MANIPULATION VON IFC PRODUKTMODELLDATEN MIT DER „EUROSTEP IFC TOOLBOX“

Siegfried Kühner und Richard Romberg
kuehner@bv.tum.de , romberg@bv.tum.de
Lehrstuhl für Bauinformatik, Technische Universität München

Kurzfassung: *Das Produktmodell Industry Foundation Classes (IFC) der IAI ist mittlerweile als Quasi-Standard für qualitativ hochwertigen Datenaustausch im Bauwesen anerkannt. Die IFC definiert Teile eines Bauwerks im sog. Express-Schema als Objekte auf einer rein formalen Ebene. Die Implementation von Software erfolgt gewöhnlich unter Verwendung von Toolboxen, die das Objektschema der IFC auf einen Klassenbaum einer objektorientierten Programmiersprache abbilden und Werkzeuge für einen effizienten Objektzugriff zur Verfügung stellen. Applikationen nutzen i.d.R. lediglich eine Teilmenge der Funktionalität dieser Toolboxen für fachspezifische Anforderungen.*

Der vorliegende Artikel gibt eine Einführung in das weit verbreitete Softwarepaket „IFC Toolbox“ der Firma Eurostep Group und demonstriert dessen Anwendung. Nach einem einführenden „Hello World“ - Beispiel wird gezeigt, wie man Informationen aus einer bestehenden IFC-Datei einliest, Attribute verändert und das modifizierte Modell wieder zurückspeichert.

1 Einleitung

Der Austausch von CAD- und Prozessdaten im Bauwesen erfährt in der Forschung wie in der Praxis eine immer größere Bedeutung. Software-Pakete, die bisher maßgeschneidert auf die Erfordernisse des Anwenders im Büro ausgeliefert wurden, benötigen Mittel um den eigenen Rahmen der Anwendung und damit das eigene Datenformat zu verlassen und ebenfalls Datenübernahmen von außen zuzulassen. Dies ist eine Konsequenz aus der zunehmend vernetzt-kooperativen Planungs- und Produktionsweise bei der Erstellung von Bauwerken. Die IAI entwickelt mit der IFC ein Datenaustausch-Format, das diese Arbeitsweise softwaretechnisch unterstützt. Es wurden bereits mehrere Schnittstellen auf der Grundlage der IFC 1.5.1-Schnittstelle zertifiziert. Das sind zur Zeit ALLPLAN FT 16 (Nemetschek), ArchiCAD 6.5 (Graphisoft), Architectural Desktop 3.0 (Autodesk) und BSpro (Olof Granlund) [5]. Der Austausch geschieht dabei im Ideal-

fall verlustfrei und ohne Datenredundanz. Ziel der Zertifizierung ist es, die Schnittstellen in diesen Punkten zu begutachten und am Idealfall zu messen.

Die IFC definiert für jedes Bauteil ein Objekt, das alle Programme, die diese Schnittstelle unterstützen, wieder als solches Bauteil interpretieren. Das ist möglich, da über die Schnittstelle nicht nur geometrische Informationen – wie bei Strichdarstellungen im DXF-Format – sondern auch semantische Informationen über den Kontext der Objekte ausgetauscht werden.

Ein Beispiel: Übergibt der Architekt seine Rohbauplanung an den Tragwerksplaner, so ist dieser in der Lage, das Datenmodell hinsichtlich der statischen Erfordernisse zu verändern oder zu erweitern und die Änderungen verlustfrei an den Architekten zurückzugeben [5].

Bei weitergehenden Überlegungen werden über die IFC-Schnittstelle verschiedene Programme aus den Bereichen AEC, statische Analyse, Projektplanung, Prozesssteuerung, Kalkulation, Kostenverwaltung, Gebäudemanagement sowie Lebenszyklus-Simulationen von Gebäuden integriert.

Bei aller Begeisterung über die potentiellen Chancen dieser Integrationsmöglichkeiten darf die softwaretechnische Umsetzung nicht unterschätzt werden. Es bedarf hohen Aufwand um Informationen aus IFC-Dateien zu extrahieren oder zu verändern. Vereinfachungen ergeben sich bei der Verwendung von Toolboxes (Ecco, eurostep, u.a.), die jeweils die Schnittstelle auf Grundlage einer IFC-Version (z.Zt. IFC 1.5.1) implementieren und Objekte zum Datenzugriff zur Verfügung stellen.

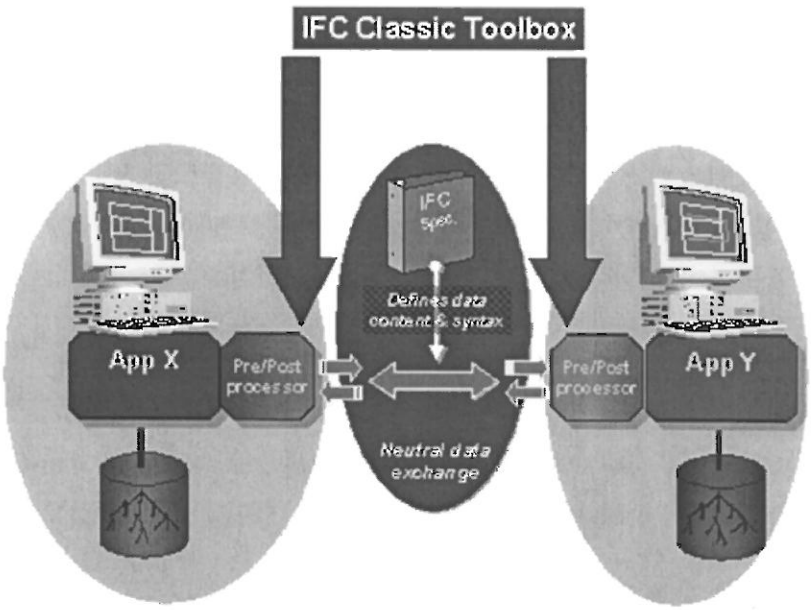

Abbildung 1: Die Rolle von Toolboxes beim Austausch von Produktmodelldaten (aus [6])

Die Toolboxes versetzten den Anwender in die Lage, innerhalb von eigenständigen Softwareentwicklungen auf Informationen aus IFC-Dateien zuzugreifen. Dieses Prinzip wird in Abbildung 1 veranschaulicht.

2 eurostep IFC Toolbox

2.1 Allgemeines

Die IFC Toolbox von eurostep Group ist das derzeit am weitesten verbreitete Werkzeug für Applikationen mit Zugriff und Verarbeitung von IFC basierten Produktmodelldaten. Das Produkt *IFC Toolbox Classic* bildet in seinen drei Versionen die entsprechenden Express Objektschemata IFC 1.5.1, IFC 2.0 sowie IFC 2.X auf eine C++ Klassenbibliothek ab.

Aktuell wird nur eine Win32 Umgebung unterstützt. Nach Angaben von eurostep sind keine maschinenabhängigen Sprachkonstrukte enthalten. Linux Portierungen müssten extra bei eurostep nachgefragt werden.

Mit der Installation der eurostep Dateien erhält man eine Debug bzw. eine Release Versionen mit jeweils den folgenden Dateien:

- ifcmain.h — Definiert Datentypen der Bibliothek IFC Toolbox
- ifcpars.h — Definiert abgebildete Datentypen des Express Schemas (Diese bindet den Header ifcmain.h ein)
- ifc151.lib — Statische Bibliothek

Wichtig ist, dass beim Linken mit der Debug-Version die Präprozessordefinitionen *DEBUG* und *IFC_BROWSE_INFO* gesetzt sein müssen.

Weitere Informationen siehe [1,2].

2.2 Abbildung von Express Datentypen auf C++ Konstrukte

- **Types** sind Basisdatentypen wie *STRING, INT* oder Aggregationen von Basisdatentypen und werden in der Toolbox als Klassen abgebildet. Beispielsweise wird der Express Datentyp *INT* zu *class IfcInt*. Diese Klasse hat lediglich eine Membervariable vom C++ Typ *int*, zusätzlich jedoch eine Reihe von Konvertierungsfunktionen (zur Vermeidung von benutzereigenen Typecasts) und Vergleichsoperatoren.

- **Enumerationen** werden als C++ *enum* sowie eine zugehörige Wrapper-Klasse zur sicheren Verwaltung dieses Datentyps übersetzt. Zur Enumeration *IfcSpaceTypeEnum* gehören damit die C++ Konstrukte *enum (IfcSpaceTypeEnumVal)* sowie *class (IfcSpaceTypeEnum)*.

- **Aggregationen:** Express unterstützt vier Arten von Aggreationen: *Bag*, *Set*, *List* und *Array*. Bags sind Mengen, in denen Objekte eines Typs mehrfach vorkommen können, während in Sets Duplikate nicht möglich sind. In der IFC Toolbox werden diese Container durch die Template Klassen *IfcBag<T>*, *IfcSet<T>*, *IfcList<T>*, *IfcXArray<T>* repräsentiert.

- **Entities** setzen sich aus **Types** zusammen und können Funktionen definieren. Entities unterstützen den Vererbungsmechanismus. Im Gegensatz zu anderen Konstrukten sind Entities referenzierbar, womit Instanzen erzeugt werden können. Die Schnittstelle zu einem Entity ist immer in einer sogenannten Handle-Klasse realisiert, während die eigentliche Implementation für den Benutzer nicht sichtbar ist. Lese- bzw. Schreibfunktionen sind in der jeweiligen Handle-Klasse enthalten. Beispiel: Zum Entity *IfcBuildingStorey* korrespondiert die Klasse *class IfcBuildingStorey_Hnd*.

3 Beispielapplikationen

3.1 IFC Toolbox „Hello World“

Der in Abbildung 2 gezeigt Quellcode zeigt wie eine IFC-Datei eingelesen und auf einzelne Instanzen zugegriffen wird. Die Funktion *Get(const IfcFile&, IfcBag<T>&)* arbeitet auf einer Instanz vom Typ *IfcFile* und extrahiert daraus alle Entities vom Typ *T*. Es sei noch darauf verwiesen, dass die Dereferenzierung des Iterators *IfcItBag<T>* über den Klammeroperator () durchzuführen ist.

Der Quellcode in Abbildung 3 zeigt den umgekehrten Vorgang. Dieser lässt sich in drei wesentliche Bestandteile gliedern: Mit den Instanzen der Typen *IfcFileDescription* und *IfcFileName* wird der Datei-Header erzeugt. Instanzen von Entities werden ausschließlich mit Member-Funktionen der Klasse *IfcFile* erzeugt, d.h. allgemein betrachtet gibt es für jeden Typ *IfcXXX* in der Klasse *IfcFile* eine korrespondierende Member-Funktion *XXX(Attribute von XXX)*. Abschließend wird die Datei mit der Funktion *WriteTheFile* geschrieben. Eine detaillierte Beschreibung der einzelnen Funktionen respektive der Parameter kann in [1], [2] nachgelesen werden.

```
// main.cpp

// IFC Toolbox Header
#include "ifcpars.h"

void main()
{
    cout << "Demo Programm fuer Ifc_151" << endl;

    // File Objekt erzeugen
    IfcFile File;

    // Verwendetes Schema testen
    if (File.ReadTheFile ("helloworld.ifc", IfcFile::RM_ONLY_HEADER))
    {
        if (!File.IsFileOfSchema()) cout << "Falsches Schema" << endl;
        else
        {
            // File lesen
            File.ReadTheFile("helloworld.ifc");

            // Alle Entities vom Typ IfcNotationFacet in einem IfcBag sammeln
            IfcBag<IfcNotationFacet_Hnd> hP;
            Get (File, hP);

            // Iteration über IfcBag und Ausgabe des Inhalts
            for (IfcItBag<IfcNotationFacet_Hnd> iP (hP); iP; iP ++)
            {
                // Dereferenzieren des Iterators mit Klammeroperator ()
                cout << "IfcNotationFacet: " << iP().NotationValue()
                     << " " << iP().PurposeOpt() << endl;
            }
            cout << "Lesevorgang war erfolgreich" << endl;
        }
    }
}
```

Abbildung 2: Lesen einer IFC-Datei

```
// main.cpp

// IFC Toolbox Header
#include "ifcpars.h"

void main()
{
    cout << "Demo Programm fuer Ifc_151" << endl;

    // File Objekt erzeugen
    IfcFile File;

    // File-Header Erzeugen: Dateibeschreibung und Dateiname
    IfcFileDescription FileDescr(IfcList<IfcString>
        (1, IfcString ("Nur eine Demo Datei")));
    IfcFileName FileName
    (
        "helloworld_out.ifc",
        IfcList<IfcString> (1, IfcString ("Author Name")),
        IfcList<IfcString> (2, IfcString ("LS BI"), IfcString ("MUC")),
        "PreProc",  "Windows 2000", "I am the author."
    );

    // Entity vom Typ NotationFacet erzeugen
    File.NotationFacet ("Hello", "world");

    // File schreiben
    if (File.WriteTheFile ("helloworld_out.ifc", IfcCLASS_1, FileName, FileDescr, TRUE))
        cout << "Schreibvorgang war erfolgreich" << endl;

}
```

Abbildung 3: Schreiben einer IFC-Datei

3.2 Verändern der Wandhöhe eines Raums

Im folgenden Abschnitt wird gezeigt, wie auf geometrische Informationen eines IFC Modells zugegriffen wird. Dazu wurde zunächst das in Abbildung 4 dargestellte Einfamilienhaus mit dem ADT 2.0 der Fa. Autodesk erstellt. Aus didaktischen Gründen wurde dieses Modell auf die in diesem Kontext notwendigen Elemente reduziert und in ein *IFC physical file* exportiert. Änderungen werden in diesem Beispiel an der Wandhöhe vorgenommen.

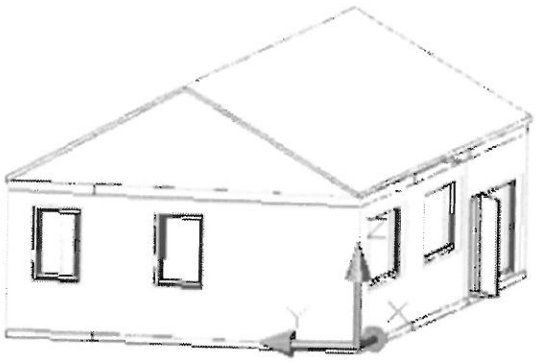

Abbildung 4: Einfamilienhaus

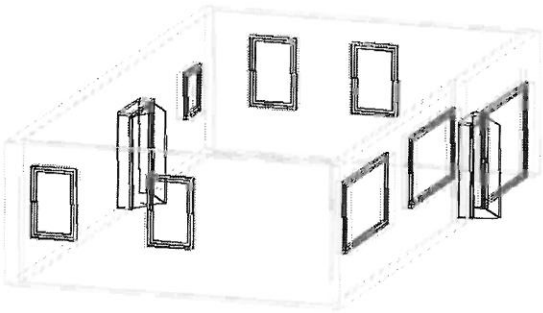

Abbildung 5: vereinfachtes Modell des Einfamilienhaus

Die in Abbildung 6 gezeigte main-Funktion enthält die aus Beispiel 3.1 bekannte Ablaufsteuerung zum Lesen einer IFC-Datei. Die anschließende Iteration greift aus dem Modell alle Wände heraus und übergibt jeweils das *Handle* einer Wand an die Funktion *changeWall*. Abschließend wird das modifizierte Modell in die Datei *haus_out.ifc* abgespeichert.

Zur Funktion *changeWall* (Abbildung 7) sei folgendes bemerkt: Die IFC Toolbox ist eine Interface-Bibliothek zum Zugriff auf IFC konforme Modelle. Daraus resultiert, dass dem Anwender das IFC Schema bekannt sein muss. Dieser Umstand wird beim Zugriff auf die geometrische Information 'Wandhöhe' ersichtlich, bei dem eine Reihe von Entities involviert sind:

IfcWall ⇒ IfcProductShapeOpt ⇒ IfcShapeRepresentation ⇒
IfcGeometricRepresentationItem ⇒ IfcAttDrivenExtrudedSolid ⇒
IfcAttDrivenExtrudedSegment ⇒ IfcRectangleProfileDef ⇒
IfcPositiveLengthMeasure

Die äußere Schleife in der Funktion *changeWall* arbeitet auf dem Entity *IfcShapeRepresentation*. Anzumerken ist hier, dass dem Entity *IfcWall* verschiedene Repräsentationen zugeordnet sein können (z.B. Bounding-Box, extrudierter Körper).

Diese Repräsentationen setzen sich je nach Komplexität der Geometrie aus einer oder mehreren Instanzen vom Typ *IfcGeometricRepresentationItem* zusammen. Die korrespondierende Klasse *IfcGeometricRepresentationItem_Hnd* gibt über seine Memberfunktionen *IsXXX()* Auskunft über den Typ der Repräsentation (z.B. *IsBoundingBox*), welcher mit der Member-Funktion *CastToXXX()* konvertiert wird (z.B. *Cast2BoundingBox*).

```
// main.cpp

// IFC Toolbox Header
#include "ifcpars.h"

void changeWall(const IfcWall_Hnd& wh);

void main()
{
    cout << "Demo Programm fuer Ifc_151" << endl;

    // File Objekt erzeugen
    IfcFile File;

    // Verwendetes Schema testen
    if (File.ReadTheFile ("haus_in.ifc", IfcFile::RM_ONLY_HEADER))
    {
        if (!File.IsFileOfSchema()) cout << "Falsches Schema" << endl;
        else
        {
            // File lesen
            File.ReadTheFile("haus_in.ifc");

            // Alle Entities vom Typ IfcWall in einem IfcBag sammeln
            IfcBag<IfcWall_Hnd> hW;
            Get (File, hW);

            // Iteration über alle IfcWall Entities
            for (IfcItBag<IfcWall_Hnd> iW(hW); iW; iW++)
            {
                changeWall(iW());
            }

            // File wieder herausschreiben
            File.WriteTheFile("haus_out.ifc", IfcCLASS_1, File.FileName(),
                                File.FileDescription());
        }
    }


}
```

Abbildung 6: main-Routine des Beispiels 'ChangeWall'

```
void changeWall(const IfcWall_Hnd& wh)
{
    double eps = 3.0;

    // Im weiteren Verlauf verwendete Entitie Handles
    IfcProductShape_Hnd hProdShape;
    IfcList<IfcShapeRepresentation_Hnd> shapeRepList;
    IfcBag<IfcGeometricRepresentationItem_Hnd> geomRepBag;
    IfcBoundingBox_Hnd hBound;
    IfcAttDrivenExtrudedSolid_Hnd hExtrSolid;
    IfcAttDrivenExtrudedSegment_Hnd hExtrSeg;
    IfcPositiveLengthMeasure posLenMeas;
    IfcRectangleProfileDef_Hnd hRectProfile;

    // Eine Wand speichert sein Geometrie in einem IfcProductShape
    hProdShape = wh.ProductShapeOpt();

    // Ein IfcProductShape hält die Geometrieinformationen in mehreren
    // Instanzen vom Typ IfcShapeRepresentation
    // Fuer den Fall einer Wand wird meistens eine BoundingBox sowie
    // ein extrudierter Körper gespeichert
    shapeRepList = hProdShape.ShapeRepresentations();

    // Iteration ueber alle Shape Representations
    for (IfcItBag<IfcShapeRepresentation_Hnd> iSR(shapeRepList); iSR; iSR++)
    {
        // Alle Items der Shape Representations in einem Bag sammeln
        geomRepBag = iSR().Items();

        // Iteration ueber alle Items der Shape Representation
        for(IfcItBag<IfcGeometricRepresentationItem_Hnd> iGeomRep(geomRepBag);
                                                          iGeomRep; iGeomRep++)
        {
            // Fall A: Shape Representation ist eine BoundingBox
            if (iGeomRep().IsBoundingBox())
            {
                // Die Bib. stellt cast-Funktionen zur Verfügung
                // damit implizite Type-Konvertierungen vermieden werden
                hBound = iGeomRep().Cast2BoundingBox();

                // Hoehe lesen
                posLenMeas = hBound.ZDim();
                // Hoehe um eps vergroessern
                posLenMeas = posLenMeas + eps;
                // Hoehe zurückspeichern
                hBound.ZDim(posLenMeas);
            }
            // Fall B: Shape ist ein extrudierter Körper
            else if (iGeomRep().IsAttDrivenExtrudedSolid())
            {
                hExtrSolid = iGeomRep().Cast2AttDrivenExtrudedSolid();

                // Ein extrudierter Körper (IfcAttDrivenExtrudedSolid) setzt sich
                // aus mehreren Segmenten zusammen (IfcAttDrivenExtrudedSegment).
                // In diesem Beispiel wird vereinfachend angenommen, dass die
                // Liste der Segmente lediglich ein Element enthält
                hExtrSeg = hExtrSolid.Segments().GetFirst();

                // In diesem Fall wird mit dem Wissen über das IFC-Modell
                // ein Rectangle als Profil erwartet.
                hRectProfile = hExtrSeg.ProfileDef().Cast2RectangleProfileDef();

                // Hoehe lesen, um eps vergroessern und zurückspeichern
                posLenMeas = hRectProfile.XDim();
                posLenMeas = posLenMeas + eps;
                hRectProfile.XDim(posLenMeas);
            }
            else
                cout << "Diese RepresentationItem ist nicht implementiert" << endl;
        }
    }
}
```

Abbildung 7: Funktion zum Verändern der Wandhöhe

4 Zusammenfassung und Ausblick

Dieser Artikel will den Einstieg in die Entwicklung von Applikationen mit Unterstützung von IFC basiertem Datenaustausch unter Verwendung der IFC Toolbox von eurostep erleichtern. An zwei Beispielen wurde der Umgang mit IFC-Dateien sowie der Manipulation von IFC-Objekten gezeigt. Die folgenden Features der IFC Toolbox konnten im begrenzten Rahmen dieses Tutoriums nicht behandelt werden, sind aber wichtige Komponenten, um den Entwicklungsprozess flexibler und effizienter zu gestalten:

- Benutzerdefinierte Entities können von der IFC Toolbox gelesen und geschrieben werden. Der Korrektheit des Entities bleibt natürlich der Applikation selbst vorbehalten.
- Um die Typsicherheit in vom Benutzer abgeleiteten Klassen sicherzustellen, können mit dem Tool „AddVirtual" Klassen der IFC Toolbox neue virtuelle Funktionen hinzugefügt werden, wobei nur das Klassen-Interface geändert wird.
- Ein sehr mächtiger Logging Mechanismus beschleunigt das Auffinden von Programmierfehlern sowie die Identifikation korrupter IFC-Dateien.

Literatur

[1] Eurostep Group AB: The IFC STEP Toolbox – Version 1.51 Addendum, Beschreibung der IFC Toolbox Classic, 1998

[2] Eurostep Group AB: The IFC STEP Toolbox – Version 2.X, Beschreibung der IFC Toolbox Classic, 2000

[3] Industrieallianz für Interoperabilität e.V: IFC Object model referencery – Industry foundation classes – Release 1.5.1 - Model Reference Documentation, http://www.iai-ev.de/spezifikation/IFC151/index.htm, 1998

[4] Industrieallianz für Interoperabilität e.V: Industry foundation classes 2x, Online Dokumentation, http://www.iai-ev.de/spezifikation/IFC2x/index.htm, 2000

[5] International Alliance for Interoperability: Web-Seite, http://iaiweb.lbl.gov/

[6] Eurostep Group AB: Eurostep IFC Toolbox Products, Web-Seite: http://www.eurostep.com