Ein Darstellungsmodell und eine Abbildungssprache für INTERLIS

Inhaltsverzeichnis

1. Einleitung

1.1 Anlass

Die Eidg. Vermessungsdirektion (V+D) ist für die Überarbeitung und Pflege von INTERLIS [1] zuständig. Nach dem Abschluss der Expertise „Inkrementelle Nachlieferung mit INTERLIS“ [6] wurde die Realisierung von INTERLIS Version 2 beschlossen [2]. Die V+D hat für INTERLIS einen Releaseprozess definiert zu dem von verschiedenen Stellen 15 Vorschläge (RFC's) eingereicht wurden. Davon behandelt ein Vorschlag das Thema „Ein Darstellungsmodell und eine Abbildungssprache für INTERLIS“ RFC-1011 [3]. Die hier vorliegende Expertise zum gleichen Thema soll den RFC-1011 bis zur Produktreife ausarbeiten.

1.2 Problemstellung

Aufgrund von Benutzeranfragen ist eine Erweiterung von INTERLIS (Version 1) erforderlich, um die Anforderungen der Praxis nach der Visualisierung von Geodaten abzudecken. Es sollen folgende Aufgabenbereiche bearbeitet werden:

- Formulierung eines Darstellungsmodells in INTERLIS, das die systemneutrale Beschreibung von Signaturen erlaubt (Symbole, Strichlierungen, Füllungen, etc.).
- Definition einer Abbildungssprache, die es erlaubt, Geoobjekte zu selektieren und sie einer graphischen Darstellung zuzuweisen.
- Erarbeitung von Beispielen, anhand derer die neuen Konzepte überprüft werden können. Bemerkung: Ein ausführliches Beispiel wird in einem separaten Auftrag zur Expertise bearbeitet.

Der oben definierte Aufgabenbereich ist noch sehr allgemein formuliert und muss daher wie folgt präzisiert werden:

- Primär sollen mit dem Darstellungmodell und der Abbildungsprache die Anforderungen des Plans für das Grundbuch abgedeckt werden. Im Moment sind die Darstellung des Plan für das Grundbuch im Form von Zeichenvorschriften vorhanden. Mit den neuen Mitteln können die Zeichenvorschriften durch computerlesbare Definitionen abgelöst werden.
- Sekundär sollen auch die Anforderungen des Leitungskatasters (SIA-405) berücksichtigt werden.
- Das Darstellungsmodell soll für weitere Anwendungen anpassbar sein.
- Die Expertise verfolgt nicht das Ziel Graphikstandards wie z.B. HP-GL, Postscript, OpenGL etc. durch einen INTERLIS-Graphikstandard abzulösen. Vielmehr wird das vorgestellte Konzept erlauben bestehende Standards auf einheitliche Weise für die Erzeugung von Graphik zu nutzen (s.a. Kapitel 5).
- Wir beschränken uns in dieser Expertise auf die Abbildung von 2D-Graphik.

1.3 Nutzen

Das von uns vorgeschlagene, in INTERLIS formulierte Darstellungsmodell, erlaubt die systemneutrale Beschreibung von Signaturen (Symbole, Strichlierungen, Schraffuren, Füllungen etc.). Die neue Abbildungsprache ermöglicht die Zuordnung von Signaturen zu Objekten des Datenmodells.

Daraus ergibt sich folgender Nutzen für den Anwender:

- Die Signaturen müssen nur einmal systemneutral definiert werden und können von allen Systemen welche das Darstellungsmodell unterstützen gemeinsam genutzt werden.
- Die Definitionen der Signaturen werden zusammen mit den Daten in der Transferdatei übermittelt.
- Die Abbildungssprache ermöglicht die Zuweisung von Datenobjekten zu Signaturen. Diese Abbildung wurde bis jetzt in jedem System in internen Tabellen definiert. Neu kann die Definition der Abbildung aus dem um die Abbildungssprache erweiterten Datenmodell entnommen werden.

1.4 Aufbau der Expertise

Die weiteren Kapitel dieser Expertise sind wie folgt aufgebaut:

- In Kapitel 2 wird eine Übersicht über die Problematik gegeben und ein allgemeines Lösungsmodell vorgestellt.
- In Kapitel 3 wird ein erstes Beispiel präsentiert.
- In Kapitel 4 wird das in INTERLIS beschriebene Darstellungsmodell vorgestellt.
- In Kapitel 5 wird die Abbilungsprache erläutert.
- In Kapitel 6 wird die Umsetzung von systemneutraler Graphik in konkrete Graphik besprochen.
- In Kapitel 7 geben wir Hinweise für die Implementierung von komplexen Graphikmodellen. Ausserdem geben wir Empfehlungen betreffend der Auswirkungen unserer Vorschläge auf den Grunddatensatz [4].
- Im Anhang ist ein Literaturverzeichnis, die Syntax der Abbildungssprache und das Darstellungsmodell in INTERLIS enthalten.

1.5 Bezug zu anderen RFC's und Expertisen

Neben dem bereits erwähnten RFC-1011 hat diese Expertise auch einen Bezug zum RFC- 1008 (OO-INTERLIS, Ein objektorientiertes INTERLIS). Der RFC-1008 wird ebenfalls im Rahmen einer Expertise [5] weiter ausgearbeitet. INTERLIS-2 bildet die Basis für die hier vorgestellten INTERLIS-Erweiterungen. Auf eine Wiederholung der Möglichkeiten von INTERLIS-2 wird hier wegen der Vermeidung von Doppelspurigkeiten verzichtet (s.a. [2] und [5]).

2. Lösungsansatz

2.1. Was ist Graphik im Allgemeinen ?

Da die Graphik immer der sichtbare Teil einer Anwendung ist (z.B. in Form eines Plots oder einer Darstellung auf dem Bildschirm), haben die meisten Benutzer eine ziemlich konkrete Vorstellung der graphischen Möglichkeiten ihres Systems. Trotzdem bleibt die Frage: *Was ist Graphik im allgemeinen*, d.h. über die Systemgrenzen hinweg ? Wenn man z.B. einen Plot etwas genauer analysiert, stellt man fest, dass eine Graphik aus vielen einzelnen Grundelementen (Graphikobjekte) aufgebaut ist. Bei den Graphikobjekten kann man im 2D-Fall folgende Typen unterscheiden:

- **Text**. Textobjekte können u.a. in verschiedenen Grössen, Schriftarten (Font's), Farben etc. dargestellt werden.
- **Linien**. Linien weisen eine Strichbreite, Strichlierung, Bemusterung (z.B. Landesgrenze) etc. auf.
- **Flächen**. Flächen können gefüllt, schraffiert, umrandet, bemustert, etc. sein.
- **Symbole**. Symbole setzen sich aus einer fixen Kombination aus Text, Linien und Flächen zusammen. Symbole können u.a. skaliert und rotiert werden.

Man sieht, dass eine Graphik auch aus Objekten besteht. Diese Objekte lassen sich natürlich grundsätzlich in INTERLIS beschreiben.

2.2. Das Darstellungsmodell

Für die Beschreibung der Graphikobjekte führen wir ein in INTERLIS formuliertes Darstellungsmodell ein (s.a. Kapitel 4). Darstellungsmodelle sind immer auch von den konkreten Anwendungen (hier: Vermessung und Leitungskataster) abhängig. Das von uns vorgeschlagene Darstellungsmodell ist daher auf seine primären Zielanwendungen optimiert (Grundbuchvermessung, Leitungskataster). Durch die Möglichkeiten von INTERLIS-2 ist es jedoch möglich das Darstellungsmodell für neue Aufgabenbereiche anzupassen.

2.3. Die Abbildungssprache

Die Objekte einer Graphik sind nicht willkürlich angeordnet, sondern haben einen Bezug zu Objekten eines konkreten Datenmodells (z.B. AV93). Man kann daher eine Graphik auch als Abbildung eines Daten- auf ein Darstellungsmodell verstehen. Als Formel kann dieser Sachverhalt wie folgt ausgedrückt werden:

Graphik = F(Datenmodell)

Für die Beschreibung der Abbildung F() fehlen uns in INTERLIS im Moment die geeigneten Mittel. In Kapitel 5 werden daher entsprechende INTERLIS-Erweiterungen (Abbildungssprache) vorgestellt.

Bemerkungen zur Abbildungsfunktion F():

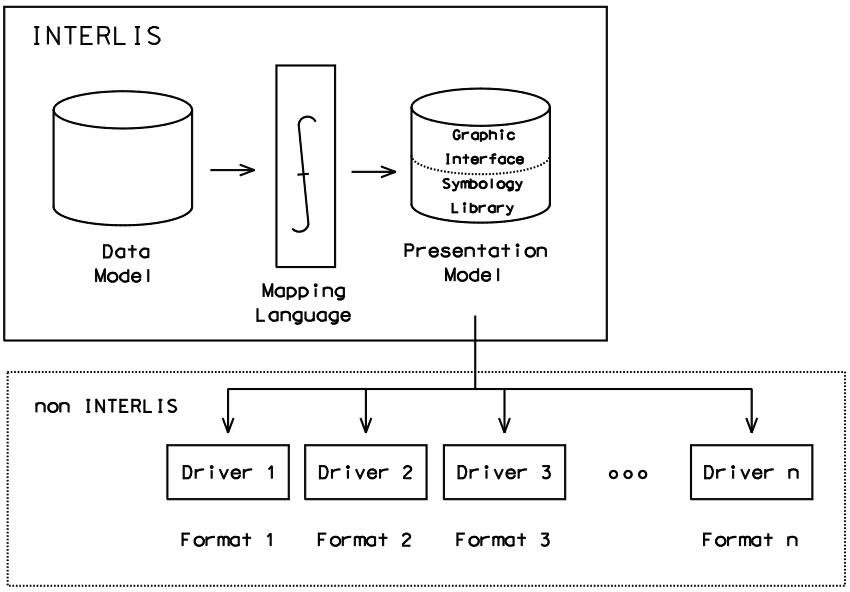
- Die Objekte der Graphik müssen nicht echt vorhanden (d.h. instanziert) sein, da sie durch die Funktion F() jederzeit aus den Objekten des Datenmodells erzeugt werden können. Eine redundante Speicherung der Graphikobjekte ist daher nicht notwendig.
- Man kann einwenden, dass es CAD-Systeme gibt in denen nur die Graphik jedoch kein Datenmodell (im Sinn einer Datenbank) existiert. In diesem Fall verhält es sich jedoch so, dass im CAD-System ein internes (allenfalls fixes) Datenmodell existiert aus dem die für den Benutzer sichtbaren Graphikobjekte durch eine systemspezifische Darstellungsfunktion F() erzeugt werden.

2.4. Von systemneutraler zu konkreter Graphik

Die Graphikobjekte werden wir in Kapitel 4 mit dem Darstellungsmodell systemneutral beschreiben. Wie bereits erläutert, wollen wir jedoch durch unsere Vorschläge nicht etablierte Graphikstandards ersetzen. Wir müssen daher angegeben, wie aus systemneutraler Graphik konkrete Graphik in einem bestimmten Format (z.B. HP-GL, Postscript, etc.) wird. Der dazu notwendige Prozess ist in Kapitel 6 beschrieben.

2.5. Der Lösungsansatz im Überblick

Zum Abschluss dieses Kapitel möchten wir unseren Lösungsansatz nochmals zusammenstellen. Folgende Figur zeigt das von uns vorgeschlagene Verfahren im Überblick.



Erläuterungen:

- Durch die Abbilungsfunktion F() werden die Objekte des Datenmodells auf das Darstellungsmodell abgebildet. Die Abbildungsfunktion F() wird in der neuen INTERLIS-Abbildungssprache formuliert.
- Das Darstellungsmodell besteht aus dem Graphikinterface und der Signaturbibliothek. Das Darstellungsmodell ist systemneutral in INTERLIS beschrieben. Durch die Möglichkeiten von INTERLIS-2 ist es möglich das Darstellungsmodell im Bedarfsfall zu erweitern oder sogar zu ersetzen.
- Das Graphikinterface ermöglicht die Erzeugung der Graphikobjekte Text, Linie, Fläche und Symbol.
- Die Signaturbibliothek enthält Text-, Linien-, Flächen- und Symbolsignaturen. Die Signaturen bestimmen die konkerete Ausprägung (z.B. Font, Farbe, Strichlierung etc.) eines Graphikprimitvs auf dem Ausgabeformat (Ausgabegerät).
- Die systemneutralen Objekte des Darstellungsmodells werden über Treiber (s.a. Kapitel 6) auf ein konkretes Graphikformat abgebildet.

3. Ein Beispiel für den Einstieg

3.1 Aufgabenstellung

Bevor nun in den folgenden Kapiteln das Darstellungsmodell und die Abbildungsprache im Detail vorgestellt werden, möchten wir ein einfaches aber vollständiges Beispiel für das Datenmodell `Grunddatensatz` (AV93) angeben. Die zu lösende Aufgabe wird wie folgt definiert:

- Fixpunkte sollen im Masstab 1:500 und 1:10'000 dargestellt werden.
- Im Massstab 1:500 soll für `LFP1` und `LFP2` ein Symbol und ein Text dargestellt werden.
- Im Massstab 1:10'000 soll nur ein Symbol dargestellt werden.

3.2 Lösung

Die Lösung sieht wie folgt aus (Erweiterungen sind fett dargestellt):

```
MODEL Grunddatensatz EXTENSION OF ILI_PRESENTATION

    DOMAIN
    !! Wertebereichsdefinitionen ...

    PRESENTATION !! mögliche Darstellungsarten
      M500;M10000;

    TOPIC Fixpunkte =
      TABLE LFP =
        Entstehung: OPTIONAL -> LFPNachfuehrung; !! Beziehung 1-mc
        Nummer: TEXT*12; !! Vergabe durch Landestopographie
        NumPos: LKoord;
        NumOri: OPTIONAL SchriftOri; !! Default: 100.0
        NumHAli: OPTIONAL HALIGNMENT; !! Default: Center
        NumVAli: OPTIONAL VALIGNMENT; !! Default: Half
        SymbolOri: OPTIONAL SchriftOri; !! Default: 0.0
        Geometrie: HKoord;
        !! etc.
        Art: (LFP1, LFP2);
        Herkunft: OPTIONAL TEXT*30; !! Vergabe durch Kanton
      CONSTRAINT
        UNIQUE Nummer;
        UNIQUE Geometrie;
      MAPPING !! Definition von F() für M500 und M10000
        M500:
          IF Art = LFP1 THEN
            P_SYMBOL(Geometrie,SymbolOri,"Fix500_LFP1",1000);
          ELSIF Art = LPF2 THEN
            P_SYMBOL(Geometrie,SymbolOri,"Fix500_LFP2",1000);
          END;
          P_TEXT(Nummer,NumPos,NumOri,NumHAli,NumVAli,"Fix500_LFP",
            1000);
        M10000:
          P_SYMBOL(Geometrie,SymbolOri,"Fix10000_LFP",1000);
      END LFP;
    END Fixpunkte

  END Grunddatensatz.
```

Erläuterungen zur Lösung:

- Da im Modell `Grunddatensatz` graphische Darstellungen definiert werden sollen, muss das Modell `Grunddatensatz` vom vordefinierten Darstellungsmodell `ILI_PRESENTATION` abgeleitet werden.
- Die Definitionen des Graphikinterface und die Symbolbibliothek werden aus `ILI_PRESENTATION` geerbt (s.a. Kapitel 4 und Anhang A3).
- Die gewünschten Darstellungsarten 1:500 und 1:10'000 müssen zuerst unter `PRESENTATION` (`M500` bzw. `M10000`) deklariert werden.
- Für die Objekte in der Tabelle `Fixpunkte.LFP` muss die Abbildungsfunktion F() angegeben werden. F() wird im `MAPPING`-Teil der Tabelle definiert.
- Für jede Darstellungsart (`M500` bzw. `M10000`) wird die Funktion F() durch Aufrufe von Graphikinterfaceprozeduren (`P_SYMBOL()` und `P_TEXT()`) kodiert.
- Im `MAPPING`-Teil ist es möglich verschiedene Darstellungsarten gleichzeitig zu definieren (hier `M500` und `M10000`).
- Die Signaturen werden den Graphikobjekten über den Signaturnamen zugewiesen (hier: `"FIX500_LFP1"`, `"FIX500_LFP2"`, `"FIX500_LFP"` und `"FIX10000_LFP"`). Die Priorität der Signaturen wird als letztes Argument des Prozeduraufrufs (hier: `1000`) übergeben.
- Die Ausprägung der Signaturen (z.B. die Geometrie der Symbole) ist im Symbolgiemodell abgelegt.

Bemerkung: Die hier vorgestellte Lösung ist bewusst sehr einfach gehalten. Die Darstellungsfunktion F() wurde daher direkt in das Datenmodell integriert. Bei komplexen Datenmodellen ist es wünschbar die Graphik vom Datenmodell zu trennen. Wir werden in Kapitel 7 auf diese Problematik zurückkommen.

4. Das Darstellungsmodell

4.1 Einleitung

Nach der Einführung der letzten Kapitel stellt sich natürlich sofort die Frage wie das Darstellungsmodell konkret aufgebaut ist. Da alle Graphikobjekte durch das Darstellungsmodell systemneutral beschrieben werden, sind durch die Eigenschaften des Darstellungsmodells auch die prinzipiellen Möglichkeiten der Graphik gegeben. Das hier vorgestellte Darstellungsmodell wird durch folgende Grundeigenschaften charakterisiert:

- Das Darstellungsmodell ist vollständig in INTERLIS beschrieben.
- Das Darstellungsmodell ist in die Teile Graphikinterface und Signaturbibliothek aufgeteilt.
- Das Graphikinterface ermöglicht die Erzeugung der Graphikobjekte Text, Linie, Fläche und Symbol durch vordefinierte Prozeduraufrufe.
- Die Signaturbibliothek stellt die graphische Ausprägung der Graphikobjekte (Signaturen) in Form einer Bibliothek bereit. Z.B. enthält die Signaturbibliothek Angaben über die Strichlierung von Linien oder die Füllung von Flächen.

Bemerkung: Ein Darstellungsmodell ist immer zu einem gewissen Grad von der konkreten Anwendung abhängig. Das hier vorgestellte Darstellungsmodell ist auf die Bedürfnisse der Vermessung und des Leitungskatasters abgestimmt. Es ist jedoch möglich das Darstellungsmodell durch die Mechanismen von INTERLIS-2 bei Bedarf zu erweitern oder sogar zu ersetzen.

4.2. Allgemeiner Aufbau

Das Darstellungsmodell ist wie folgt aufgebaut:

```
MODEL ILI_PRESENTATION

   !! Deklarationen

   !! Graphikinterface

   TOPIC SYMBOLOGY
      !! Signaturbibliothek
   END SYMBOLOGY.

END ILI_PRESENTATION.
```

Erläuterungen:

- Die Deklarationen enthalten die Definitionen von Graphiktypen und von allgemein verwendbaren Graphikprozeduren (s.a. Anhang A3).
- Das Graphikinterface ist in Form von Prozedurdefinitionen vorhanden, d.h. die Graphikobjekte werden nicht instanziert sondern direkt an die Graphiktreiber weitergeleitet.
- Die Signaturbibliothek ist als Topic `SYMBOLOGY` implementiert. Das Topic `SYMBOLOGY` wird von jedem Datenmodell, das sich von `ILI_PRESENTATION` ableitet automatisch geerbt. Die Objekte der Signaturbibliothek müssen vom Benutzer datenmässig erfasst (instanziert) werden.

4.3. Das Graphikinterface

Das Graphikinterface definiert die Grundeigenschaften der Graphikobjekte Text, Linie, Symbol und Fläche (Surface). Es bildet die Schnittstelle zu den Treiberprogrammen. Jedes Graphikobjekt enthält einen

Verweis auf ein Signaturobjekt aus der Signaturbibliothek. Zusammen mit der Signatur und der Priorität kann das Graphikobjekt auf einem Ausgabegerät dargestellt werden. Nachfolgend ist die Definition des Graphikobjekts Text in INTERLIS angegeben (für alle weiteren Definitionen s.a. Anhang A3):

```
!! Graphikinterface
FUNCTION
   P_TEXT (
     TXT       : TEXT; !! Textinhalt
     GEOMETRY  : COORD2; !! Position
     ROTATION  : DEGREES; !! Rotation
     HALI      : HALIGNMENT; !! Alignment
     VALI      : VALIGNMENT; !! Alignment
     SYMBOLOGY : P_TEXTID; !! Verweis auf Signaturbibliothek
     PRIORITY  : P_PRIORITY !! Priorität der Signatur
   ) = // Erzeugt ein Textobjekt auf dem Ausgabegerät //;
```

Erläuterungen:

- Das Graphikinterface enthält nur die Grundeigenschaften der Graphikobjekte (hier: Inhalt, Position, Rotation, Alignment).
- Über das Argument `SYMBOLOGY` wird die Verbindung zu einer Signatur der Signaturbibliothek (s.a. 4.4) hergestellt.
- Mit dem Argument `PRIORITY` wird die Priorität festgelegt mit der die Signatur ausgegeben wird.
- Der Aufruf einer Graphikinterfaceprozedur im `MAPPING`-Teil erzeugt ein Graphikobjekt. Das Graphikobjekt wird jedoch nicht im Darstellungsmodell abgespeichert sondern, sondern direkt an den Treiber für die Darstellung auf einem Ausgabegerät (Ausgabeformat) weitergegeben (s.a. Kapitel 6).

4.4. Die Signaturbibliothek

In der Signaturbibliothek sind Signaturen als Bibliothek abgelegt. Folgende Signaturen werden unterstützt:

- **Textsignatur** bestehend aus Font, Grösse, Farbe, etc.
- **Symbolsignatur** zusammengesetzt aus Text-, Linien- und Flächenobjekten.
- **Liniensignatur** mit der Möglichkeit Mehrfachlinien, Strichlierungen und Bemusterungen (Symbol entlang Linie) zu definieren.
- **Flächensignatur** bestehend aus Randsignatur, Flächenfüllung, Flächenschraffur und Flächenmuster.

Neben den individuellen Eigenschaften der Signaturen unterstützt jede Signatur folgende Eigenschaften:

- **Clipbereich**. Über den Clipbereich kann eine Signatur freigestellt werden (z.B. Freistellen von Grenzpunktsymbolen auf Grenzlinien).
- **Color**. Farben werden im Darstellungsmodell als RGB (Rot-Grün-Blau) Werte angegeben. Farben können unter einem Namen (z.B. `black`) in der Tabelle `COLORS` definiert werden.

Die vollständige Signaturbibliothek ist im Anhang A3 in INTERLIS formuliert (Topic `SYMBOLOGY`).

4.4.1 Eigenschaften von Textsignaturen

Für Textsignaturen (`TEXT_SYMBOLOGY`) kann die Schriftart (`FONTNAME`) die Höhe der Schrift (`HIGHT`) und diverse Textattribute (`SLANTED` kursiv, `UNDERLINED` unterstrichen, `STRICKED` durchgestrichen) angegeben werden. Bei internen Fonts (`TYPE = internal`) wird die Definition des Fonts in den Tabellen `FONTLIB`, `FONTSYMBOL`, `FONTSYMBOL_LINE` und `FONTSYMBOL_SURFACE` übermittelt. Bei externen Fonts (`TYPE = external`) wird nur der Name des Fonts in `FONTNAME` übertragen. Die Clippingparameter (`CLIPBOX`, `CLIPDIST`) haben folgende Bedeutung:

- CLIPBOX: Mit CLIPBOX wird ein rechteckiger Clipbereich um den aktuellen Text definiert.
- CLIPDIST: Mit CLIPDIST wird ein Clipbereich entlang jedes Buchstabens des aktuellen Texts definiert.

4.4.2 Eigenschaften von Symbolsignaturen

Ein Symbol (SYMBOL_SYMBOLOGY) besteht aus Linien- und Flächenobjekten die in den Tabellen FONTSYMBOL_LINE und FONTSYMBOL_SURFACE definiert werden. Zusätzlich kann einem Symbol ein Skalierungsfaktor (SCALE) und ein Clipsymbol (CLIPSYMBOL) zugeordnet werden. Die Symbolgeometrie wird relativ zum Ursprung 0.000/0.000 definiert. Der Ursprung dient später bei der Plazierung des Symbols als Einfügepunkt. Symbolegeometrien werden im Darstellungsmodell analog wie Zeichen eines Textfonts behandelt. Symbolgeometrien werden daher in Fontbibliotheken zusammengefasst (Tabelle FONTLIB).

4.4.3 Eigenschaften von Liniensignaturen

Eine Liniensignatur (LINE_SYMBOLOGY) kann durch eine oder mehrere Teillinien definiert werden. Für jede Teillinie muss ein Abstand von der Achse (OFFSET) angegeben werden. Für jede Teillinie kann ausserdem die Darstellungsart NORMAL, DASHED und PATTERNED gewählt werden. Die Darstellungsarten haben folgende Bedeutung:

- NORMAL: Durchgezogene Linie. In WIDTH kann die Breite der Linie angegeben werden.
- DASHED: Die Teilinie wird strichliert gezeichnet. Das Strichmuster wird durch die Tabelle LINE_DASH definiert. Das Strichmuster wird jeweils nach der Länge DLENGTH wiederholt.
- PATTERNED: Ein oder mehrere Symbole werden entlang der Teillinie gezeichnet. Das Symbolmuster wird jeweils nach der Länge PLENGTH wiederholt. Alle Symbole werden gemäss der aktuellen Neigung der Linie orientiert.
- Für jede Linie kann ein Start- (SSYMBOL) und ein Endsymbol (ESYMBOL) definiert werden. Die Symbole werden gemäss der Neigung am Anfang bzw. am Ende der Linie orientiert.

Für jede Liniensignatur kann mit CLIPSTYLE eine Clipsignatur angegeben werden. Linien sind orientiert. Die Liniensignatur wird jeweils vom Anfang bis Ende der Linie gezeichnet.

4.4.4 Eigenschaften von Flächensignaturen

Eine Flächensignatur (SURFACE_SYMBOLOGY) kann gefüllt oder nicht gefüllt dargestellt werden. Ausserdem kann die Signatur der Randlinie (BORDER) angegeben werden (die Orientierung der Randlinie ist so definiert, dass die Fläche immer rechts der Randlinie liegt). Folgende Füllmöglichkeiten werden angeboten (die allenfalls auch kombiniert werden können):

- FILL_COLOR: Die Fläche wird mit der Farbe vollständig ausgefüllt.
- HATCH_SYMB: Die Fläche wird schraffiert. Der Abstand zwischen den Schraffuren wird in HATCH_OFFSET angegeben. Die Schraffierung kann in der Interfaceprozedur P_SURFACE() über die Parameter HATCH_ANG und HATCH_ORG parametrisiert werden. Bemerkung: Für die Schraffurlinie kann eine beliebige Liniensignatur (HATCH_SYMB) angegeben werden. Durch die Angabe einer mit Symbolen gemusterten Liniensignatur kann die Füllung der Fläche mit Symbolen erreicht werden.

Der Clipbereich wird mit CLIP angegeben. Dabei kann man wählen ob die Fläche innerhalb (inside) oder ausserhalb des Objekts (outside) als Clipbereich gelten soll.

4.5. Benutzung des Darstellungsmodells

Damit das Darstellungsmodell in einem Datenmodell benutzt werden kann, muss das Datenmodell vom Darstellungsmodell durch Vererbung abgeleitet werden (MODEL <Name> EXTENSION OF

`ILI_PRESENTATION`, s.a. INTERLIS-2). Erst durch das Erben vom Darstellungsmodell sind alle notwendigen Definitionen des Darstellungsmodells im Datenmodell verfügbar. Bemerkung das Datenmodell muss nicht unbedingt Daten enthalten. Es kann z.B. ausschliesslich aus `VIEW's` bestehen (s.a. Kapitel 7).

5. Die Abbildungssprache

5.1 Einleitung

Mit der Abbildungssprache kann die Funktion F() definiert werden und somit die Verbindung zwischen Datenmodell und Darstellungsmodell hergestellt werden. Die Funktion F() erfüllt dabei folgende Aufgaben:

- Selektion der Daten im Datenmodell (*Was* wird dargestellt?).
- Angabe der Abbildung auf das Darstellungsmodell (*Wie* wird etwas dargestellt?).

Da für die meisten Datenmodelle mehre Darstellungsarten angegeben werden können, ist zudem die Angabe des Kontexts in welchem die Funktion F() arbeitet (z.B. unterschiedliche Darstellungen für 1:500, 1:10000, etc.) notwendig.

5.2 INTERLIS-Erweiterungen

Die Abbildungssprache wurde nicht als eigenständige Sprache definiert, sondern als Erweiterung des bestehenden (OO-)INTERLIS formuliert. Dies hat u. A. den Vorteil, dass alle Möglichkeiten von INTERLIS Version 2 [2] direkt für die Abbildungssprache zur Verfügung stehen. Folgende INTERLIS-Konstrukte werden zusätzlich eingeführt:

- Auf Modellstufe: Das `PRESENTATION`-Konstrukt. Mit dem `PRESENTATION`-Konstrukt können alle für ein Datenmodell möglichen Darstellungsarten definiert werden (Angabe des Kontext). Falls dies nötig ist, kann eine Darstellungsart zusätzlich parametrisiert werden.
- Auf Tabellenstufe: Der neue `MAPPING`-Teil (Was und Wie). Zusammen mit dem `IF`-Ausdruck (Was) und den Funktionsaufrufen (Wie) ermöglicht der `MAPPING`-Teil die vollständige Angabe der Funktion F().

5.2.1 Das PRESENTATION-Konstrukt

Das `PRESENTATION`-Konstrukt hat folgende Syntax (neue Syntaxteile sind fett dargestellt):

```
Datenmodell = 'MODEL' Modell-Name '='
                 { Modell-Wertebereichsdef
                 | Darstellungsdef
                 | Modell-Funktionsdef }
                 (* Thema *)
              'END' Modell-Name '.'

Darstellungsdef = 'PRESENTATION'
                     (* Darstellung ';' *)

Darstellung = Darstellungsname [ Parameterliste ]

Parameterliste = '(' Parameterdef ' { ',' Parameterdef } ')'

Parameterdef = Parametername ':' Wertebereich
```

Das `PRESENTATION`-Konstrukt hat folgende Bedeutung:

- Mit dem `PRESENTATION`-Konstrukt werden alle für ein Datenmodell verfügbaren Darstellungsarten deklariert.
- Nur für die im `PRESENTATION`-Konstrukt definierten Darstellungsarten können im `MAPPING`-Teil einer Tabelle verwendet werden.

- Der Darstellungsname wird später zusammen mit allfälligen Parameterwerten dem Treiberprogramm übergeben (s.a. Kapitel 6).
- Die in der Parameterliste definierten Parameter können im MAPPING-Teil als Variablen benutzt werden. Mit der Angabe von Parametern kann daher die Ausführung der Abbildungsfunktion F() gesteuert werden. Ein typischer Parameter ist z.B. die Plannummer für die Darstellung des Grundbuchplans 1:500.

5.2.2 Der MAPPING-Teil

Der MAPPING-Teil hat folgende Syntax (neue Syntaxteile sind fett dargestellt):

```
Tabelle = 'TABLE' Tabellen-Name '='
             Attribute
             Konsistenzdef
             Abbildungsregeln
          'END' Tabellen-Name ';'

SichtDef = 'VIEW' Sicht-Name 'ON' (Sichtweise | Erweiterungsdef) '='
             [ Sicht-Attribute ]
             Abbildungsregeln
           'END' Sicht-Name ';'

Abbildungsregeln =
   'MAPPING'
       (* Abbildungsregel *)

Abbildungsregeln =
   'MAPPING'
       (* Abbildungsregel *)

Abbildungsregel =
   Darstellungsname ':'
       Abbildungsanweisungen

Abbildungsanweisungen =
   { (Prozeduraufruf | IF_Anweisung) ';' }

IF_Anweisung =
   'IF' Logischer_Ausdruck 'THEN'
        Abbildungsanweisungen
   { 'ELSIF' Logischer_Ausdruck 'THEN'
        Abbildungsanweisungen }
   [ 'ELSE'
        Abbildungsanweisungen ]
   'END'
```

Der MAPPING-Teil hat folgende Bedeutung:

- Im MAPPING-Teil wird die Funktion F() für eine Tabelle oder einen View, für eine bestimmte Darstellungsart angegeben. Die möglichen Darstellungsarten müssen vorher unter PRESENTATION deklariert werden.
- Die Funktionsaufrufe im MAPPING-Teil erzeugen die entsprechenden Graphikobjekte im Darstellungsmodell. Z.B. erzeugt der Aufruf P_TEXT() ein Textobjekt. Die möglichen Funktionsaufrufe und ihre Parameter werden durch das Graphikinterface bestimmt.
- Bei der Erzeugung einer Darstellung durch ein Darstellungsprogramm werden für alle Objekte des Datenmodells die Darstellungsfunktionen für eine bestimmte Darstellungsart aufgerufen und ausgeführt. Die dabei erzeugten Graphikobjekte werden durch einen Treiber (s.a. Kapitel 6) in konkrete Graphik umgesetzt.

Der MAPPING-Teil einer Tabelle kann auch als spezielle INTERLIS-Methode verstanden werden. Bei der Erzeugung der Graphik 1:500 wird z.B. an alle Tabellenobjekte die Methode M500 (mit allfälligen Argumenten) gesendet. Die Tabellen antworten auf die Methode indem sie ihre Objekte entsprechend darstellen. Obwohl der MAPPING-Teil als spezielle Methode verstanden werden kann, wurde aus folgenden Gründen auf die Benutzung der INTERLIS-Methodensyntax verzichtet:

- Die MAPPING-Syntax ist handlicher und übersichtlicher. Der Benutzer muss weniger kodieren. Ausserdem ist sofort klar, dass es sich bei der Definition um eine Abbildungsdefinition und nicht um die Definition einer allgemeine Methode handelt.
- Der INTERLIS-Compiler kann die Syntax der IF_Anweisung überprüfen. In einer Methode könnte der IF-Teil nur als Erläuterung angegeben werden.
- Das PRESENTATION-Konstrukt ermöglicht die Definition aller für ein Modell zulässigen Abbildungen. Für INTERLIS-Methoden existiert kein entsprechendes Mittel.

Trotz der Argumente für einen separaten MAPPING-Teil, werden konkrete GIS-Systeme den MAPPING-Teil intern wahrscheinlich gleich wie INTERLIS-Methoden behandeln.

6. Von neutraler zu konkreter Graphik

6.1 Einleitung

Es bleibt noch die Frage zu klären wie aus der systemneutralen Graphik des Darstellungsmodells konkrete Graphik wird. Der Umsetzungsprozess von INTERLIS-Graphik in konkrete Graphik wird durch INTERLIS selbst nicht festgelegt. Trotzdem wollen wir hier angeben welche Schritte ein Treiberprogramm für die Erzeugung von konkreter Graphik durchführen muss.

6.2 Der Graphiktreiber

Durch einen Graphiktreiber werden die im Darstellungsmodell systemneutral beschriebene Graphik in ein konkretes Graphikformat umgesetzt. Der Umsetzungsprozess läuft nach folgendem Algorithmus ab:

```
Sortierung der Graphikobjekte nach Priorität.

FOR Priorität := 1 TO MAXPRIORITY DO

   FOR obj IN Graphikobjekte WHERE obj.PRIORITY=Priorität DO
     Freistellen (Clipping) des Graphikobjekts im Graphikbuffer.
   END_FOR

   FOR obj IN Graphikobjekte WHERE obj.PRIORITY=Priorität DO
     Symbolisieren des Graphikobjekts im Graphikbuffer.
   END_FOR

END_FOR

Ausgabe des Graphikbuffers auf ein Ausgabegerät.
```

Die einzelnen Schritte sind nachfolgend beschrieben.

6.2.1 Sortierung nach Priorität

Falls das Freistellen von Objekten unterstützt werden soll, müssen die Objekte vor dem Freistellen nach Priorität sortiert werden.

6.2.2 Freistellen des Graphikobjekts

Das Objekt wird gemäss den Angaben im der Signaturbibliothek im Graphikbuffer freigestellt aber noch nicht gezeichnet. Bemerkungen: Der Graphikbuffer ist ein temporärer Speicherbereich in dem die Graphik vor der Ausgabe auf das Ausgabegerät aufgebaut wird. Das Freistellen der Objekte ist für Rasterausgabegeräte (Bildschirm, Rasterplotter) einfach realisierbar, für Vektorausgabegeräte (Vektorplotter, Vektorgraphikdatei (DXF)) jedoch aufwendig.

6.2.3 Symbolisieren des Objekts

Vor der eigentlichen Ausgabe des Objekts wird die Symbologie gemäss den Signaturtabellen auf das Objekt angewendet. Z.B. muss die graphische Ausprägung der Symbole und die Signatur von Linien und Flächen erzeugt werden. Die so erzeugten Objekte werden im Graphikbuffer abgelegt.

6.2.4 Ausgabe des Graphikbuffers

Zum Schluss wird der gesamte Graphikbuffer an ein Ausgabegerät übermittelt (z.B Bildschirm, Plotter, Vektordatei, Rasterdatei, etc.). Da der Graphikbuffer normalerweise in Benutzerkoordinaten aufgebaut wird (d.h. in m), ist unmittelbar vor der Ausgabe eine Transformation der Objekte in Gerätekoordinaten notwendig.

6.3 Typen von Graphiktreibern

Es ist möglich und auch zulässig, dass ein bestimmtes Treiberprogramm nicht alle Möglichkeiten des INTERLIS-Darstellungsmodells unterstützt. Grob unterscheidet man folgende Typen von Graphiktreibern:

- **Typ-1**. Dieser Treibertyp unterstützt das INTERLIS-Darstellungsmodell (`ILI_PRESENTATION`) vollstängig. Ausserdem ist der Treiber in der Lage die Prioritäten und das Clipping richtig zu handhaben.
- **Typ-2**. Dieser Treibertyp unterstützt das INTERLIS-Darstellungsmodell (`ILI_PRESENTATION`) vollstängig. Der Treiber implementiert jedoch weder Clipping noch Prioritäten.
- **Typ-3**. Dieser Treibertyp unterstütz das INTERLIS-Darstellungsmodell nicht.

Die heute verfügbaren Darstellungs- bzw. Transferprogramme entsprechen Typ-3. Es ist zu erwarten, dass in Zukunft zuerst Typ-2 und später Typ-1 Treiber erhältlich sein werden. Die Hersteller von Treiberprogrammen müssen angeben, welche Teile des Darstellungsmodells von ihrem Treiber unterstützt werden.

7. Hinweise und Empfehlungen

7.1 Strukturierung der Graphik

Bei komplexen Datenmodellen mit vielen verschiedenen Darstellungsarten ist es oft wünschenswert die Graphik vollständig vom Datenmodell zu trennen. Für die Trennung eines Datenmodells in Kerndaten und Graphikdaten stellen wir jedoch keine neuen Mittel bereit. Die Möglichkeiten sind bereits in INTERLIS-2 vorhanden. Wir möchten hier die Trennung am bereits vorgestellten Beispiel aus Kaptitel 2 demonstrieren. Die im Kapitel 2 gestellte Aufgabe wird wie folgt erweitert:

- Das Datenmodell `Grundatensatz` soll in Kerndaten und Graphikdaten aufgeteilt werden.
- Die Kerndaten sollen im Modell `Grunddatensatz_Kern` abgelegt werden.
- Die Graphikdaten 1:1500 und 1:10'000 sollen in einem zusätzlichen Datenmodell `Grunddatensatz_Graphik` verwaltet werden.

Die Lösung wird in den folgenden Abschnitten erläutert. Bemerkung: Wir hätten auch verlangen können, dass die Graphikdaten 1:500 und die Graphikdaten 1:10'000 je in einem eigenen Datenmodell verwaltet werden. Im konkreten Fall muss der Modellierer entscheiden, welche Variante für die Lösung seiner Aufgabe am günstigsten ist.

7.1.1 Kerndatenmodell

Das Kerndatenmodell `Grunddatensatz_Kern` ist nachfolgend dargestellt:

```
MODEL Grunddatensatz_Kern

    DOMAIN
    !! Wertebereichsdefinitionen ...

    TOPIC Fixpunkte =
      TABLE LFP =
        Entstehung: OPTIONAL -> LFPNachfuehrung; !! Beziehung 1-mc
        Nummer: TEXT*12; !! Vergabe durch Landestopographie
        Geometrie: HKoord;
        Art: (LFP1, LFP2);
        Herkunft: OPTIONAL TEXT*30; !! Vergabe durch Kanton
      CONSTRAINT
        UNIQUE Nummer;
        UNIQUE Geometrie;
      END LFP;
    END Fixpunkte

  END Grunddatensatz_Kern.
```

Erläuterungen zum Kerndatenmodell:

- Das Modell `Grunddatensatz_Kern` enthält nur noch die Kerndaten ohne Graphikdaten.
- Alle darstellungsrelevanten Attribute wurden gegenüber der ersten Version gelöscht (z.B. `NumPos`, `NumOri`, `NumHAli`, `NumVAli`, `SymbolOri`).

7.1.2 Graphikmodell

Das Graphikmodell `Grunddatensatz_Graphik` ist nachfolgend angegeben (Erweiterungen betreffend Graphik sind fett dargestellt):

```
MODEL Grunddatensatz_Graphik

    EXTENSION OF ILI_PRESENTATION
    DEPENDING ON Grunddatensatz_Kern

    PRESENTATION !! mögliche Darstellungsarten
      M500,M10000;

    TOPIC Fixpunkte =
      TABLE LFP =
        Objekt: -> Grunddatensatz.Fixpunkte.LFP;
        NumPos: LKoord;
        NumOri: OPTIONAL SchriftOri; !! Default: 100.0
        NumHAli: OPTIONAL HALIGNMENT; !! Default: Center
        NumVAli: OPTIONAL VALIGNMENT; !! Default: Half
        SymbolOri: OPTIONAL SchriftOri; !! Default: 0.0
      MAPPING !! Definition von F() für M500 und M10000
        M500:
          IF Art = LFP1 THEN
            P_SYMBOL(Objekt.Geometrie,SymbolOri,"Fix500_LFP1",1000);
          ELSIF Art = LPF2 THEN
            P_SYMBOL(Objekt.Geometrie,SymbolOri,"Fix500_LFP2",1000);
          END;
          P_TEXT(Objekt.Nummer,NumPos,NumOri,NumHAli,NumVAli,
          "Fix500_LFP",1000);
        M10000:
          P_SYMBOL(Objekt.Geometrie,SymbolOri,"Fix10000_LFP",1000);
      END LFP;
    END Fixpunkte

  END Grunddatensatz_Graphik.
```

Erläuterungen zum Graphikmodell:

- Das Graphikmodell muss vom Darstellungsmodell `ILI_PRESENTATION` mit `EXTENSION OF` abgeleitet werden.
- Das Graphikmodell enthält nur die Graphikdaten (`NumPos`, `NumOri`, `SymbolOri` etc.) und die Definition der Darstellungsfunktion F().
- Da sich das Graphikdatenmodell auf Objekte des Kerndatenmodells bezieht, wird eine entsprechende Beziehung zwischen Graphikdatenmodell und Kerndatenmodell mit `DEPENDING ON` eingeführt.

7.1.3 Variante

Wenn ein bestehendes Datenmodell (z.B. der Grunddatensatz) nicht verändert werden darf, kommt die in 7.1.1 und 7.1.2 präsentierte Lösung nicht in Frage. Man kann sich aber wie folgt behelfen (Änderungen gegenüber der letzten Lösung sind fett dargestellt):

```
MODEL Grunddatensatz_Graphik

    EXTENSION OF ILI_PRESENTATION
    DEPENDING ON Grunddatensatz

    PRESENTATION !! mögliche Darstellungsarten
      M500,M10000;

    TOPIC Fixpunkte =
      VIEW LFP ON Grunddatensatz.Fixpunkte.LFP =
      MAPPING !! Definition von F() für M500 und M10000
        M500:
          IF Art = LFP1 THEN
            P_SYMBOL(Geometrie,SymbolOri,"Fix500_LFP1",1000);
```

```
            ELSIF Art = LPF2 THEN
              P_SYMBOL(Geometrie,SymbolOri,"Fix500_LFP2",1000);
            END;
            P_TEXT(Nummer,NumPos,NumOri,NumHAli,NumVAli,
            "Fix500_LFP",1000);
          M10000:
            P_SYMBOL(Geometrie,SymbolOri,"Fix10000_LFP",1000);
        END LFP;
      END Fixpunkte

  END Grunddatensatz_Graphik.
```

Erläuterungen zur Lösungsvariante:

- Das Datenmodell Grunddatensatz bleibt unverändert, d.h. die Graphikdaten (hier `NumPos`, `NumOri`, `NumVAli`, `NumHAli`, `SymbolOri`) werden nicht separat im Graphikmodell gespeichert.
- Anstelle von Tabellen verwenden wir in dieser Variante Views. Die Funktion F() kann analog zu Tabellen auch für Views deklariert werden. Durch die Verwendung von Views erreichen wir, dass keine neuen Graphikobjekte angelegt werden müssen.

7.2 Konsequenzen für den Grunddatensatz

In der aktuellen Version des Grunddatensatz (AV93) bestehen zwei Problemkreise:

1. Die Graphikdaten wurden mit den Kerndaten des Grunddatensatz vermischt (Strukturproblem).
2. Die Darstellungsfunktion und die Signaturbibliothek sind im Grunddatensatz nicht vorhanden (fehlende Graphikdefinition).

Die Behebung von Punkt 1 führt lediglich zu einer Verbesserung der Modellstruktur des Grunddatensatz. Eine rasche Verbesserung von Punkt 1 ist daher nicht zwingend notwendig. An der Implementierung von Punkt 2 sind jedoch bereits heute viele Anwender interessiert. Wir empfehlen daher folgendes Vorgehen:

- Das Graphikmodell für den Grunddatensatz sollte möglichst bald in INTERLIS formuliert und die Signaturbibliothek als INTERLIS-Datensatz erfasst werden. Durch die Möglichkeiten von INTERLIS V2 kann das Graphikmodell als eigenständiges Modell implementiert werden (s.a. 7.1.3). Das Modell `Grunddatensatz` kann dadurch vorläufig in seiner aktuellen Form bestehen bleiben.
- Bei einer allfälligen Überarbeitung des Grunddatensatz sollte die Strukturierung des Grunddatensatz in Hinblick auf die Graphik verbessert werden.

Anhang

A1 Literaturverzeichnis

[1] Eidg. Vermessungsdirektion. INTERLIS ein Daten-Austausch-Mechanismus für Land-Informations-Systeme, Oktober 1991

[2] Eidg. Vermessungsdirektion. INTERLIS ein Datenaustausch-Mechanismus für Land-Informationssysteme. Draft Version 2, Januar 1998

[3] Stefan Keller, Eidg. Vermessungsdirektion. RFC-1011: Ein Darstellungsmodell und eine Abbildungssprache für INTERLIS, August 1997

[4] Eidg. Justizdepartement. Datensatz der amtl. Vermessung, 1993

[5] Josef Dorfschmid, ADASYS AG. Expertise betreffend OO-INTERLIS -einer objektorientierten Erweiterung von INTERLIS, Januar 1998

[6] Michael Germann, Josef Dorfschmid. Inkrementelle Nachlieferung mit INTERLIS, Februar 1997

A2 Syntax der Abbildungssprache

A2.1 PRESENTATION-Konstrukt

Das PRESENTATION-Konstrukt hat folgende Syntax (neue Syntaxteile sind fett dargestellt):

```
Datenmodell = 'MODEL' Modell-Name '='
                 { Modell-Wertebereichsdef
                 | Darstellungsdef
                 | Modell-Prozedurdef }
                 (* Thema *)
              'END' Modell-Name '.'

Darstellungsdef = 'PRESENTATION'
                     (* Darstellung ';' *)

Darstellung = Darstellungsname [ Parameterliste ]

Parameterliste = '(' Parameterdef ' { ',' Parameterdef } ')'

Parameterdef = Parametername ':' Wertebereich
```

A2.2 MAPPING-Teil

Der MAPPING-Teil hat folgende Syntax (neue Syntaxteile sind fett dargestellt):

```
Tabelle = 'TABLE' Tabellen-Name '='
             Attribute
             Konsistenzdef
             Abbildungsregeln
          'END' Tabellen-Name ';'

SichtDef = 'VIEW' Sicht-Name 'ON' (Sichtweise | Erweiterungsdef) '='
             [ Sicht-Attribute ]
             Abbildungsregeln
           'END' Sicht-Name ';'

Abbildungsregeln =
   'MAPPING'
       (* Abbildungsregel *)

Abbildungsregel =
   Darstellungsname ':'
       Abbildungsanweisungen

Abbildungsanweisungen =
   { (Prozeduraufruf | IF_Anweisung) ';' }

IF_Anweisung =
   'IF' Logischer_Ausdruck 'THEN'
        Abbildungsanweisungen
   { 'ELSIF' Logischer_Ausdruck 'THEN'
        Abbildungsanweisungen }
   [ 'ELSE'
        Abbildungsanweisungen ]
   'END'
```

A3 Darstellungsmodell in INTERLIS

```
!! INTERLIS-Darstellungsmodell
!!
!! Version 1.0
!!
!! Stand 22.5.1998

MODEL ILI_PRESENTATION

   DOMAIN

      !! Alle Groessenangaben von Signaturen in mm

      P_LINE     = POLYLINE WITH (STRAIGHTS, ARCS) VERTEX COORD2;
      P_SURF     = SURFACE WITH (STRAIGHTS, ARCS) VERTEX COORD2;
      P_INT      = [-2000000000 .. 2000000000];
      P_FLOAT    = [-2000000000.000 .. 2000000000.000];
      P_ANGLE    = DEGREES 0.000 360.000;
      P_PRIORITY = [0 .. 9999];
      P_BOOLEAN  = (no,yes);

      P_FONTID = TEXT*30;
      P_LINEID = TEXT*30;
      P_TEXTID = TEXT*30;
      P_SYMBID = TEXT*30;
      P_SURFID = TEXT*30;

   !! Allgemeine Graphikprozeduren

   PROCEDURE

      OFFSET2(p:COORD2;y:P_FLOAT;x:P_FLOAT):COORD2 =
         // Addiert den Offset y/x zum Punkt p //;
      OFFSET3(p:COORD3;y:P_FLOAT;x:P_FLOAT;z:P_FLOAT):COORD3 =
         // Addiert den Offset y/x/z zum Punkt p //;
      SUBSTRING(t:TEXT;index1:P_INT;index2:P_INT):TEXT =
         // Liefert von t den Teilstring von index1 bis index2.
            Bemerkung: Der Index des 1. Buchstabens ist 1 //;
      FORMAT(format:TEXT; ...):TEXT =
         // Liefert einen gemaess format formatierten Text
            (analog zur C-Funktion printf()) //;

   !! Graphikinterface

   PROCEDURE

      P_TEXT (
         TXT       : TEXT;
         GEOMETRY  : COORD2;
         ROTATION  : DEGREES,OPTIONAL; !! Default 0.0
         HALI      : HALIGNMENT,OPTIONAL; !! Default Center
         VALI      : VALIGNMENT,OPTIONAL; !! Default Half
         SYMBOLOGY : P_TEXTID;
         PRIORITY  : P_PRIORITY
      ) = // Erzeugt ein Textobjekt //;

      P_SYMBOL (
         GEOMETRY  : COORD2;
         ROTATION  : DEGREES,OPTIONAL ; !! Default 0.0
         SYMBOLOGY : P_SYMBID;
         PRIORITY  : P_PRIORITY
      ) = // Erzeug ein Symbolobjekt //

      P_LINE (
         GEOMETRY  : P_LINE;
         SYMBOLOGY : P_LINEID;
         PRIORITY  : P_PRIORITY
      ) = // Erzeugt eine Linienobjekt //;

      P_SURFACE (
         GEOMETRY    : P_SURF;
         HATCH_ANG   : DEGREES,OPTIONAL; !! Default 0.0
         HATCH_ORG   : COORD2,OPTIONAL; !! Default 0.0/0.0
         SYMBOLOGY   : P_SURFID;
         PRIORITY    : P_PRIORITY
      ) = // Erzeugt ein Flaechenobjekt //;
```

```
TOPIC SYMBOLOGY =

   !! Farbtabelle

   TABLE COLORS =
      NAME : TEXT*30;
     R    : [0 .. 255];
     G    : [0 .. 255];
     B    : [0 .. 255];
   CONSTRAINT
      UNIQUE NAME;
   END COLORS;

   !! Font- und Symbolbibliothek

   TABLE FONTLIB =
      NAME        : P_FONTID;
      TYPE        : (symbol,text);
      BOTTOM_BASE : P_FLOAT,OPTIONAL; !! nur fuer Textfont
   END FONTLIB;

   TABLE FONTSYMBOL =
      OBJECT  : -> FONTLIB;
      ASCII   : [0 .. 255],OPTIONAL; !! nur fuer Textzeichen
      SPACING : P_FLOAT,OPTIONAL; !! nur fuer Textzeichen
   END FONTSYMBOL;

   TABLE FONTSYMBOL_LINE =
      OBJECT   : -> FONTSYMBOL;
      COLOR    : -> COLORS;
      WIDTH    : P_FLOAT;
      GEOMETRY : P_LINE;
   END FONTSYMBOL_LINE;

   TABLE FONTSYMBOL_SURFACE =
      OBJECT    : -> FONTSYMBOL;
      FILLCOLOR : -> COLORS;
      GEOMETRY  : P_SURF;
   END FONTSYMBOL_SURFACE;

   !! Linestylebibliothek

   TABLE LINESTYLE =
      START_SYMBOL : -> FONTSYMBOL,OPTIONAL;
      END_SYMBOL   : -> FONTSYMBOL,OPTIONAL;
   END LINESTYLE;

   TABLE LINESTYLE_SOLID =
      OBJECT : -> LINESTYLE;
      OFFSET : P_FLOAT;
     COLOR  : -> COLORS;
      WIDTH  : P_FLOAT;
   END LINESTYLE_SOLID;

   TABLE LINESTYLE_DASHED =
      OBJECT  : -> LINESTYLE;
     OFFSET  : P_FLOAT;
      COLOR   : -> COLORS;
      WIDTH   : P_FLOAT;
      DLENGTH : P_FLOAT;
   END LINESTYLE_DASHED;

   TABLE LINE_DASH =
      OBJECT : -> LINESTYLE_DASHED;
      SDIST  : P_FLOAT;
      EDIST  : P_FLOAT;
   CONSTRAINT
      SDIST <= EDIST;
   END LINE_DASH;

   TABLE LINESTYLE_PATTERN =
      OBJECT  : -> LINESTYLE;
     OFFSET  : P_FLOAT;
      PLENGTH : P_FLOAT;
   END LINESTYLE_PATTERN;

   TABLE LINE_PATTERN_SYMBOL =
      OBJECT   : -> LINESTYLE_PATTERN;
      SYMBOL   : -> FONTSYMBOL;
      SCALE    : P_FLOAT;
```

```
         COLOR    : -> COLORS,OPTIONAL;
         WIDTH    : P_FLOAT,OPTIONAL;
          DIST     : P_FLOAT;
          OFFSET   : P_FLOAT;
          ROTATION : P_ANGLE;
      END LINE_PATTERN_SYMBOL;

      !! Textsignatur

      TABLE TEXT_SYMBOLOGY =
         NAME       : P_TEXTID;
         TYPE       : (internal,external);
         FONTNAME   : P_FONTID; !! fuer internal Name eines Fonts aus FONTLIB
         HIGHT      : P_FLOAT;
         SLANTED    : P_BOOLEAN;
         UNDERLINED : P_BOOLEAN;
         STRIKED    : P_BOOLEAN;
          COLOR      : -> COLORS,OPTIONAL;
         WIDTH      : P_FLOAT,OPTIONAL;
          CLIPBOX    : P_FLOAT,OPTIONAL;
          CLIPDIST   : P_FLOAT,OPTIONAL;
      CONSTRAINT
         UNIQUE NAME;
      END TEXT_SYMBOLOGY;

      !! Symbolsignatur

      TABLE SYMBOL_SYMBOLOGY =
         NAME       : P_SYMBID;
         SYMBOL     : -> FONTSYMBOL;
         SCALE      : P_FLOAT;
         COLOR      : -> COLORS,OPTIONAL;
         WIDTH      : P_FLOAT,OPTIONAL;
          CLIPSYMBOL : -> FONTSYMBOL,OPTIONAL;
         CLIPSCALE  : P_FLOAT,OPTIONAL;
      CONSTRAINT
         UNIQUE NAME;
      END SYMBOL_SYMBOLOGY;

      !! Liniensignatur

      TABLE LINE_SYMBOLOGY =
         NAME      : P_LINEID;
         LINESTYLE : -> LINESTYLE;
         COLOR     : -> COLORS,OPTIONAL;
         WIDTH     : P_FLOAT,OPTIONAL;
         CLIPSTYLE : -> LINESTYLE,OPTIONAL;
      CONSTRAINT
         UNIQUE NAME;
      END LINE_SYMBOLOGY;

      !! Flaechensignatur

      TABLE SURFACE_SYMBOLOGY =
         NAME         : P_SURFID;
         FILLCOLOR    : -> COLORS,OPTIONAL;
         BORDER       : -> LINE_SYMBOLOGY,OPTIONAL;
         CLIP         : (inside,outside),OPTIONAL;
         HATCH_SYMB   : -> LINE_SYMBOLOGY,OPTIONAL;
         HATCH_OFFSET : P_FLOAT,OPTIONAL;
      CONSTRAINT
         UNIQUE NAME;
      END SURFACE_SYMBOLOGY;

   END SYMBOLOGY.

END ILI_PRESENTATION.
```