Ausdrückliche Definition von Komponenten(objekte)

Rony G. **Flatscher**, Prof. Dr., E-Mail: Rony.Flatscher@wu-wien.ac.at
Wirtschaftsuniversität Wien (http://www.wu-wien.ac.at),
Abteilung für Wirtschaftsinformatik (http://wwwi.wu-wien.ac.at),
Augasse 2-6
A-1090 Wien/Vienna

Abstract. Im Zusammenhang mit der Definition von Komponenten(objekten) erscheint es als wünschenswert, die Zusammensetzung von Komponententeilen ausdrücklich zu modellieren und damit einer Reflektion zur Laufzeit von Komponenten zugänglich zu machen. In dieser Arbeit wird zunächst eine Technik vorgestellt, die im Rahmen der Erstellung des EIA/CDIF-Standards über Jahre hinweg entwickelt wurde. Daran schließt sich eine kurze, kritische Würdigung dieses Ansatzes an sowie ein Lösungsvorschlag für ein erkanntes und als wichtig angesehenes Problem.

1. Einleitung

In der Wirtschaftsinformatik werden Ganze-/Teilebeziehungen[1] üblicherweise mit dem Beziehungstyp „is_part_of“ beziehungsweise mit der Umkehrung davon „aggregate“ ausgedrückt (vgl. beispielsweise [FerSin98] oder [Wede92]). Hierbei wird allerdings nicht ausdrücklich zwischen der *Definition* eines aus Komponenten bestehenden Objekttyps („das Ganze“) und seiner konstituierenden Teile („den Teilen“) unterschieden.
Die amerikanische Industriestandardisierungsorganisationsinitiative „EIA/CDIF“ („Electronic Industries Association/Alliance; CASE Data Interchange Format“), die ihre Arbeit 1987 begonnen hat, setzte sich Jahre mit diesem Problem auseinander, das sich vor allem mit der Definition von Metamodellen für die Datenflußmodellierung (vgl. [CDIF96a]) und die konzeptionelle Datenmodellierung (vgl. [CDIF96b]) stellte.[2]

---

[1] [Wede92] spricht in diesem Zusammenhang von „mereologischen Strukturen“.

[2] Eine Aufbereitung sämtlicher standardisierter EIA/CDIF-Metamodelldefinitionen in Form von HTML-Dateien findet sich in [W3CDIF_M2].

2. EIA/CDIF's Allgemeiner Strukturierungsmechanismus[3] (ASM)

Der „General Structuring Mechanism“ (GSM, deutsch: Allgemeiner Strukturierungsmechanismus, abgekürzt: ASM) wurde von EIA/CDIF zum ersten Mal in [CDIF96a] eingeführt und erklärt. Abbildung 1 stellt ein erweitertes Entity-Relationship-Diagramm dar, das die Entitätstypen und Beziehungstypen des ASM darstellt.

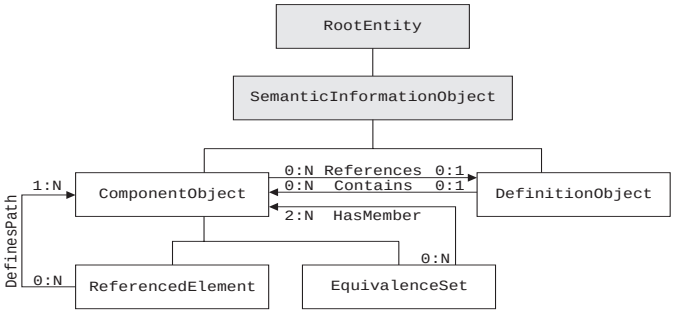

Abbildung 1: EIA/CDIFs Allgemeiner Strukturierungsmechanismus.[4]

Die Definition von Entitätstypen, die aus weiteren Entitätstypen zusammengesetzt sind, erfolgt ausdrücklich, indem dafür ein eigener Entitätstyp „DefinitionObject“ (DO; deutsch: „DefinitionsObjekt“) vorgesehen ist, der aus beliebig vielen „ComponentObject“ (CO; deutsch: „“KomponentenObjekt“) zusammengesetzt sein kann. Dies ergibt sich aus dem Beziehungstyp „0:1 DefinitionObject.*Contains*.ComponentObject 0:N“. Ein DefinitionsObjekt kann selbst als Komponente in einem anderen DefinitionsObjekt auftreten, indem es über den Beziehungstyp „0:N ComponentObject.*References*.DefinitionObject 0:1“ referenziert wird. Ein KomponentenObjekt kann hierbei, wie dies aus den Kardinalitäten hervorgeht, genau ein DefinitionsObjekt referenzieren. Somit können die Bestandteile (KomponentenObjekte) eines Ganzen (DefinitionsObjekt) selbst aus Strukturen bestehen, die nach demselben Schema erstellt worden sind.

Durch die ausdrückliche Trennung von DefinitionsObjekt und KomponentenObjekt wird es möglich, weitere Informationen ausdrücklich hinzuzufügen: [5]

- Der Beziehungstyp „0:N **ReferencedElement**.*DefinesPath*.ComponentObject **1**:N“ ermöglicht das eindeutige Referenzieren von jenen KomponentenObjekten, die in einem

---

[3] Vgl. in diesem Zusammenhang auch die Ausführungen in [Flat98], S. 229ff.

[4] Entnommen aus [Flat98], S.230.

[5] Vgl. auch die entsprechenden Ausführungen in [CDIF96a], [Ernst98] sowie [Flat98].

Modell möglicherweise für verschiedene DefinitionsObjekte gemeinsam benutzt werden. Dies wird durch das Festlegen von entsprechenden Pfaden realisiert.

- Der Beziehungstyp „0:N **EquivalenceSet**.*HasMember*.ComponentObject **2**:N“ erlaubt das semantisch ausdrückliche Gleichsetzen von zwei ansonsten verschiedenen KomponentenObjekten.

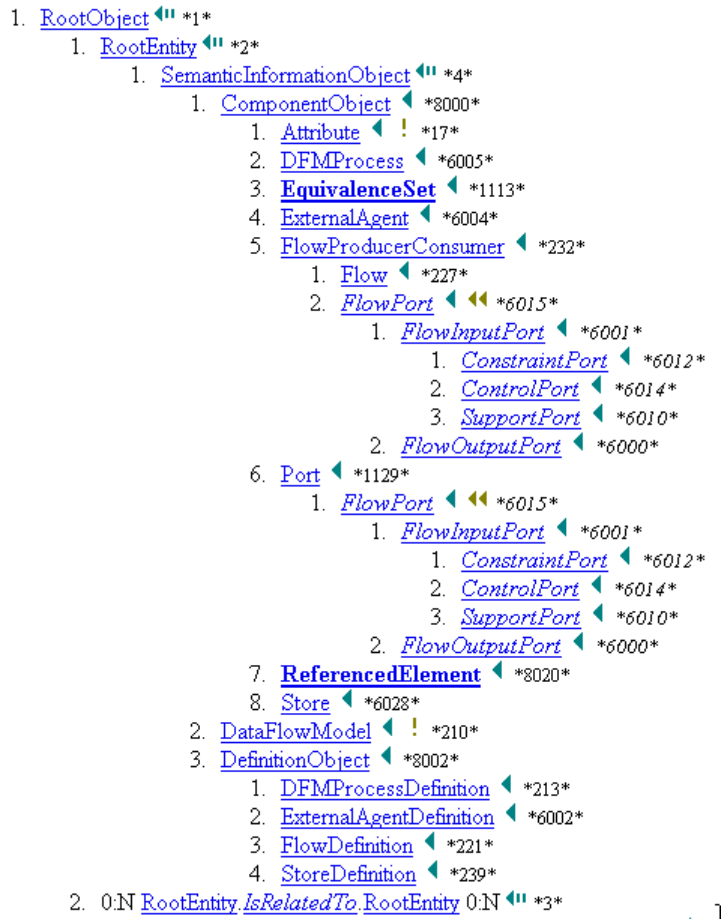

Abbildung 2: Ausschnitt aus dem EIA/CDIF Metamodell „Datenflußmodellierung“.[6]

3. Kritische Diskussion des EIA/CDIF’s ASM

Der für die Erstellung der standardisierten EIA/CDIF-Metamodelle eingesetzte Allgemeine Strukturierungsmechanismus erlaubt es, ausdrücklich die Definition von Ganzen und seinen Teilen voneinander zu unterscheiden.

Abbildung 2 zeigt einen Ausschnitt[7] der Generalisierungshierchie des standardisierten EIA/CDIF Metamodells für die Datenflußmodellierung[8]. Aus dieser Abbildung kann

---

[6] Kursiv dargestellte Bezeichner weisen auf den Einsatz von Mehrfachvererbung hin.

eindeutig erkannt werden, welche Entitätstypen DefinitionsObjekte und welche KomponentenObjekte sind.

In diesem Metamodell wird beispielsweise das DefinitionsObjekt „StoreDefinition“ aus den KomponentenObjekten „Store“, „FlowPort“, „Flow“ und „Attribute“ gebildet, das DefinitionsObjekt „DFMProcessDefinition“ aus den KomponentenObjekten „DFMProcess“, „Store“, „ExternalAgent“, „FlowPort“, „Flow“ und „Attribute“.

Allerdings können diese Bildungsregeln nicht aus den Strukturen der Generalisierungshierarchie selbst *nicht* abgeleitet werden, sondern sind in den entsprechenden EIA/CDIF-Metamodellstandards in den begleitenden Texten festgelegt. Dieses Problem besteht, da das EIA/CDIF Meta-Metamodell (vgl. [CDIF94]) keine Möglichkeit vorsieht, die KomponentenObjekte aus denen ein bestimmtes DefinitionsObjekt bestehen darf ausdrücklich zu definieren.

In [Flat98] wurde ein Versuch unternommen, dieses Problem ohne Änderung des EIA/CDIF-Meta-Metamodells zu lösen. Dazu wurde ein EIA/CDIF-konformes Metamodell namens „M2Level“ spezifiziert, in dem mit Hilfe von Referenzen auf MetaObjekte (Instanzen von Entitätstypen des Meta-Metamodells, die selbst die Metamodelle bilden) Bezug genommen wird. Somit wird es möglich, auf Metamodellebene ausdrücklich vorzusehen, welche DefinitionsObjekte aus welchen KomponentenObjekten bestehen dürfen. Abbildung 3 stellt die Generalisierungshierarchie von „M2Level“ dar.

Mit Hilfe des Beziehungstyps „0:1 **SetOfValidComponents**.*ForBuildingStructureOf*.-DefinitionObjectReference **1**:1“ kann eine Menge („SetOfValidComponents“) von KomponentenObjekte für ein beliebiges DefinitionsObjekt festelegt werden. Die entsprechend zulässigen KomponentenObjekte werden über „0:N **SetOfValidComponents**.*Contains*.-ComponentObjectReference **1**:N“ definiert.

---

[7] Dieser Ausschnitt stammt aus den öffentlich zur Verfügung stehenden HTML-Dateien von den EIA/CDIF-Metamodelldefinitionen in [W3CDIF_M2] und zeigen u.a. auch die vordefinierten Surrogatwerte für die spezifizierten EIA/CDIF-MetaObjekte.

[8] Vgl. hierzu [CDIF96a], [Flat98] beziehungsweise [W3CDIF_M2].

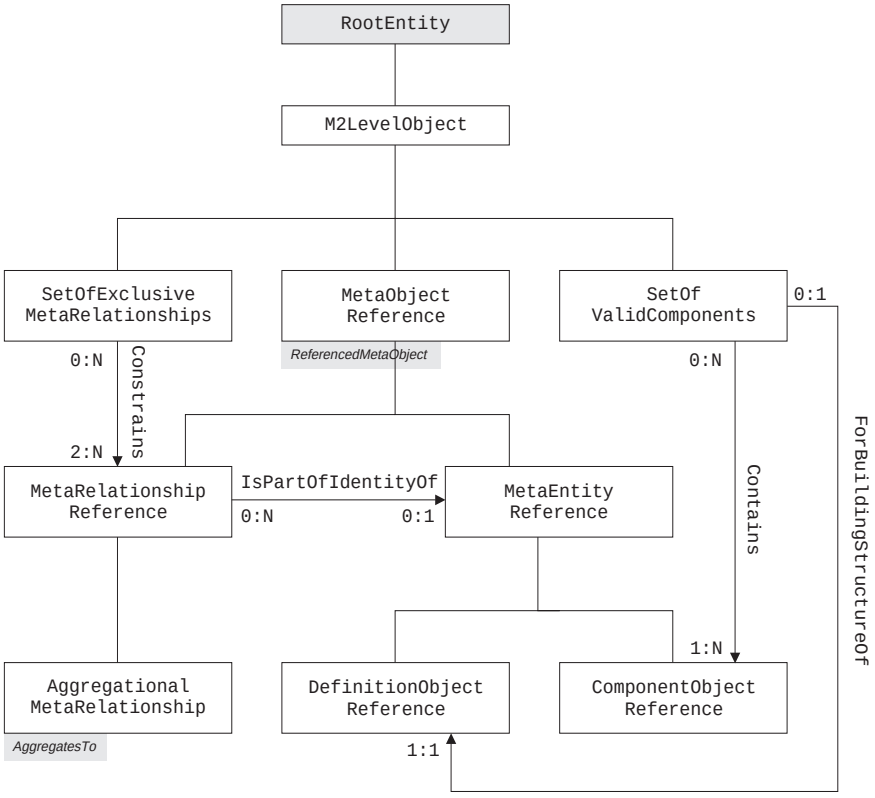

Abbildung 3: Das Metamodell „M2Level“.[9, 10]

4. Zusammenfassung

In dieser Arbeit wurde versucht, die ausdrückliche Definition von Komponenten(objekten) anhand des Allgemeinen Strukturierungsmechanismus von EIA/CDIF vorzustellen. Die Definition von Ganzen wird ausdrücklich mit Hilfe der Entitätstypen „DefinitionObject“ und „ComponentObject“ ermöglicht. Durch diese Trennung wird möglich, für die Wiederverwendung von gemeinsam genutzten KomponentenObjekten auch eindeutige Pfade anzugeben, wozu der Entitätstyp „ReferencedElement“ herangezogen wird. Eine weitere, wichtige Anwendung besteht darin, daß man paarweise KomponentenObjekte mit Hilfe des Entitätstyps „EquivalenceSet“ ausdrücklich gleichsetzen kann.

Ein wesentlicher Mangel im EIA/CDIF-Ansatz wird darin gesehen, daß die für die Erstellung eines DefinitionsObjektes erlaubten KomponentenObjekte nicht innerhalb von EIA/CDIF-entsprechenden Generalisierungshierarchien angegeben werden können. Durch die

---

[9] Entnommen aus [Flat98], S.357.

[10] Weitere Ausführungen und Diskussionen zum Metamodell „M2Level“ sowie zu den weiteren Problemen, die damit gelöst werden sollen, finden sich in [Flat98], S. 356ff.

Einführung von Referenzen auf die MetaObjekte des EIA/CDIF Meta-Metamodells wird es möglich, auf Metamodellebene dieses Problem zu lösen. Hierzu wurde das Metamodell „M2Level“ vorgestellt, in dem Teile ausdrücklich für die Lösung dieses Mangels entworfen wurden.

5. Quellenverzeichnis

[CDIF94] „CDIF – Framework for Modeling and Extensibility“, Interim Standard, EIA/IS-107, EIA 1994.

[CDIF96a] „CDIF – Integrated Meta-model, Data Flow Subject Area“, Interim Standard, EIA/IS-115, EIA 1996.

[CDIF96b] „CDIF – Integrated Meta-model, Data Modeling Subject Area“, Interim Standard, EIA/IS-114, EIA 1996.

[Ernst98] Ernst J.: „Contributions to the Integration of Tools and Techniques for the Development of Heterogeneous Embedded Real-Time Systems“, Forschungsbericht des Forschungszentrums für Informatik (FZI), Karlsruhe 1998.

[FerSinz98] Ferstl O.K., Sinz E.J.: „Grundlagen der Wirtschaftsinformatik, Band 1“, Oldenbourg, München/Wien 1998.

[Flat98] Flatscher R.G.: „Meta-Modellierung in EIA/CDIF“, ADV-Verlag, Wien 1998.

[W3CDIF] WWW-Homepage of the „Electronic Industries Association (EIA), CASE Data Interchange Format (CDIF) committee“, EIA/CDIF. URL (1999-04-09): „`http://www.eigroup.org/cdif/index.html`“.

[W3CDIF_M2] HTML-Aufbereitungen sämtlicher EIA/CDIF-Metamodelldefinitionen. URL (1999-04-09): „`http://www.wu-wien.ac.at/wi/cdif`“.

[Wede92] Wedekind H.: „Objektorientierte Schema-Entwicklung“, BI Wissenschaftsfverlag, Mannheim/Wien/Zürich 1992.