/SC4: ISO 10303: Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure; IS, ISO 1994