Schweizerische Zentralstelle für Baurationalisierung CRB

Projekt OGIP

OGIP Wissenschaftlicher Schlussbericht

Optimierung der Gesamtanforderungen (Kosten/Energie/Umwelt) - ein Instrument für die Integrale Planung

Autor Heitz Sandro, t.h.e. Software GmbH, Karlsruhe, Deutschland

Inhaltsverzeichnis Seite

0 Grobziel

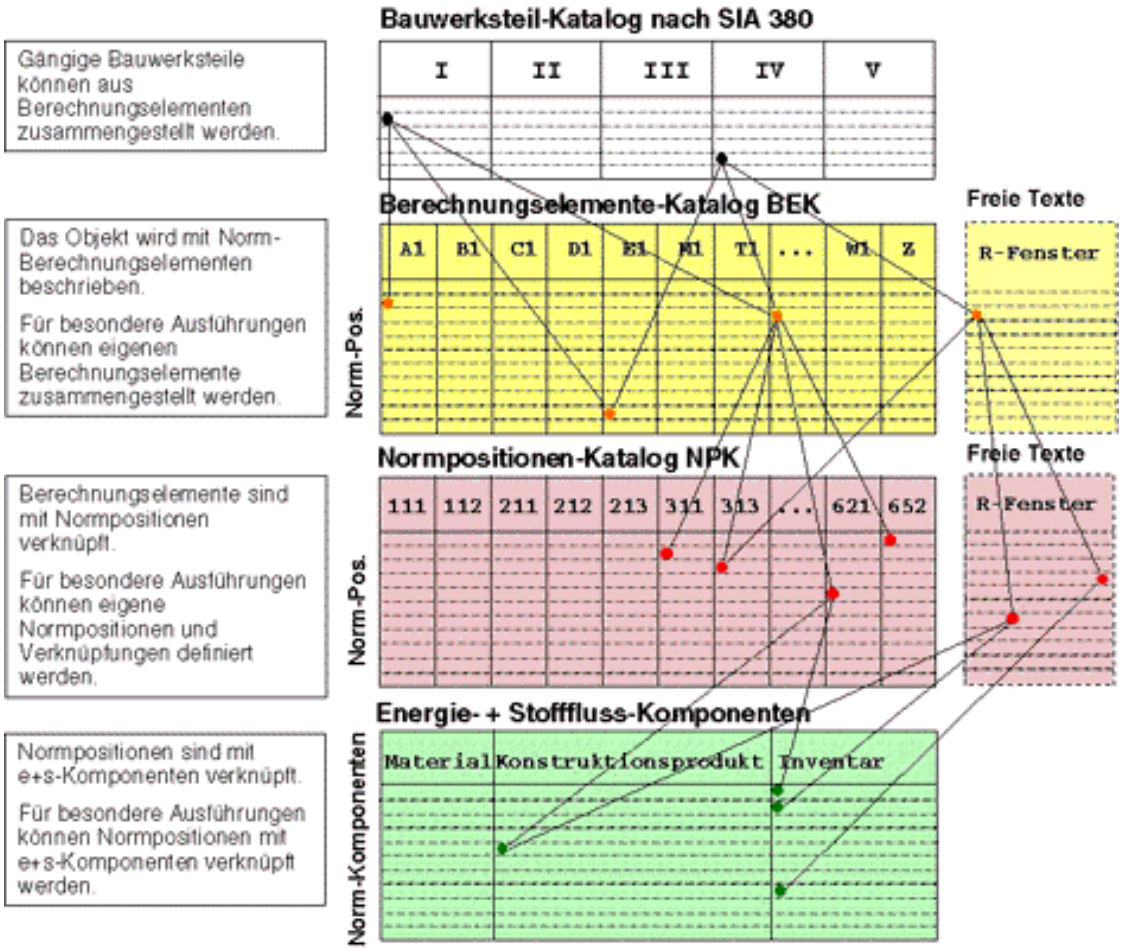
Entwicklung eines in die CRB-Arbeitsmittel integrierten Anwenderprogramms für die Energie- und Stoffflussbilanzierung von Gebäuden.

1 Systemziel

Das Projekt OGIP hatte zum Ziel ein Instrument zu schaffen, welches auf der Grundlage eines standardisierten Verfahrens die Berechnung von Kennwerten für Direkte- und Externe Baukosten, Graue Energie und Umweltbelastung auf der Grundlage einer durchgängigen Methodik (Ausführungsbeschreibung mittels BEK) ermöglichen sollte. Das Instrument sollte praxistauglich sein, indem es in die Arbeitsmittel des CRB (BEK und NPK) integriert werden sollte. Das Instrument sollte den Vergleich von Baustoffen, Konstruktionen oder auch ganzen Gebäuden ermöglichen und es sollte die Möglichkeit bestehen Referenzobjekte ablegen zu können. Das Arbeitsinstrument sollte (wie andere Arbeitsinstrumente, zB. Kostenplanung) die Transparenz der Daten von BEK (Berechnungselemente-Katalog) und NPK (Normpositionen-Katalog) bis zu den e+s-Komponenten (Baustoffe, Maschine, Produkte) sicherstellen. Des weiteren war die Berechnung des Energieverbrauchs nach SIA 380/1 „Energie im Hochbau“ vorgesehen. Diese sollte in die Arbeitsmittel des CRB integriert sein.

2 Kurzdarstellung der Projektergebnisse

In OGIP wurde in allen Teilen auf die Arbeit aus dem Projekt OGIP/DATO zurückgegriffen. Änderungen ergaben sich nur durch die Integration der „Langtexte“ des Berechnungselemente-Katalogs, die Integration des Detailaufbaus und durch eine Veränderung des Aufbaus von „freien Berechnungselementen“. Auf Anregung der beteiligten Fachverbände werden in OGIP „freie Berechnungselemente“ auf der Grundlage von vorhandenen Berechnungselementen definiert und die Energie- und Stoffflussbilanzierung findet nicht in „Schichten“ des Elements, sondern über den Detailaufbau und die Normpositionen statt. Die folgende Abbildung zeigt die Struktur von OGIP auf. Sie reicht von den Bauwerksteilen (ganz oben), welche für die Berechnung der SIA 380/1 herangezogen werden, über den Berechnungselemente-Katalog, den Normpositionen-Katalog bis zu Energie- und Stoffffluss-Komponenten (ganz unten).
Aus der Grafik wird auch ein wichtiges Prinzip von OGIP deutlich. Der Anwender kann alle Datensätze mit „freien Einträgen“ erweitern, bis auf die Energie- und Stoffffluss-Komponenten. Diese sind abschliessend und nicht veränderbar werden jedoch bei einer Neuausgabe des Programms aktualisiert.



Wie der Abbildung zu entnehmen ist, ist OGIP vollständig in die Arbeitsmittel des CRB integriert. Die direkten Baukosten werden auf der Grundlage der Richtpreise des NPK berechnet. Die externen Baukosten und die Umweltbelastung auf der Grundlage der Kennwerte der Energie- und Stoffffluss-Komponenten. Der Energieverbrauch nach SIA 380/1 wird auf der Grundlage von Bauwerksteilen berechnet unter der Angabe von Energieerzeugungsarten, welche selbst wieder zu den Energie- und Stoffffluss-Komponenten gehören. Der Vergleich von Baustoffen, von Konstruktionen oder auch ganzen Gebäuden ist möglich, wenn auch umständlich, da OGIP immer die Struktur eines „Objekts“ oder „Projekts“ voraussetzt. Es folgt hier den anderen Arbeitsinstrumenten, die auf der Grundlage von CRB-Arbeitsmitteln realisiert sind (Kostenplanung, Devisierung). OGIP bietet die vollständige Transparenz über alle Detaillierungsstufen, wenn auch gesagt werden muss, dass der Durchgriff von Berechnungselementen auf Normpositionen, Energie- und Stoffflusskomponenten und deren Kennwerten etwas langwierig ist. Bedingt durch die Tiefe der Strukturierung aber auch durch die Benutzerführung des Anwenderprogramms.

3 Darstellung einiger Kritikpunkte am Projektergebnis

Durch die Integration von OGIP in die CRB Arbeitsmittel ist die Kenntnis der Elementmethodik eine zwingende Voraussetzung des Gebrauchs von OGIP. Darüber hinaus ergibt sich durch die verschiedenen Arbeitsmittel des CRB eine Anwendungslogik, die in manchen Fällen der Benutzung von OGIP zu einem für den Anwender unnötigen Eingabeaufwand führt (immer betrachtet in Bezug auf sein Arbeitsziel). Ein weiteres Problem ist die Masse an Daten und die grosse Detaillierung, wobei BEK und NPK aber nur die Gegebenheiten der Praxis widerspiegeln. Aus diesen Gründen ist für den

Gebrauch von OGIP eine Einführung (Besuch eines Kurses) unumgänglich. Durch die hohe Integrierung von OGIP stellen sich Berechnung durch OGIP dem Anwender als „black box“ dar. Ein Nachvollzug ist theoretisch zwar möglich aber faktisch nicht machbar. An dieser Stelle sei jedoch darauf verwiesen, dass Kostenplanungsprogramme zum Teil unter dem gleichen Problem (aus Sicht des Anwenders) leiden, wobei dort die Anwender aber über Vorstellungen verfügen, welche Grössen angemessen sind oder nicht. In Bezug der Umweltbelastung gibt es diese Vorstellungen nicht und dementsprechend hoch ist der Wunsch nach Transparenz. Weitere Kritikpunkte sind technischer Art und auf den beta-Status des Anwenderprogramms zurückzuführen.
Bei den Energie- und Stoffffluss-Komponenten wurden einige als Fehler erkannt und behoben. Andere stehen noch im Verdacht, Mängel aufzuweisen. Die Transformationen der Normpositionen konnten aufgrund der Masse der Daten nur auf „Ausrutscher“ hin untersucht werden (besonders grosse Werte, zu kleine Werte). Fehler können somit nicht ausgeschlossen werden.

4 Projektergebnisse im Einzelnen

4.1 Integration in die CRB-Arbeitsmittel

OGIP ist (wie es ja auch der Abbildung zu entnehmen ist) vollständig in die Arbeitsmittel des CRB integriert. Erweitert wurden die Arbeitsmittel des CRB durch die Energie- und Stoffffluss-Komponenten, durch die Bauwerksteile und durch zusätzliche Angaben auf Stufe Berechnungselement.

4.1.1 Normpositionen und Energie- und Stoffffluss-Komponenten

Normpositionen beschreiben Leistungen, die erbracht werden müssen. Diese Leistungen setzen sich zusammen aus Angaben zum:

- Lohn
- Inventar
- Maschinen
- Material

Lohn ist für die Energie- und Stoffflussbilanzierung nicht von Interesse. Die restlichen Angaben werden in der Transformation erfasst und in entsprechenden Einträge abgebildet:

- Maschine
  - Laufzeit
  - Vorhaltezeit
    (aus ihr wird die Laufzeit mittels Daten aus der SBIL berechnet)
  - Entfernung
    (nur mit einer Angabe zur Masse)
- Inventar
  - Hilfs- oder Arbeitsmittel
  - Menge
  - Verwendbarkeit oder Vorhaltezeit
    (aus ihnen wird der Grad der Abnutzung berechnet)
- Material
  - Konstruktionsprodukt
  - Flussrichtung
    (wird das Material eingebracht oder ist das Material zu entsorgen)
  - Menge

Aus den Angaben kann dann auf der Grundlage der bewerteten kumulierten Sachbilanzen die Energie- und Stoffflüsse berechnet werden.

4.1.2 Berechnungselemente

Die Berechnungselemente wurden in OGIP um Angaben zur Lebensdauer und zum Wärmedurchgangswiderstand erweitert. Die Lebensdauer ist notwendig für die Berechnung der Energie- und Stoffflüsse während des Betriebs (Unterhalt, Erneuerung), die Wärmedurchgangswiderstände sind notwendig für die Berechnung des k-Wert (U-Wert) in der SIA 380/1, welche über die Bauwerksteile geschieht.

4.1.3 Bauwerksteile und Berechnungselemente-Katalog

Um die SIA 380/1 berechnen zu können, mussten die Arbeitsmittel des CRB um das Konzept der Bauwerksteile erweitert werden, da die Berechnungselemente selbst im den meisten Fällen keine Konstruktionen nach SIA 380/1 darstellen, sondern nur Teile von Konstruktionen (zB. die Aussendämmung).
In OGIP kann der Anwender Berechnungselemente deshalb in Bauwerksteile zusammenfassen und somit einen Schichtenaufbau für die k-Wert (U-Wert) Berechnung erzeugen.

4.2 Baukosten

Die Baukosten werden in OGIP auf der Grundlage der Kostenrichtwerte des Normpositionen-Katalogs berechnet. Diese können über den Detailaufbau zum Kostenrichtwert eines Berechnungselements zusammengefasst werden und auf der Stufe Objekt aggregiert werden. Die Kostenrichtwerte können nur in „freien Positionen“ beeinflusst werden.

4.3 Externe Baukosten, Kosten des Betriebs und Umweltbelastung

Bei jeder Berechnung wird von den Energie- und Stoffffluss-Komponenten ausgegangen. Ihnen hängen bewertete Sachbilanzen und einigen von ihnen auch Kostenrichtwerte an. Auf der Grundlage dieser Daten werden alle Berechnungen durchgeführt. Eine detaillierte Beschreibung findet sich unter „Berechnung der Energie- und Stoffflüsse“.

4.4 Baustoffe, Konstruktionen und Gebäude

Der Vergleich von Baustoffen und Konstruktionen ist in OGIP nicht unmittelbar möglich. Bei Konstruktionen müssen Objekte angelegt, die Konstruktionen in gleichen Mengen eingefügt und am Ende Auswertungen gemacht werden, damit die Konstruktionen verglichen werden können.
Baustoffe können in OGIP betrachtet werden aber nur über Umwege direkt verglichen werden (in Form einer Auswertung). Hierzu müssen freie Berechnungselemente aufgebaut, Objekte erzeugt, die Berechnungselemente in gleichen Mengen eingefügt und am Ende Auswertungen gemacht werden, damit die Baustoffe über Grafiken miteinander verglichen werden können.

5 Funktionalität von OGIP im Einzelnen

5.1 Umweltbilanz

Mit OGIP können Umwelteinwirkungen von Bauwerksteilen und Bauwerken berechnet und beurteilt werden. Eine Ökobilanz umfasst in der Regel den gesamten Lebenszyklus eines Produktes bzw. Bauwerkes, von der Bereitstellung der Rohstoffe für Baustoffe oder Bauteile, der Errichtung des Bauwerkes, der Nutzung, der Bauwerkserhaltung bis zum Abbruch und der Entsorgung (s.a. „Berechnung der Energie- und Stoffflüsse“). Die ökologische Wirkungsbilanz umfasst die gesamten Energie- und Stoffflüsse, d.h. den Ressourcen- und Energieverbrauch, die Luft-, Wasser- und Bodenbelastungen sowie die Abfallmenge und deren Entsorgung.

5.1.1 Ermittlung des Ökoinventars eines Bauwerks
In OGIP wird die Ausführung eines Bauwerkteils bzw. ganzen Bauwerks mit den Berechnungselementen (BEK) beschrieben. Bei der Auswertung der Umweltbilanz werden im Hintergrund dann die entsprechenden Energie- und Stoffflüsse, die sog. Ökoinventare, aller beteiligten Prozesse berücksichtigt.

5.1.2 Erstellen der Wirkungsbilanz
Das Ökoinventar muss nun nach einer anerkannten Methode, welche die Auswirkungen auf die Umwelt beschreibt, bewertet (Gewichtung) werden. In OGIP wird mit den Ökofaktoren 1997 (BUWAL SR Nr. 297) bewertet welche die Wirkungen in Umweltpunkten UBP ausdrücken.

5.1.3 Auswertung
Das Ökoinventar bzw. dessen Bewertung (Wirkungsbilanz) muss sachkundig interpretiert werden. Sinnvollerweise werden Varianten eines Bauvorhabens und evtl. auch geeignete Referenzobjekte aus dem SKK (Stofffluss-Kennwerte-Katalog) miteinander verglichen. Durch den direkten Vergleich können ökologische Schwachstellen aufgedeckt sowie Handlungsoptionen und Verbesserungsvorschläge ausgearbeitet werden. Die Ökofaktoren werden in OGIP auf gebräuchliche Nutzungs-Einheiten (wie in der Kostenplanung z.B. Wohnungseinheiten, Büroplatz) bezogen.

5.1.4 Grundlagen der Umweltbilanzierung in OGIP
Grundlage der Energie- und Stofffluss-Daten in OGIP sind die sog. e+s-Komponenten (Energie- und Stofffluss-Komponenten), die aus den Normpositionen (NPK) hergeleitet und zum sog. e+s-Inventar der Position zusammengesetzt worden sind. e+s-Komponenten sind Maschinen (Kranzug, Lastwagen-Stunden), Inventardaten (Gerüste, Signaltafeln, Schalungen usw.) oder auch Produkte (Beton, Alufolie, Fenster, Türen usw.). Für die Normpositionen des NPK wurden die e+s-Inventare aus den Kalkulationsgrundlagen der Fachverbände abgeleitet. In OGIP können über Reservepositionen eigene Berechnungselemente oder Normpositionen erzeugt und in die Bilanzierung von Bauvorhaben eingebunden werden.

5.1.5 Was OGIP nicht ist
OGIP ist kein Werkzeug, um Produkte bzw. Fabrikate (Türen, Fenster, Ventilatoren, Kücheneinrichtungen, Sanitärgegenstände usw.) miteinander zu vergleichen oder zu bilanzieren. Die verwendeten Ökoinventare sind genau wie die Kosten Durchschnittswerte, und lassen deshalb solche Detailanalysen nicht zu.

5.2 Energie

In OGIP werden Heizenergiebedarf, Energiebedarf für Warmwasser und Elektrizitätsverbrauch auf der Grundlage der Eingabe von "Bauwerksteilen", deren Mengengliederung, den Klimadaten aus SIA 381/2 "Klimadaten", und modifizierter Standardnutzungen nach SIA 380/1 "Energie im Hochbau" berechnet und in der Energiekennzahl zusammengefasst. Die Energiekennzahl ist in OGIP die Summe der Teilenergiekennzahlen der Endenergieträger, (SIA 180/4 "Elektrische Energie im Hochbau, S. 6.)

5.2.1 Heizenergieverbrauch
Berechnet wird in OGIP der Jahresenergieverbrauch, welcher sich unter Berücksichtigung der Verluste der Wärmeerzeugung aus dem Heizenergiebedarf bestimmen lässt.

5.2.2 Energieverbrauch für Warmwasser
Der Energieverbrauch für Warmwasser wird in OGIP aufgrund der modifizierten Standardnutzungen der Tabelle D 1-1, SIA 380/1, S. 65 berechnet.

5.2.3 Elektrizitätsverbrauch
Der Elektrizitätsverbrauch wird auf der Grundlage der Standardnutzungen der modifizierten Tabelle D 1-1, SIA 380/1, S. 65 festgelegt.

5.2.4 Was OGIP nicht ist!
OGIP ist kein Programm, um den behördlichen Nachweis nach SIA 380/1 zu führen. Die Nachweismöglichkeit ist für die Vollversion vorgesehen.

5.3 Kosten

OGIP beruht auf der EKG (Elementkostengliederung), dem BEK (Berechnungselemente-Katalog) und dem NPK (Normpositionen-Katalog).

5.3.1 Erstellungskosten
Berechnet werden in OGIP die Erstellungskosten auf der Grundlage der Kostenrichtwerte des NPK, welche entlang des Detailaufbaus in den einzelnen Berechnungselementen zusammengeführt werden. Der Kostenrichtwert der einzelnen Berechnungselemente (BEK-Positionen) wird dann aufgrund der Mengengliederung der BEK-Positionen im Objekt aufaggregiert.

5.3.2 Kosten der Erneuerung
Eine Prognose der Kosten der notwendigen Erneuerungen eines Bauwerks ist nur auf kurze Zeit ohne grössere Abweichungen berechenbar. In OGIP wird jedoch kein kurzer Zeitraum, sondern die mutmassliche Lebensdauer des Objekts betrachtet. Im Augenblick werden in OGIP aufgrund dieser Problematik die Kosten für die Erneuerung prognostisch auf der Grundlage der Kostenrichwerte für den Neubau der einzelnen BEK-Positionen ermittelt. Unterschlagen werden hierbei die Kosten der Entsorgung der anfallenden Stoffe. Mutmassliche Kostensteigerungen, die vielleicht aus der Vergangenheit hätten gewonnen werden können, werden in OGIP ignoriert. Die Kosten der Erneuerung entsprechen in OGIP der jährlichen Abschreibung der einzelnen Berechnungselemente unter der Berücksichtigung der Lebensdauern wie sie durch das Amt für Bundesbauten (jetzt Bundesamt für Bauten und Logistik) festgesetzt wurden.

5.3.3 Rückbaukosten (Kosten des Abrisses)
In OGIP werden keine Kosten für den Rückbau veranschlagt, da hierfür noch keine Kostenrichtwerte im BEK existieren.

5.3.4 Betriebskosten
In OGIP werden nur für die Betriebskosten nach SIA 380/1 "Energie im Hochbau" berechnet. Es werden heute übliche Energiepreise zugrunde gelegt, ohne dass auf regionale Unterschiede eingegangen wird.

5.3.5 Was OGIP nicht ist
OGIP ist kein Kostenplanungsprogramm. Die Kostenberechnung in OGIP basiert nicht auf Kostenkennwerten, die aus abgerechneten Projekten des "eigenen Büros" gewonnen wurden. In OGIP kann kein Kennwertaufbau mit Marktpreisen wie in den üblichen Kostenplanungsprogrammen angelegt werden. OGIP folgt streng und einzig den Kostenrichtwerten wie sie der NPK als Arbeitsmittel des CRB vorgibt.

6 Berechnung der Energie- und Stoffflüsse

Die Berechnung der Energie- und Stoffflüsse folgt einem streng hierarchischen Modell, wobei es zwei Berechnungspfade gibt:

- Berechnung der SIA 380/1, Ausgabe 1988
- Berechnung der Energie- und Stoffflüsse

6.1 SIA 380/1

Die Berechnung der SIA 380/1 erfolgt nach den entsprechenden Formeln der genannten Norm und auf der Grundlage von modifizierten Standardnutzungen (es sei hier auf das Anwenderhandbuch, Verweis „SIA 380/1“ verwiesen). Die Berechnung erfolgt entlang dem Pfad:

- Objekt
- Teilobjekt (unterschiedliche Standardnutzungen und Energieerzeugungsarten)
- Bauwerksteil
- Berechnungselement

Berechnet werden Qh, Qww und Ee für jedes einzelne Teilobjekt. Danach werden diese mit den Wirkungsgraden der Energieerzeugungsarten in Verbindung gebracht und anteilig auf das Objekt hochgerechet und dort als Eh, Eww und Ee abgelegt. Da die Energieerzeugungsarten mit Kostenwerten versehen sind, können die Betriebskosten für ein Jahr berechnet werden. Weitere Jahre werden als Wiederholung des erstes Jahres betrachtet. Qh, Qww und Ee eines jeden einzelnen Objekts werden aber auch mit den bewerteten Sachbilanzen der Energieerzeugungsarten auf das Objekt aufsummiert. Der Wirkungsgrad wird hierbei nicht herangezogen, da er bereits in den bewerteten Sachbilanzen der Energieerzeugungsarten enthalten ist.

6.2 Energie und Stoffflüsse

Die Berechnung der Energie- und Stoffflüsse erfolgt entlang des Pfades:

- Objekt
- Berechnungselement
- Normposition
- Energie- und Stoffffluss-Komponenten

Es wird also ein „Top-Down"-Verfahren angewendet. Vom Objekt her wird über die Menge der Berechnungselemente gegangen. Jedes Berechnungselement geht dann über die Normpositionen, die über den Detailaufbau mit ihm verknüpft sind. Jede Normposition geht über die Energie- und Stoffffluss-Komponenten, die mit ihm verknüpft sind und berechnet anhand der Angaben zu Inventar, Menge und Flussrichtung die Energie- und Stoffflüsse. Das Ergebnis wird dann rückwärts den Pfad entlang aufsummiert.

Die Berechnung erfolgt für folgende „Phasen":

- Erstellung
- Unterhalt, Erneuerung
- Rückbau

Der Betrieb wird in der Berechnung der SIA 380/1 abgedeckt.

6.2.1 Erstellung (Normposition und Energie- und Stoffffluss-Komponente)

Die Berechnung der Energie- und Stoffflüsse der Erstellung folgt einem einfachen Muster. Zuerst wird eine Typprüfung in der folgenden Reihenfolge durchgeführt.

- Maschine
- Inventar
- Konstruktionsprodukt

Handelt es sich um eine Maschine, dann wird ermittelt, ob eine Laufzeit oder eine Vorhaltezeit vorliegt. Liegt eine Vorhaltezeit vor, dann wird anhand der aus der SBIL gewonnenen Daten die rechnerische Laufzeit ermittelt. Es liegt nun eine Laufzeit vor. Diese Laufzeit muss nun noch in die entsprechende Einheit der bewerteten Sachbilanz umgerechnet werden. Danach kann der Eintrag berechnet werden: jedes Einzelkriterium mal ermittelter Menge.

Handelt es sich um ein Inventardatum, dann wird die Menge berechnet, so dass diese in der Einheit der bewerteten Sachbilanz der jeweiligen Energie- und Stoffflusskomponente vorliegt (zB. „PE-Folie" in kg). Aus diesem Grund besitzen Energie- und Stoffflusskomponenten Angaben zur Rohdichte, zur Masse pro gegebener Fläche oder auch zur Masse pro Stück. Danach wird über die Angabe der Vorhaltezeit oder der Verwendbarkeit der Anteil errechnet, der in die Berechnung einfliesst. Bei einer Verwendbarkeit von „1" ist der Anteil „1". Bei einer Verwendbarkeit von „2" ist der Anteil „0.5". Die Vorhaltezeit wird in Bezug gebracht mit der Lebensdauer der Energie- und Stoffffluss-Komponente. Diese ist im Augenblick bei allen Komponenten gleich und wurde auf „7" Jahren festgesetzt. Aus der Relation von Vorhaltezeit und Lebensdauer

kann wiederum ein Anteil errechnet werden: „2“ Monate aus „7“ Jahren wäre ein Anteil von „0.024..“. Bei Inventardaten wird nun sowohl die bewerte Sachbilanz für die Erstellung wie auch der Entsorgung in die Berechnung des Energie- und Stoffflusses gemäss Menge und Anteil einbezogen.
Handelt es sich um ein Konstruktionsprodukt, so wird die Menge auf die Einheit der bewerteten Sachbilanz der Energie- und Stoffffluss-Komponente umgerechnet und dann berechnet. Wurde ein Verschnitt angegeben, so fliesst dieser Verschnitt als Entsorgung in die Bilanz ein. Verschnitte oder Abbrüche sind durch die Flussrichtung „raus“ gekennzeichnet. In diesem Falle wird die bewertete Sachbilanz der Entsorgung der Energie- und Stoffffluss-Komponente herangezogen.

6.2.2 Rückbau (Normposition und Energie- Stoffffluss-Komponente)
Der Abriss kann direkt bei der Berechnung der Erstellung ermittelt werden. Er ist gleichzusetzen mit dem Punkt „Konstruktionsprodukt“ der Berechnung der Erstellung, so das Konstruktionsprodukt kein Verschnitt oder Abbruch ist. Beim der Berechnung des Rückbaus wird jedoch immer die bewertete Sachbilanz der Entsorgung der Energie- und Stoffffluss-Komponenten herangezogen. Alle Maschinen, die für einen Rückbau notwendig sind, fliessen im Augenblick nicht in die Bilanz ein, da der Rückbau nur eine rechnerische Variante der Erstellung ist.

6.2.3 Unterhalt, Erneuerung
Unterhalt oder Erneuerung sind wie Rückbau eine einfache Projektion auf der Grundlage der Lebensdauern der Berechnungselemente und der Berechnungen der Erstellung. Iterativ werden die Stoffflüsse für alle vielfache der Lebensdauer innerhalb der Gesamtlebensdauer des Bauwerks berechnet, wobei „Arbeiten“, die in die letzen zehn Jahre fallen nicht mehr betrachtet werden. Das Problem, dass Normpositionen keine „Lebensdauer“ besitzen, sondern nur Berechnungselemente wurde gelöst, indem bei der Berechnung das Berechnungselement seine Lebensdauer an die Normposition weitergibt und diese für den Moment der Berechnung somit eine Lebensdauer besitzt. Jeder Einzeleintrag entspricht rechnerisch der Summe der Berechnungen aus Erneuerung und Abriss, da von der einfachen Annahme ausgegangen wird, dass das Teil zuerst entfernt und dann wieder eingesetzt wird.

6.2.4 Normposition zu Berechnungselement
Die berechneten Energie- und Stoffflüsse einer Normposition fliessen gemäss ihres Faktors aus dem Detailaufbau eines Berechnungselements in dieses ein.

6.2.5 Berechnungselement zu Objekt
Die berechneten Energie- und Stoffflüsse der Berechnungselemente können nun auf der Ebene Objekt gemäss ihrer Einzelmengen zusammengefasst werden. Bei der Berechnung des Objekts müssen nun die Lebensdauern der Berechnungselemente betrachtet werden. Eine Lebensdauer gleich „0“ bedeutet implizit, dass das Berechnungselement die Lebensdauer des Objekts hat. Es „verursacht“ somit keine weiteren Stoffflüsse in Erneuerung / Unterhalt. Hat ein Berechnungselement eine Lebensdauer ungleich „0“, so muss der in Erneuerung / Unterhalt berechnete Datensatz iterativ über die Lebensdauer des Objekts immer wieder eingesetzt werden. Hätte ein Berechnungselement die Lebensdauer „12“, so würde also im Jahre „12“ nach Erstellung, „24“, „36“ und so fort der Stoffffluss immer wieder anfallen. In OGIP wird mit einer Lebensdauer von „80“ Jahren gerechnet. Stoffflüsse, die in die letzten „10“ Jahre fallen werden abgeschnitten (nicht betrachtet). In unserem Falle würde also der Stoffffluss im Jahre „72“ nicht mehr betrachtet werden. Zum Schluss werden dann alle Stoffflüsse zwischen Erstellung und Ende der Lebensdauer des Objekts aufsummiert. Es liegt der Gesamtstoffffluss aus Erneuerung / Unterhalt vor. Mit dem Betrieb kann genauso verfahren werden.

6.2.6 Zusammenfassung
In OGIP werden somit vier unterschiedliche Ergebnisdatensätze geführt:

- Erstellung

- Erneuerung / Unterhalt
- Rückbau
- Betrieb

In der Auswertung werden sie entsprechend der EKG (Elementkostengliederung) zusammengefasst. Problematisch ist hier die Berechnung „Betrieb“, da die EKG keine Elementgruppen für diesen Bereich vorsieht. Die Berechnungen von Erstellung, Erneuerung / Unterhalt und Rückbau werden zusammengelegt.

7 Rohdaten (Eingangsdaten zum Anwenderprogramm)

Als Grundlage des Anwenderprogramms dienen die ESD-Datensätze (Rohdaten). Sie sind anlalog zu den sonstigen CRB-Rohdaten aufgebaut. Besprochen werden hier nur die Datensätze bezüglich der Energie- und Stoffflussbilanzierung des BEK und des NPK. Auf die Datensätze der SIA 380/1 wird nicht eingegangen, da diese aus den Normen entnommen werden können.
Ebenfalls nicht beschrieben sind Objektinfo, BEK / NPK und der Detailaufbau. Diese Datensätze und der konforme Umgang mit ihnen ist in den diversen EDV-Anleitungen des CRB beschrieben (Auskunft über die CRB-Geschäftsstelle in Zürich).

7.1 ESD2510

ESD2510 ist eine Art von Ressourcendatei. Alle Texte sind in dieser Datei zusammengefasst. Eine Übersetzung wird somit erleichtert.

Feldnr.	Inhalt	Länge	Beispiel
1	Version	4	1998
2	Sprachcode	1	1
3	CodeNr. Begriff	6	000033
4	Länge Text	4	0022
5	Begriff	variabel	Umweltbelastungspunkte

7.1.1 Beispiel

```
199810000340001m
199810000350002m2
199810000980010Erstellung
199810000990004k.a.
```

7.1.2 Hinweis

Die *Code-Nr. "000000"* hat keinen zugewiesenen Texteintrag. Diese *Code-Nr*. ist gleichzusetzen mit NULL (Nil, null), so sie in anderen ESD-Rohdaten referenziert wird.

7.2 ESD2520

Kriterien der Berechnungselemente. Die Berechnungselemente eines Elements (zB. "A1") sind jeweils in einer Datei zusammengefasst. Die Dateien sind durch die Element-ID als Dateierweiterung voneinander diskriminiert (zB. "ESD2520.A1").

Feldnr.	Inhalt	Länge	Beispiel
1	Version	4	1998
2	Kapitel	2	A1
3	HPOS	3	111
4	UPOS	3	101
5	Zeilennummer	3	001
6	Bez. Phase	6	000098
7	Lebensdauer	3	000
8	Bez. Kriterium	6	000033
9	Bez. Einheit	6	000099
10	Wert	18	Expotentialdarstellung

7.2.1 Beispiel

```
1998A2111101001000098000000330000993.333350000000e+01
```

7.2.2 Hinweis

Lebensdauer "000" bedeutet, dass die Lebensdauer des Berechnungselement die Lebensdauer des Bauwerks ist. Die Begriffe zu *Bez. Phase*, *Kriterium* und *Einheit* sind aus ESD2510 zu entnehmen. ESD2520 kann vollständig über den Detailaufbau aus ESD2530 berechnet werden. Es wäre somit weder notwendig, ESD2520 persistent abzulegen, noch ESD2520 einzulesen.
Für *Phase* können folgende Zuweisungen existieren:

- *Erstellung*
- *Erneuerung || Unterhalt*
- *Abriss || Rueckbau*
- *Betrieb*

Es ist daran zu denken, dass die Datenstruktur im Falle *Erneuerung || Umbau* die *Lebensdauer* ebenfalls verwalten muss. Im Augenblick wird *Betrieb* nicht genutzt. Diese Belegung ist vorgesehen für Reinigungs-, Wartungsarbeiten an einem Element.

7.3 ESD2530

Kriterien der Normpositionen. Die Normpositionen eines Kapitels (zB. "113") sind jeweils in einer Datei zusammengefasst. Die Dateien sind durch die Kapitel-ID als Dateierweiterung voneinander diskriminiert (zB. "ESD2530.113").

Feldnr.	Inhalt	Länge	Beispiel
1	Version	4	1998
2	Kapitel	2	333
3	HPOS	3	111
4	UPOS	3	101
5	Zeilennummer	3	001
6	Bez. Phase	6	000098
7	Lebensdauer	3	000
8	Bez. Kriterium	6	000033
9	Bez. Einheit	6	000099
10	Wert	18	Expotentialdarstellung

7.3.1 Beispiel

```
19984331111010010000980000000330000993.333000000000e+01
```

7.3.2 Hinweis

Lebensdauer *"000"* bedeutet, dass die Lebensdauer der Normposition die Lebensdauer des jeweiligen Berechnungselements ist, von welchem sie gerade aus betrachtet wird. Die Begriffe zu *Bez. Phase*, *Kriterium* und *Einheit* sind aus ESD2510 zu entnehmen. Für *Phase* gelten die gleichen Belegungen wie in ESD2520. ESD2530 kann vollständig aus ESD2540 berechnet werden. Es ist somit weder notwendig, ESD2530 persistend abzulegen noch ESD2530 einzulesen.

7.4 ESD2540

e+s-Inventar der Normpositionen. Da ESD2540 auf ESD2550 zurückgreift, sollte ESD2550 vor ESD2540 eingelesen werden.

Feldnr.	Inhalt	Länge	Beispiel
1	Version	4	1998
2	Kapitel	3	333
3	HPOS	3	111
4	UPOS	3	101
5	Zeilennummer	3	001
6	Bez. Phase	6	000098

7	Lebensdauer	3	000
8	Flussrichtung	1	0 || 1
9	Art	2	01
10	Bez. e+s-Komp.	6	000331
11	Bez. Einheit	6	000034
12	Menge	18	Expotentialdarstellung

7.4.1 Beispiel

```
199843311110100100009800010100033100003400.123300000000e+01
```

7.4.2 Hinweis

Lebensdauer *"000"* bedeutet, dass die Lebensdauer der Normposition (oder eben des Berechnungselements) gilt. Die Begriffe zu *Bez. Phase*, *e+s-Komp.*, *Einheit* werden aus ESD2510 entnommen. Vor Betrachtung der *Flussrichtung* muss *Art* betrachtet werden. *Bez. Phase* hat keinerlei Bedeutung auf dieser Ebene.

7.4.2.1 Für *Art* gelten folgende Zuweisungen

- *1* == Maschine
- *2* == Transport
- *3* == Produkt
- *4* == Inventar
- *5* == Verschnitt
- *6* == Abriss
- *7* == Abfall
- *8* == Energie
- *9* == Kosten

Produkt, Verschnitt, Abriss und Abfall gehen nach Konstruktionsprodukt (im Sinne von Kapitel „Berechnung der Energie- und Stoffflüsse“). Maschine und Transport nach Maschine (im Sinne von Kapitel „Berechnung der Energie- und Stoffflüsse“). Energie sind Energieerzeugungsarten der SIA 380/1 oder Betriebsmittel von Maschinen (Diesel für Baumaschinen, etc.). Kosten ist ein bisher nicht belegter Reservebegriff.

7.4.2.2 Für *Flussrichtung* gelten folgende Zuweisungen

- *1* == hinaus
  Die genannte e+s-Komponente wird entsorgt. Es handelt sich also um einen Abbruch.
- *0* == hinein
  Die genannte e+s-Komponente wird eingebracht. Es handelt sich also um das Einbringen von Material.

7.4.2.3 Kombinatorik von *Art* und *Flussrichtung*

Wenn *Art* gleich *Maschine*, *Transport* oder *Energie* ist, dann ist *Flussrichtung* von keiner Bedeutung. Wenn *Art* gleich *Inventar* ist kann die *Flussrichtung* ebenfalls ignoriert werden. *Art Produkt*, *Abfall*, *Verschnitt* und *Abriss* sind gleich zu behandeln.

7.4.2.4 Algorithmischer Ablauf

- Ist *Art* gleich *Maschine*, *Transport* oder *Energie*
  Initialisiere Maschine, Transport oder Energie indem *Bez. e+s-Komp.* aufgelöst wird, *Bez. Einheit* als Einheit und *Menge* als Menge gesetzt wird.
- Ist *Art* gleich *Inventar*
  Initialisiere Inventar indem *Bez. e+s-Komp.* aufgelöst wird und setze *Bez. Einheit* als Einheit und *Menge* als Menge.
- Ist Art gleich Produkt, Abfall, Verschnitt oder Abriss
  Initialisiere Produkt indem *Bez. e+s-Komp.* aufgelöst wird und setze *Bez. Einheit* als Einheit, *Menge* als Menge und *Flussrichtung* als Flussrichtung.

Durch die Auflösung von *Bez. e+s-Komp.* (s.a. ESD2550) ist der Zugriff auf die bewerteten Sachbilanzen sichergestellt. Bei Produkten liegen diese sowohl als Entsor-

gung wie auch als Herstellung vor. Bei Maschinen, Energie oder Transporten liegen nur Herstellung vor. Aus diesem Grund ist *Flussrichtung* auch nur bei Produkten von Interesse.

7.5 ESD2541

Auflistung der "nicht relevanten" Normpositionen. Positionen, die keinen Energie- und Stoffffluss aufweisen.

Feldnr.	Inhalt	Laenge	Beispiel
1	Jahrgang	4	1998
2	kapitel	3	311
3	hpos	3	123
4	upos	3	456

7.5.1 Beispiel

```
1998311111101
```

7.5.2 Hinweis

Nicht relevante Positionen dürfen bei der Berechnung aber nicht vollständig übergangen werden. Ihr Kostenrichtwert muss auf jeden Fall aggregiert werden. Elegant kann dieses Probem dadurch gelöst werden, dass bei Normpositionen auf jeden Fall eine bewertete Sachbilanz erzeugt wird und diese ein Kriterium „Kostenrichtwert“ erhält. Nicht relevante oder nicht transformierte Normpositionen lassen sich so genauso behandeln wie transformierte Normpositionen (Positionen, die Einträge nach ESD2540 haben).

7.6 ESD2550

Die e+s-Komponenten, aus welchen die e+s-Inventare der Normpositionen zusammengesetzt sind.

Feldnr.	Inhalt	Länge	Beispiel
1	Jahrgang	4	1998
2	Zeilennummer	3	001
3	Bez. Komponente	6	000313
4	code Emis 2000	6	000114
5	Art	2	01
6	hst/ents/attr	1	0
7	Bez. Eigenschaft	6	000631
8	Bez. Einheit	6	000034
9	Menge	18	Expotentialdarstellung

7.6.1 Beispiel

```
19980010003130002010110006310000340.12330000000e+01
```

7.6.2 Hinweis

Code Emis 2000 wird einfach uebernommen. Die Begriffe für *Bez. Komponente*, *Eigenschaft*, *Einheit* sind aus ESD2510 zu entnehmen.

7.6.2.1 Für Art gelten folgende Zuweisungen

- *1* == Maschine
- *2* == Transport
- *3* == Produkt
- *4* == Inventar
- *5* == Verschnitt
- *6* == Abriss
- *7* == Abfall
- *8* == Energie
- *9* == Kosten

7.6.2.2 Zuweisungen hest/ents/attr

Für die Gruppe hest/ents/attr gelten folgende Zuweisungen

- *1* == Herstellung
  Es folgt ein Kriteriums-Wert-Paar der bewerteten Sachbilanz zur Herstellung (Maschinen und Energie und besitzen nur eine Datensatz in Herstellung).
- *2* == Entsorgung
  Es folgt ein Kriteriums-Wert-Paar der bewerteten Sachbilanz zur Entsorgung.
- *0* == Attribut
  Beschreibung zu einem Attribut.

7.6.2.3 Zusammenhang mit *hest/ents/attr*

In Zusammenhang mit *hest/ents/attr* == *0* ("Attribut") können folgende Fälle auftauchen, nachdem über die ESD2510 der begriff geholt wurde:

- *flmasse* == Masse bezogen auf eine Flaeche (explizite Einheit)
- *rohdichte* == Rohdichte (explizite Einheit)
- *deponierung* == Deponierung (keine Einheit)
- *verwendbarkeit* == Verwendbarkeit (zB. bei Folien); implizit in "mal"
- *abschr* == Abschreibungszeit, Lebensdauer (implizit in "Jahren"). In OGIP wird eine vordefinierte Abschreibungszeit von 7.0 Jahren benutzt.
- *bzgmasse* == Masse (explizite Einheit) in Relation zu "bzgmenge"
- *bzgmenge* == Menge (explizite Einheit) in Relation mit "bzgmassen"

Hier liegt eines der Hauptprobleme der ESD-Daten. Programminterne Steuergrössen werden über die gleiche Datei vermittelt wie sprachsensitive Begriffe. Dies ist eine potentielle Fehlerquelle und muss verbessert werden.

7.6.2.4 Datensätze für Öko-Kriterien

Die Datensätze der Öko-Kriterien (beschrieben unter *hest/ents/attr* == 1 || 2 beziehen sich auf diese beiden Attribute.

- *bzgmasse* == 1 kg
- *bzgmenge* == 1000 St

Datensatz zu Entsorgung oder Herstellung bezieht sich auf 1 kg pro 1000 St. Wenn in einer Komponente keine bzgmasse angegeben ist, dann bezieht sich der Datensatz allein auf bzgmenge (zB. Beton mit *bzgmenge* == „1“ „kg“).

7.6.2.5 Weitere Attribute

- *vt* == Verrechnungstage (explizite Einheit)
- *std* == Einsatzstunden (explizite Einheit) in den Verrechnungstagen
- *wirkungsgrad* == Wirkungsgrad bei Energieerzeugern (zB. Gas-Brennwert)

Mittels *vt* und *std* lassen sich Angaben des Anwenders in Form der Vorhaltedauer realisieren, da sich mit der Aussage der Verrechnungstage und der Einsatzstunden in diesen Verrechnungstagen über einen Dreisatz die Einsatzstunden einer beliebigen Vorhaltezeit berechnen lassen.

7.6.2.6 Algorithmischer Ablauf

Falls man die Typen, *Energie*, *Maschine*, *Produkt* als eigene Klasse ablegt ist es das beste, wenn man eine *Factory* anlegt, die die Instanzen entsprechend erzeugt. Falls man einen Datentyp benutzen will, dann sollte ein *Flag* für die Typunterscheidung eingeführt werden. Hier wird davon ausgegangen, dass Typen existieren und eine *Factory* benutzt wird und dass die die Typen eine gemeinsame Oberklasse haben. Diese soll *Komponente* genannt werden.

Instanziieren eines Objekts:

- Ist *Art* gleich Produkt, dann wird *Produkt* instanziiert.

- Ist *Art* gleich Transport, dann wird *Maschine* instanziiert.
- Ist *Art* gleich Maschine, dann wird *Maschine* instanziiert.
- Ist *Art* gleich Energie, dann wird Energie instanziiert.

Andere Fälle existieren im Augenblick nicht in ESD2550. Es folgt die Abarbeitung der Attribute (oder Eigenschaften).
Die gemeinsame Superklasse *Komponente* arbeitet folgende Attribute ab:

- *0; bzgmenge* == Bezugsmenge der Komponente auf die sich die bewerte Sachbilanz der Herstellung oder der Entsorgung bezieht (zB. „1“ „St“).
- *1; Herstellung* == Es folgt ein Einzelwert der bewerteten Sachbilanz, welche die Herstellung abbildet. Alle Einzelwerte dieser bewerteten Sachbilanz werden sinnvollerweise in einer zusammenhängenden Datenstruktur abgelegt.

Die Klasse *Produkt* arbeitet folgende Attribute ab. Alle anderen Attribute werden an die Superklasse weitergereicht (zB. mit explizitem Aufruf der Methode der Superklasse).

- *2; Entsorgung* == Es folgt ein Einzelwert der bewerteten Sachbilanz, welche die Entsorgung abbildet (s.a. oben).
- *0; bzgmasse* == Masse der Komponente pro Bezugsmenge.
- *0; rohdichte* == Rohdichte der Komponente.
- *0; flmasse* == Masse pro „1“ „m2“ (der Bezug ist implizit).
- *0; deponierung* == Dies ist ein Sonderfall! Der einzutragende Wert steht unter *Bez. Einheit*. *Wert* ist völlig uninteressant (s.a. „Anhang“, Listing „esProdukt“, Methode „addAttribute“).
- *0; verwendbarkeit* == Verwendbarkeit (bereits diskutiert unter „Berechnung der Energie- und Stoffflüsse“).
- *0; abschr* == Abschreibungszeit (bereits diskutiert unter „Berechnung der Energie- und Stoffflüsse“).

Die Klasse *Energie* arbeitet folgende Attribute ab:

- *0; wirkungsgrad* == Wirkungsgrad

Die Klasse *Maschine* arbeitet folgende Attribute ab:

- *0; vt* == Verrechnungstage.
- *0; std* == Einsatzstunden.

Der Zusammenhang zwischen Verrechnungstagen und Einsatzstunden kann aus der SBIL entnommen werden.

7.7 ESD2560

Attribute der Berechnungselemente.

Feldnr.	Inhalt	Laenge	Beispiel
1	Jahrgang	4	1998
2	Kapitel	2	A1
3	HPos	3	141
4	UPos	3	101
5	Bez. Attr	6	000098
6	Bez. Einheit	6	000101
7	typ	2	01
8	value	variabel	100

7.7.1 Beispiel

```
1998A1141101000098000101101100
```

7.7.2 Hinweis

Bez. Attr, *Einheit* sind aus ESD2510 zu entnehmen.

7.7.2.1 Für Typ gelten folgende Zuweisungen:

- *01* == Ganzzahl (Integer)
- *02* == Gleitkommazahl (Double)
- *03* == Text (String)

7.7.2.2 Folgende Attribute sind im Augenblick in ESD2560 beschrieben:

- *ldauer* == Lebensdauer (explizite Einheit)
- *kWert* == k-Wert oder U-Wert (explizite Einheit)
- *gWert* == g-Wert (keine Einheit)
- *glasanteil* == Glasanteil (explizite Einheit)
- *wdw_r* == Wärmedurchgangswiderstand einer homogenen Konstruktion (explizite Einheit)
- *wdw_ro* == Wärmedurchgangswiderstände oben einer inhomogenen Konstruktion (explizite Einheit)
  wdw:ro stellt einen Sonderfall dar, insoweit als die einzelnen Durchgangswiderstände das Postfix "1", "2", ... bekommen haben (zB. "wdw_ro1"). Besser wäre es natürlich gewesen, es wäre ein Typ Collection of Double eingeführt worden.
- *wdw_ru* == Wärmedurchgangswiderstand unten, inhomogene Konstruktion (explizite Einheit)

wdw_r kann nicht in Kombination mit *wdw_ro** oder *wdw_ru* auftreten.

7.8 ESD2570

Sortierung der e+s-Komponenten nach Kategorien.

Feldnr.	Inhalt	Laenge	Beispiel
1	Jahrgang	4	1998
2	BezKategorie	6	000011
3	BezE+S-Komp	6	000022

7.8.1 Beispiel

```
1998000011000022
```

7.8.2 Hinweis

BezKategorie und *BezE+S-Komp* sind aus ESD2510 zu entnehmen. ESD2570 wurde erst zum Schluss in die ESD-Datensätzen aufgenommen. Im Grunde genommen hätte die Kategorie oder die Kategorien auch eine Eigenschaft oder Eigenschaften in ESD2550 sein können. Es ist vorgesehen, dass eine Energie- und Stoffflusskomponente zu mehreren Kategorien gehören kann. Die Datenstrukturen sind entsprechend vorzusehen.

8 Struktur des Anwenderprogramms OGIP

Für eine detaillierte Darstellung sei hier auf das Hanbuch von OGIP, „Struktur von OGIP“ verwiesen. OGIP unterteilt sich in zwei Hauptgruppen:

- Kataloge
- Objekt

Die Kataloge unterteilen sich in die Gruppen:

- Bauwerksteile und Bauwerksteil bearbeiten
- Berechnungselemente und ein freies Berechnungselement definieren
- Normpositionen und eine freie Normposition definieren

Das Modul Objekt untergliedert sich in:

- Objektinformation erfassen mit Objekt öffnen, neu anlegen und kopieren
- Grundmengen zum Objekt erfassen (zB. Nettogeschossfläche)

- Objektgliederung erfassen (wichtig für die SIA 380/1)
- Ausführung beschreiben (mittels Berechnungselementen, Bauwerksteilen oder Hauptpositionen)
- Auswertung durch die Gegenüberstellung eines Objekts mit Referenzobjekten

Eine detaillierte Darstellung der Einzelfunktionen der Module findet sich im OGIP Anwenderhandbuch.

9 Weiterführung der Projektergebnisse und Umsetzung in die Praxis

9.1 *β-Version*

Zeitraum: 2000 bis 2001
Für Interessierte wird ein spezielles β-User-Dienstsleistungspaket angeboten. Dieses umfasst:
>Anwenderprogramm
>Schulung (2x ½ Tag)
>Support (Betreuung in der Praxis durch Kursleiter)
>Erfa-Austausch
Die Kosten für die einzelnen Teilnehmer für das gesamte Paket sind mit SFr. 420.— kalkulliert.
Die Schulung ist als Vertiefungsmodul zu den Kursen im Bereich Elementmethode konzipiert. "Einführung in das Anwenderprogramm OGIP" ist das erste Teilmodul, "Grundlagen zu OGIP und Ökobilanzierung – Möglichkeiten der praktischen Anwendung" das zweite. CRB und ZEN führen die Kurse ausschliesslich gemeinsam durch. Alle Kurse werden im ersten Quartal 2000 durchgeführt.

9.2 Vollversion

Zeitraum: ab 2001
In der Vollversion werden die Erfahrungen aus der β-Phase bereits berücksichtigt werden. Die Herausgabe des Programmes und der Vertrieb wird zu 100% durch das CRB wahrgenommen.
Die Ausgestaltung des Angebotes wird analog dem aus der β-Phase erfolgen. Ausgehend von den Erfahrungen aus der β-Phase soll jedoch ein breiter Kreis von möglichen Anwendern angesprochen werden.
Über die ganze Zeitspanne 2000 bis 2001 werden die Entscheidungsträger (Bauherren und Planer) zusätzlich mit regelmässigen Presseinformationen zu OGIP sensibilisiert. Ausserdem besteht für alle zugänglich eine Informationsquelle unter www.ogip.ch.

10 Literatur

SBIL	Schweizerische Bauinventar Liste
OGIP/DATO	N. Kohler et al: OGIP/DATO Optimierung von Gesamtenergieverbrauch, Umweltbelastung und Baukosten, Schlussbericht April 1996, Bezug EMPA Dübendorf
SIA180	Norm: SIA 180, Ausgabe 1988, Wärmeschutz im Hochbau, Schweizerischer Ingenieur- und Architekten-Verein
SIA 1801	Empfehlung: SIA 180/1, Ausgabe 1988, Nachweis des mittleren k-Wertes der Gebäudehülle, Schweizerischer Ingenieur- und Architekten-Verein
SIA1804	Empfehlung: SIA 180/4, Ausgabe 1982, Energiekennzahl, Schweizerischer Ingenieur- und Architekten-Verein
SIA3801	Empfehlung: SIA 380/1, Ausgabe 1988, Energie im Hochbau, Schweizerischer Ingenieur- und Architekten-Verein
SIA3811	Empfehlung: SIA 381/1, Ausgabe 1980, Baustoff-Kennwerte, Schweizerischer Ingenieur- und Architekten-Verein
SIA3812	Empfehlung: SIA 381/2, Ausgabe 1988, Klimadaten zu Empfehlung 380/1 "Energie im Hochbau", Schweizerischer Ingenieur- und Architekten-Verein
SIA3813	Empfehlung: SIA 381/3, Ausgabe 1982, Heizgradtage der Schweiz, Schweizerischer Ingenieur- und Architekten-Verein
HANDBUCH	Anwenderhandbuch von OGIP, HTML-Dokument, CRB, Zürich

Weitere Literaturhinweise finden sich unter http://www.ogip.ch, Verweis Beteiligte, Verweis Literatur.

11 Anhang

Es sind lauffähige Beispiele, die einen möglichen Umgang mit den esd-Rohdaten deutlich machen sollen. Ausschneiden und der java-Konvention auf jeweils eine Datei verteilen. Alle Dateien in einem Verzeichnis ablegen. "javac *.java" eingeben. Danach können esd2510, esd2520, esd2540, esd2550 und esd2560 mit zB. "java esd2510" aufgerufen werden.

11.1 Listing zu esd2510

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

/**
 * bezug herstellen zwischen einer codierung und einem menschlich
 * verstaendlichen text. das exception modell ist natuerlich sehr
 * einfach gehalten. in 'echt' wuerde es natuerlich etwas ausgefeilter
 * sein.
 * esd2510 wuerde man zB. als singleton implementierten
 * esd2510 nutzt esd2510Paar
 */
class esd2510 extends esdOgip {

  // hin und her (man weiss ja nie)
  // mehrsprachig in einer anwendung ginge so natuerlich nicht
  private HashMap codeZuBegriff = new HashMap ();
  private HashMap begriffZuCode = new HashMap ();
  public boolean isBegriff (String begriff) {
    return begriffZuCode.containsKey (begriff);
  }
  public boolean isCode (String code) {
    return codeZuBegriff.containsKey (code);
  }
  public String getBegriff (String code)
      throws esd2510Exception {
    if (codeZuBegriff.containsKey (code))
      return ((esd2510Paar)codeZuBegriff.get(code)).getBegriff();
    // wird eine ausnahme (exception)
    throw new esd2510Exception ();
  }
  public esd2510Paar getBegriffsobjekt (String code)
      throws esd2510Exception {
    if (codeZuBegriff.containsKey (code))
      // in der ganzen anwendung wuerde man sinnvollerweise nur dieses
      // objekt plazieren
      return (esd2510Paar)codeZuBegriff.get(code);
    throw new esd2510Exception ();
  }
  public String getCode (String begriff)
      throws esd2510Exception {
    if (begriffZuCode.containsKey (begriff))
      return ((esd2510Paar)begriffZuCode.get(begriff)).getCode();
    // wird eine ausnahme (exception)
    throw new esd2510Exception ();
  }
  public void setBegriff (String code, String begriff)
      throws esd2510Exception {
    if (isCode(code)) {
      String oldVal = getBegriff (code);
      if (oldVal.equals (begriff))
        return;
```

```
      // wirf eine ausnahme (exception)
      throw new esd2510Exception ();
    }
    if (isBegriff(begriff)) {
      String oldVal = getCode (begriff);
      if (oldVal.equals (code))
        return;
      // wirf eine ausnahme (exception)
      throw new esd2510Exception ();
    }
    esd2510Paar paar = new esd2510Paar (code, begriff);
    codeZuBegriff.put (code, paar);
    begriffZuCode.put (begriff, paar);
  }
  /**
   * nichts zu tun (alles ist initialisiert; dank sei JAVA)
   */
  public esd2510 () {

  }

  public static void main (String[] args) {
    /**
     * man wuerde natuerlich einen FileReader instanziieren und
     * einen tokenizer schreiben, der die datei esd2510 entsprechend
     * aufbereitet. hier wurde darauf verzichtet.
     * zB.
     * esdTokenizer als baseclass
     * esd2510Tokenizer als subclass
     */

    esd2510 esd = new esd2510 ();

    try {
      String key = new String ("01");
      String value = new String ("hello");

      // ein bezug herstellen
      System.out.println ("(1) " + key + "+" + value);
      esd.setBegriff (key, value);
      // den begriff auslesen
      System.out.println (key + "=" + esd.getBegriff (key));
      // das objekt auslesen (toString von esd2510Paar gibt begriff
      // zurueck)
      System.out.println (key + "=" + esd.getBegriffsobjekt (key));
      // den code auslesen
      System.out.println (value + "=" + esd.getCode (value));
      // zweiter versuch wird ignoriert
      System.out.println ("(2) " + key + "+" + value +
                          " (das wird ignoriert)");
      esd.setBegriff (key, value);
      // eine ausnahme provozieren
      value = new String ("Dies wird eine exception geben!");
      System.out.println ("(3) " + key  + "+" + value);
      esd.setBegriff (key, value);

    } catch (Exception ex) {
      ex.printStackTrace ();
    }
  }
}

/*
 * eof esd2510
 */
```

11.2 Listing zu esd2520 und esd2530

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esd2520 extends esdOgip {

  public void addKriterium (int release,
                            String id,
                            int phase,
                            int lebensdauer,
                            esd2510Paar kriterium,
                            esd2510Paar einheit,
                            double wert) {
    esdKriterien kriterien = getEsdKriterien (lebensdauer, release,
                                              id, phase);
    // ein eintrag unter pos mit wert und einheit erzeugen
    kriterien.setNewKriterium (kriterium, wert, einheit);
  }
  private esdKriterien getEsdKriterien (int lebensdauer, int release,
                                        String id, int phase) {
    // nur ein fake
    // vorstellbar, dass hier eine entsprechende query abgesetzt
    // wuerde oder dass ueber entsprechende Referenzen 'navigiert'
    // werden koennte
    return new esdKriterien (release, id, phase, lebensdauer);
  }

  public esd2520 () {

  }
  /**
   * main
   */
  public static void main (String[] args) {
    /**
     * man wuerde natuerlich einen FileReader instanziieren und
     * einen tokenizer schreiben, der die datei esd2510 entsprechend
     * aufbereitet. hier wurde darauf verzichtet.
     * zB.
     * esdTokenizer als baseclass
     * esd2520Tokenizer als subclass
     * ------------------------------------
     * in dieser art
     * int release = tokenizer.getRelease ();
     * String id = tokenizer.getPosId ();
     * int zeile = tokenizer.getZeilennummer ();
     * int phase = tokenizer.getPhase ();
     * int lebensdauer = tokenizer.getLebensdauer ();
     * esd2510Paar kriterium = tokenizer.getKriterium ();
     * esd2510Paar einheit = tokenizer.getEinheit ();
     * double wert = tokenizer.getWert ();
     *
     */

    try {

      // sinnvollerweise werden solche sprachneutralen defines
      // ueber eine util-klasse hergestellt
      int ERSTELLUNG = 1;

      int release = 1999; // that's crude
      String id = new String ("A1 121.101");
```

```
      int phase = ERSTELLUNG;
      int lebensdauer = 0;
      esd2510Paar kriterium = new esd2510Paar ("01", "UBP");
      esd2510Paar einheit = new esd2510Paar ("02", "Punkte");
      double wert = 0.1234;

      esd2520 esd = new esd2520 ();

      // dies muss iterativ geschehen bis die id wechselt
      // dann muss eine neue instanz von esd2520 angelegt werden
      System.out.println ("Ein Kriterium einfuegen = " +
                          release + " / " +
                          id + " / " +
                          phase + " / " +
                          lebensdauer + " / " +
                          kriterium + " / " + /* remember: toString */
                          einheit + " / " +
                          wert);
      esd.addKriterium (release,
                        id,
                        phase,
                        lebensdauer,
                        kriterium,
                        einheit,
                        wert);

    } catch (Exception ex) {
      ex.printStackTrace ();
    }
  }
}

/* eof
 *
 */
```

11.3 Listing zu esd2540

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esd2540 extends esdOgip {

  /**
   * ein lookup waere hier natuerlich das richtige
   * als beispiel ok
   * damit die esKomponente bereits vorhanden sind muss esd2550
   * vor esd2540 eingelesen werden
   */
  public esKomponente getEsKomponente (int art, esd2510Paar bezeichnung)
       throws esKomponenteException {
    switch (art) {
    case 1:
    case 2:
      // Maschine, Transport
      // >> Maschine
      return new esMaschine (art, bezeichnung);
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
```

```
    // Produkt, Inventar, Verschnitt, Abriss, Abfall
    // >> Produkt
    return new esProdukt (art, bezeichnung);
  case 8:
    // Energie
    // >> Energie
    return new esEnergie (art, bezeichnung);

  default:
  }
  // wenn eine esKomponente nicht in esd2550 beschrieben war,
  // dann gibt es diese exception
  throw new esKomponenteException ();
}

public esd2540 () {

}

/**
 * main
 */
public static void main (String[] args) {
  /**
   * man wuerde natuerlich einen FileReader instanziieren und
   * einen tokenizer schreiben, der die datei esd2510 entsprechend
   * aufbereitet. hier wurde darauf verzichtet.
   * zB.
   * esdTokenizer als baseclass
   * esd2540Tokenizer als subclass
   * ---------------------------------------
   * in dieser art
   * int release = tokenizer.getRelease ();
   * String id = tokenizer.getPosId ();
   * int zeile = tokenizer.getZeilennummer ();
   * int phase = tokenizer.getPhase ();
   * int flussrichtung = tokenizer.getFlussrichtung ();
   * int art = tokenizer.getArt ();
   * art ist eigentlich typ
   * esd2510Paar bezKomp = tokenizer.getBezeichnungKomponente ();
   * esd2510Paar bezEinheit = tokenizer.getEinheit ();
   * double wert = tokenizer.getWert ();
   * oder auch getMenge
   *
   */

  try {

    esd2540 esd = new esd2540 ();

    int ERSTELLUNG = 0;
    int REIN = 0;
    int MASCHINE = 1;

    int release = 1999;
    String id = new String ("333/94 111.111");
    // ausgabejahr wuerde der tokenizer am besten mitliefern
    // zeile kann man ignorieren (wenn die id wechselt kommt ein
    // neues inventar)
    int phase = ERSTELLUNG;
    int lebensdauer = 0;
    int flussrichtung = REIN;
    int art = MASCHINE;
    esd2510Paar bezKomp = new esd2510Paar ("03", "Bagger");
    esd2510Paar bezEinh = new esd2510Paar ("04", "min");
    double wert = 13.0;

    // ein lookup waere hier natuerlich besser
    npkPosition position = new npkPosition (release, id);
    // esd2541 wuerde abgearbeitet werden, indem die methode
```

```
        // position.setNichtRelevant ()
        // aufgerufen werden wuerde

        esInventar inventar = new esInventar (release, id);
        position.setInventar (inventar);
        esKomponente komponente = esd.getEsKomponente (art, bezKomp);

        System.out.println ("Eine Komponente = " +
                            id + " / " +
                            release + " / " +
                            phase + " / " +
                            lebensdauer + " / " +
                            flussrichtung + " / " +
                            art + " / " +
                            komponente + " / " + // komponente.toString
                            bezEinh + " / " +
                            wert );

        // die arbeit tun
        inventar.addKomponente ( phase,
                                 lebensdauer,
                                 flussrichtung,
                                 art,
                                 komponente,
                                 bezEinh,
                                 wert);

      } catch (Exception ex) {
        // wenn einem sonst nichts einfaellt ist das immer gut
        ex.printStackTrace ();
      }
    }
}

/* eof
 *
 */
```

11.4 Listing zu esd2550

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esd2550 {

  /**
   * ok, da esd2550 die komponenten tatsaechlich anlegt
   */
  public esKomponente newEsKomponente (int art, esd2510Paar bezeichnung)
       throws esKomponenteException {
    switch (art) {
    case 1:
    case 2:
      // Maschine, Transport
      // >> Maschine
      return new esMaschine (art, bezeichnung);
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
      // Produkt, Inventar, Verschnitt, Abriss, Abfall
```

```
    // >> Produkt
    return new esProdukt (art, bezeichnung);
  case 8:
    // Energie
    // >> Energie
    return new esEnergie (art, bezeichnung);

  default:
  }
  // wenn eine esKomponente nicht in esd2550 beschrieben war,
  // dann gibt es diese exception
  throw new esKomponenteException ();
}

public esd2550 () {

}

/**
 * main
 */
public static void main (String[] args) {
  /**
   * man wuerde natuerlich einen FileReader instanziieren und
   * einen tokenizer schreiben, der die datei esd2510 entsprechend
   * aufbereitet. hier wurde darauf verzichtet.
   * zB.
   * esdTokenizer als baseclass
   * esd2550Tokenizer als subclass
   * ---------------------------------------
   * in dieser art
   * int release = tokenizer.getRelease ();
   * int zeile = tokenizer.getZeilennummer ();
   * esd2510Paar bezKomp = tokenizer.getBezeichnungKomponente ();
   * String codeEmis = tokenizer.getCodeEmis ();
   * int art = tokenizer.getArt ();
   * int typ = tokenizer.getTyp ();
   * String bezAttr = tokenizer.getBezeichnungAttribute();
   * esd2510Paar bezEinheit = tokenizer.getEinheit ();
   * double wert = tokenizer.getWert ();
   * oder auch getMenge
   */

  try {

    esd2550 esd = new esd2550();

    int ERSTELLUNG = 0;
    int REIN = 0;
    int PRODUKT = 3;
    int HERSTELLUNG = 1;
    int ATTRIBUT = 0;

    int release = 1999;
    esd2510Paar bezKomp = new esd2510Paar ("05", "Beton");
    int art = PRODUKT;
    int typ = HERSTELLUNG;
    esd2510Paar bezAttr = new esd2510Paar ("01", "UBP");
    esd2510Paar bezEinh = new esd2510Paar ("02", "Punkte");
    double wert = 0.0123;

    // im gegensatz zu esd2540 (== getEsKomponente) wird hier
    // ein newEsKomponente
    esKomponente komponente = esd.newEsKomponente (art, bezKomp);

    komponente.addAttribute (release,
                             typ,
                             bezAttr,
                             bezEinh,
                             wert);
```

```
      // zweites Beispiel
      typ = ATTRIBUT;
      bezAttr = new esd2510Paar ("10", "rohdichte");
      bezEinh = new esd2510Paar ("11", "kg/m3");
      wert = 2400.0;

      // die komponente haben wir
      komponente.addAttribute (release,
                               typ,
                               bezAttr,
                               bezEinh,
                               wert);

    } catch (Exception ex) {
      // wenn einem sonst nichts einfaellt ist das immer gut
      ex.printStackTrace ();
    }
  }
}

/* eof
 *
 */
```

11.5 Listing zu ESD2560

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esd2550 {

  /**
   * ok, da esd2550 die komponenten tatsaechlich anlegt
   */
  public esKomponente newEsKomponente (int art, esd2510Paar bezeichnung)
       throws esKomponenteException {
    switch (art) {
    case 1:
    case 2:
      // Maschine, Transport
      // >> Maschine
      return new esMaschine (art, bezeichnung);
    case 3:
    case 4:
    case 5:
    case 6:
    case 7:
      // Produkt, Inventar, Verschnitt, Abriss, Abfall
      // >> Produkt
      return new esProdukt (art, bezeichnung);
    case 8:
      // Energie
      // >> Energie
      return new esEnergie (art, bezeichnung);

    default:
    }
    // wenn eine esKomponente nicht in esd2550 beschrieben war,
    // dann gibt es diese exception
    throw new esKomponenteException ();
  }
```

```
public esd2550 () {

}

/**
 * main
 */
public static void main (String[] args) {
  /**
   * man wuerde natuerlich einen FileReader instanziieren und
   * einen tokenizer schreiben, der die datei esd2510 entsprechend
   * aufbereitet. hier wurde darauf verzichtet.
   * zB.
   * esdTokenizer als baseclass
   * esd2550Tokenizer als subclass
   * ---------------------------------------
   * in dieser art
   * int release = tokenizer.getRelease ();
   * int zeile = tokenizer.getZeilennummer ();
   * esd2510Paar bezKomp = tokenizer.getBezeichnungKomponente ();
   * String codeEmis = tokenizer.getCodeEmis ();
   * int art = tokenizer.getArt ();
   * int typ = tokenizer.getTyp ();
   * String bezAttr = tokenizer.getBezeichnungAttribute();
   * esd2510Paar bezEinheit = tokenizer.getEinheit ();
   * double wert = tokenizer.getWert ();
   * oder auch getMenge
   */

  try {

    esd2550 esd = new esd2550();

    int ERSTELLUNG = 0;
    int REIN = 0;
    int PRODUKT = 3;
    int HERSTELLUNG = 1;
    int ATTRIBUT = 0;

    int release = 1999;
    esd2510Paar bezKomp = new esd2510Paar ("05", "Beton");
    int art = PRODUKT;
    int typ = HERSTELLUNG;
    esd2510Paar bezAttr = new esd2510Paar ("01", "UBP");
    esd2510Paar bezEinh = new esd2510Paar ("02", "Punkte");
    double wert = 0.0123;

    // im gegensatz zu esd2540 (== getEsKomponente) wird hier
    // ein newEsKomponente
    esKomponente komponente = esd.newEsKomponente (art, bezKomp);

    komponente.addAttribute (release,
                             typ,
                             bezAttr,
                             bezEinh,
                             wert);

    // zweites Beispiel
    typ = ATTRIBUT;
    bezAttr = new esd2510Paar ("10", "rohdichte");
    bezEinh = new esd2510Paar ("11", "kg/m3");
    wert = 2400.0;

    // die komponente haben wir
    komponente.addAttribute (release,
                             typ,
                             bezAttr,
                             bezEinh,
                             wert);
```

```
    } catch (Exception ex) {
      // wenn einem sonst nichts einfaellt ist das immer gut
      ex.printStackTrace ();
    }
  }
}

/* eof
 *
 */
```

11.6 Klassen zu den Listings

11.6.1 esdOgip

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

abstract class esdOgip {
  public esdOgip () {
  }
}

/* eof
 *
 */
```

11.6.2 esd2510Paar

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

/**
 * denkbar, dass diese klasse via JDBC auf eine entsprechende
 * Datenbankstruktur abgebildet wird.
 * auf jeden fall sinnvoll wenn die instanzen als teil eines
 * flyweight-pool betrachtet werden
 */
class esd2510Paar {
  private String code;
  private String begriff;
  public String getBegriff () {
    return begriff;
  }
  public String getCode () {
    return code;
  }
  public String toString () {
    return begriff;
  }
  public esd2510Paar (String code, String begriff) {
    this.code = new String (code);
    this.begriff = new String (begriff);
```

```
  }
}

/* eof
 *
 */
```

11.6.3 bekPosition

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class bekPosition /* extends crbPosition */ {

  String id;
  int release;
  public String toString () {
    if (id != null)
      return "(" + release + ") " + id;
    return "()";
  }

  public void addAttribut (esd2510Paar bezAttr, esd2510Paar bezEinh,
                           int typ, String value) {
    if (typ == 1) {
      int v = (new Integer(value)).intValue();

      System.out.println (this + " / " +
                          "integer value = " + v + " / " +
                          bezAttr + " / " +
                          bezEinh);

    } else if (typ == 2) {
      double v = (new Double(value)).doubleValue();

      System.out.println (this + " / " +
                          "double value = " + v + " / " +
                          bezAttr + " / " +
                          bezEinh);

    } else if (typ == 3) {
      String v = value;

      System.out.println (this + " / " +
                          "string value = " + v + " / " +
                          bezAttr + " / " +
                          bezEinh);

    } else {
      // ein exception waere angebracht
    }
  }

  public bekPosition (int release, String id) {
    /**
     * etas sinnvolles mit release und id anstellen
     */
    this.release = release;
    this.id = id;
  }
}

/* eof
```

```
 *
 */
```

11.6.4 npkPosition

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class npkPosition /* extends crbPosition */ {

  // 0 == unbekannt || 1 == transformiert || 2 == nicht relevant
  private int esdStatus = 0;

  // das waere besser in crbPosition aufgeboden. dann waere es auch
  // moeglich fuer bekPositionen inventare anzulegen
  private esInventar inventar;
  public void setInventar (esInventar inventar) {
    this.inventar = inventar; // gc macht den rest
    esdStatus = 1;
  }
  public void setNichtRelevant () {
    this.inventar = null;
    esdStatus = 2;
  }
  public esInventar getInventar ( /* void */ /* NoNo das war einmal */ ) {
    return inventar;
  }

  public npkPosition (int release, String id) {
    /**
     * etas sinnvolles mit release und id anstellen
     */
    release = 0;
    id = null;
  }
}

/* eof
 *
 */
```

11.6.5 esInventar

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esInventar {

  // um die esKomponenten zusammenzuhalten
  private Vector inventar = new Vector ();

  public void addKomponente (int phase,
                             int lebensdauer,
                             int flussrichtung,
                             int art,
                             esKomponente komponente,
```

```
                              esd2510Paar einheit,
                              double wert) {
    // ein neuer eintrag
    esEintrag eintrag = new esEintrag (phase,
                                       lebensdauer,
                                       flussrichtung,
                                       art,
                                       komponente,
                                       einheit,
                                       wert);
    // und dazu
    inventar.addElement (eintrag);
  }


  public esInventar (int release, String id) {
    // etwas sinnvolles mit den argumenten anfangen
    release = 0;
    id = null;
  }
}

/* eof
 *
 */
```

11.6.6 esEintrag

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */


import java.io.*;
import java.util.*;

class esEintrag {

  // accessors sollten natuerlich auch geschrieben werden
  // gruppe der eigenschaften, die ueber esd2540 kommen
  private esKomponente komponente;
  private int lebensdauer;
  private int phase;
  private int flussrichtung;
  // daran denken: Inventar ...
  private int art;
  private esd2510Paar einheit;
  private double wert;
  // gruppe der eigenschaften, die zusaetzlich im anwenderprogramm
  // sinn machen
  private int verwendbarkeit;
  // ok so, aber eigentlich will man ja einheiten mitverwalten
  private double vorhaltezeit;
  // was der anwender bei eigens definierten eintraegen auch immer dazu
  // zu sagen hat
  private String anmerkung;

  public esEintrag (int phase,
                    int lebensdauer,
                    int flussrichtung,
                    int art,
                    esKomponente komponente,
                    esd2510Paar einheit,
                    double wert) {
    /**
     * aus diesen informationen koennen alle stoffluesse berechnet werden
     */
```

```
    this.phase = phase;
    this.lebensdauer = lebensdauer;
    this.flussrichtung = flussrichtung;
    this.komponente = komponente;
    this.einheit = einheit;
    this.wert = wert;

    this.verwendbarkeit = -1; // nicht gesetzt
    this.vorhaltezeit = -1.0;
    // in java sowieso: this.anmerkung = null;
  }
}

/* eof
 *
 */
```

11.6.7 esKomponente

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esKomponente extends esdOgip {

  // attribute die allen gemeinsam sind
  protected int art;
  protected esd2510Paar bezeichnung;

  protected esd2510Paar bezug; // auch hier fehlt die einheit
  protected esdKriterien kriterienHerstellung;

  public String toString () {
    if (bezeichnung != null)
      return "(" + art + ") = " + bezeichnung.getBegriff ();
    return "null";
  }

  public void addAttribute (int release,
                            int typ,
                            esd2510Paar bezAttr,
                            esd2510Paar bezEinh,
                            double wert) {
    System.out.println ("esKomponente.addAttribute = " +
                        typ + " / " +
                        bezAttr + " / " +
                        bezEinh + " / " +
                        wert);
    // kriterien (typ == 1) gehen analog zu esProdukt
    // attribute analog zu unterklassen
  }

  public esKomponente (int art, esd2510Paar bezeichnung) {
    this.art = art;
    this.bezeichnung = bezeichnung;
  }
}

/* eof
 *
 */
```

11.6.8 esProdukt

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esProdukt extends esKomponente {

  // auch hier fehlen die einheiten
  protected double masseProBezug;
  protected double rohdichte;
  protected double masseProFlaeche; // bei folien
  protected esd2510Paar deponierung;
  protected esdKriterien kriterienEntsorgung;
  protected int verwendbarkeit;
  protected int abschreibungszeit = 0; // quasi lebensdauer

  public void addAttribute (int release,
                            int typ,
                            esd2510Paar bezAttr,
                            esd2510Paar bezEinh,
                            double wert) {
    System.out.println ("esProdukt.addAttribute = " +
                        typ + " / " +
                        bezAttr + " / " +
                        bezEinh + " / " +
                        wert);
    if (typ == 0 && bezAttr.getBegriff().equals ("flmasse")) {
      masseProFlaeche = wert;
    } else if (typ == 0 && bezAttr.getBegriff().equals ("rohdichte")) {
      System.out.println ("Rohdichte = " + wert);
      rohdichte = wert;
    } else if (typ == 0 && bezAttr.getBegriff().equals ("deponierung")) {
      // der Sonderfall
      deponierung = bezEinh;
    } else if (typ == 0 && bezAttr.getBegriff().equals ("verwendbarkeit")) {
      verwendbarkeit = (int)wert;
    } else if (typ == 0 && bezAttr.getBegriff().equals ("abschr")) {
      abschreibungszeit = (int)wert;
    } else if (typ == 0 && bezAttr.getBegriff().equals ("bzgmasse")) {
      masseProBezug = wert;
    } else if (typ == 2) {
      // das wird abgewickelt wie in esd2520
      addKriterium (release,
                    bezAttr,
                    bezEinh,
                    wert);

    } else {
      super.addAttribute (release,
                          typ,
                          bezAttr,
                          bezEinh,
                          wert);
    }
  }
  private void addKriterium (int release,
                             esd2510Paar kriterium,
                             esd2510Paar einheit,
                             double wert) {
    if (kriterienEntsorgung == null)
      kriterienEntsorgung = new esdKriterien (release);
    kriterienEntsorgung.setNewKriterium (kriterium, wert, einheit);
  }
```

```
  private int getKriteriumsPos (esd2510Paar kriterium) {
    return 0;
  }

  public esProdukt (int art, esd2510Paar bezeichnung) {
    super (art, bezeichnung);
  }
}

/* eof
 *
 */
```

11.6.9 esEnergie

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esEnergie extends esKomponente {

  protected double wirkungsgrad;

  public void addAttribute (int release,
                            int typ,
                            esd2510Paar bezAttr,
                            esd2510Paar bezEinh,
                            double wert) {
    // selbst als beispiel schlecht
    if (typ == 0 && bezAttr.getBegriff().equals("wirkungsgrad")) {
      // typ == 0 >> Attribut
      this.wirkungsgrad = wert;

    } else {
      // vielleicht kann super etwas damit anfangen
      super.addAttribute (release,
                          typ,
                          bezAttr,
                          bezEinh,
                          wert);
    }
  }

  public esEnergie (int art, esd2510Paar bezeichnung) {
    super (art, bezeichnung);
  }
}

/* eof
 *
 */
```

11.6.10 esMaschine

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;
```

```
class esMaschine extends esKomponente {

  // auch hier fehlen die einheiten
  protected double verrechnungstage;
  protected double einsatzstunden;

  public void addAttribute (int release,
                            int typ,
                            esd2510Paar bezAttr,
                            esd2510Paar bezEinh,
                            double wert) {
    if (typ == 0 && bezAttr.getBegriff().equals ("vt")) {
      verrechnungstage = wert;

    } else if (typ == 0 && bezAttr.getBegriff().equals ("std")) {
      einsatzstunden = wert;

    } else {
      super.addAttribute (release,
                          typ,
                          bezAttr,
                          bezEinh,
                          wert);
    }
  }

  public esMaschine (int art, esd2510Paar bezeichnung) {
    super (art, bezeichnung);
  }
}

/* eof
 *
 */
```

11.6.11 esdKriterien

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esdKriterien extends esdOgip {

  // diese klasse koennte eine menge von kriterien in einem Array
  // verwalten
  public esdKriterien (int release, String id,
                       int phase, int lebensdauer) {
    /**
     * etwas sinnvolles mit den argumenten anstellen
     * this.release = release;
     * this.id = id;
     * this.phase = phase
     * (sinnvollerweise waere das eine oder andere member in esdOgip)
     */
    release = 0;
    id = null;
    phase = 0;
    lebensdauer = 0;
  }
  public esdKriterien (int release) {
    /**
     * etwas sinnvolles tun
     */
    release = 0;
```

```
  }
  public void setNewKriterium (esd2510Paar kriterium,
                               double wert, esd2510Paar einheit) {
    /**
     * etwas sinnvolles mit den argumenten anstellen
     * zB.
     * int pos = getKriteriumsPos (kriterium);
     * esdKriterium kriterium = new esdKriterium (wert, einheit);
     * kriterien.add (pos, kriterium);
     * datenbanknaehere repraesentationen sind denkbar
     */
  }
  public void setKriterium (int pos, double wert) {
    /**
     * wert eines kriterium auf 'wert' setzen
     */
  }
  // und so weiter (was immer eben sinnvoll ist)
}

/* eof
 *
 */
```

11.6.12 esdKomponenteException

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esKomponenteException extends Exception {
  public esKomponenteException () {

  }
}

/* eof
 *
 */
```

11.6.13 esd2510Exception

```
/**
 * t.h.e. Software GmbH, 1999, Projekt OGIP, Beispiel-Listing
 *
 * java 1.2.x
 */

import java.io.*;
import java.util.*;

class esd2510Exception extends Exception {
  public esd2510Exception () {

  }
}

/* eof
 *
 */
```

11.7 Ausgabe der Programme

```
c:/TEMP/yyy/java # java esd2510
(1) 01+hello
01=hello
01=hello
hello=01
(2) 01+hello (das wird ignoriert)
(3) 01+Dies wird eine exception geben!
esd2510Exception
     at esd2510.setBegriff(esd2510.java:59)
     at esd2510.main(esd2510.java:112)
c:/TEMP/yyy/java # java esd2520
Ein Kriterium einfuegen = 1999 / A1 121.101 / 1 / 0 / UBP / Punkte / 0.1234
c:/TEMP/yyy/java # java esd2540
Eine Komponente = 333/94 111.111 / 1999 / 0 / 0 / 0 / 1 / (1) = Bagger / min /
13.0
c:/TEMP/yyy/java # java esd2550
esProdukt.addAttribute = 1 / UBP / Punkte / 0.0123
esKomponente.addAttribute = 1 / UBP / Punkte / 0.0123
esProdukt.addAttribute = 0 / rohdichte / kg/m3 / 2400.0
Rohdichte = 2400.0
c:/TEMP/yyy/java # java esd2560
(1999) A1 121.101 / integer value = 25 / ldauer / Jahre
(1999) A1 121.101 / double value = 0.7 / glasanteil / Faktor
c:/TEMP/yyy/java #
```