Praktikum Sprachimplementierung mit Werkzeugen

Uwe Kastens

WS 1999/2000

Ziele:

Die Teilnehmer sollen lernen

- **anwendungsspezifische Sprachen** zu entwerfen**,**
- **Implementierungsaufgaben** dafür zu erkennen, einzuordnen und zu beschreiben,
- **Spezifikationstechniken** für die zentralen Werkzeuge von Eli anzuwenden,
- das **Eli-System wirksam einzusetzen** und zu bedienen.

Sie sollen ein **selbstdefiniertes Anwendungsprojekt** durchführen können.

Beispiele für Projektthemen

- **Generator für Datenstrukturen**
- **Literatur-Report-Generator**
- **Steuersprache für Meßrechner**
- **Makrosprache zum Test von Instrumenten**
- **Programmstrukturen nach HTML übersetzt**
- **Spracherweiterungen (z. B. Occam)**
- **Notenschrift-Generator**
- **Interpretierer für Entscheidungstabellen**
- **Tischrechner**
- **Lindenmeyer-Grammatiken zur Bilderzeugung**

Sprachen in der Software-Entwicklung

Spezifikations- und Modellierungssprachen:

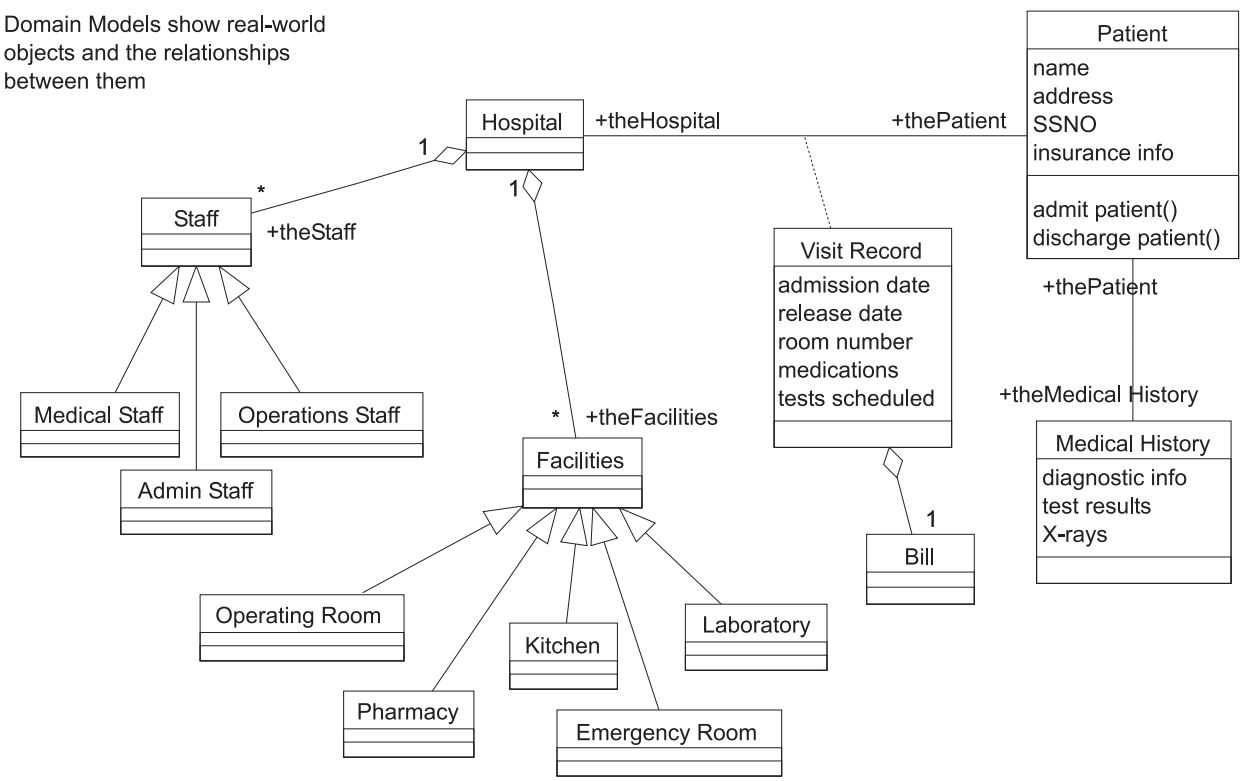
SDL Specification and Description Language (CCITT):

```
block Dialogue;
  signal
    Money, Release, Change, Accept, Avail, Unavail, Price,
    Showtxt, Choice, Done, Flushed, Close, Filled;
  process Coins referenced;
  process Control referenced;
  process Viewpoint referenced;
  signalroute Plop
    from env to Coins
      with Coin_10, Coin_50, Coin_100, Coin_x;
  signalroute Pong
    from Coins to env
      with Coin_10, Coin_50, Coin_100, Coin_x;
  signalroute Cash
    from Coins to Control
      with Money, Avail, Unavail, Flushed, Filled;
    from Control to Coins
      with Accept, Release, Change, Close;

  ...

  connect Pay and Plop;
  connect Flush and Pong;
endblock Dialogue;
```

UML Unified Modeling Language:



Sprachen in der Software-Entwicklung

Domain-Specific Language (DSL):
Sprache zugeschnitten auf ein spezielles Anwendungsgebiet

Application Generators:
Implementierung einer DSL durch einen Programmgenerator

Beispiele:

- **Simulation mechatronischer Regelsysteme**
- **Robotersteuerung**
- **Meßdatenerfassung**
- **Steuerung von Prüfständen für Kfz-Instrumente**
- **Literatur-Report-Generator:**

```
string name =   InString "Which author?";
int since =     InInt "Since which year?";
int cnt = 0;

"\nPapers of ", name, " since ", since, ":\n";
[ SELECT name <= Author && since <= Year;
  cnt = cnt + 1;
  Year, "\t", Title, "\n";
]
"\n", name, " published ", cnt, "papers.\n";
```

aus U. Kastens: Construction of Application Generators Using Eli, Workshop on Compiler Techniques for Application Domain Languages ..., Linköping, April 1996

zum Thema:

C.W. Krueger: Software Reuse, ACM Computing Surveys 24, June 1992

Conference on DSL (USENIX), Santa Babara, Oct. 1997

ACM SIGPLAN Workshop on DSL (POPL), Paris, Jan 1997

Sprachen in der Software-Entwicklung

Programmiersprachen als Quellsprachen:

- **Programmanalyse**
  Aufrufgraphen, Ablaufgraphen, Variablenabhängigkeiten
  z. B. für Year-2000-Problem
- **Strukturerkennung**
  z. B. für Reengineering

Programmiersprachen als Zielsprachen generiert aus

- **Spezifikationen (SDL, OMT, UML)**
- **graphischen Strukturmodellen**
- **DSL, Application Generator**

=> Übersetzeraufgabe: Source-to-Source-Übersetzung

Praktikumsplan

Termin	Vorlesung	Tutorium	Vor- u. Nacharbeit
1	Eli: Aufgabenzerlegung		Novice-Guide
2		Novice-Guide durcharbeiten	
3	konkrete u. abstrakte Syntax		
	Berechnungen im Baum		Computations in Trees
4		Programmanalyse I	
5	Modulbenutzung, Namensanalyse,		
	Eigenschaften von Objekten		
6		Programmanalyse II	
			Core-Spezifikation
7	Minisprache Core vorstellen		
		Minisprache Core erweitern	
8	Zieltext erzeugen		
		Core nach C übersetzen	
9	Projekt: Struktur-Generator vorstellen		
		Projekt: Struktur-Generator bearbeiten	
10 - 14	Eigene Projekte definieren		
		und bearbeiten	
15	Vorstellung der Projektergebnisse		

Compilation Tasks for Source-to-Source Translation

text in a domain specific language

Structuring	**Lexical analysis**	**Scanning** **Conversion**
	Syntactic analysis	**Parsing** **Tree construction**
Translation	**Semantic analysis**	**Name analysis** **Type analysis**
	Transformation	**Data mapping** **Action mapping**
Encoding	Code generation	Execution-order Register allocation Instruction selection
	Assembly	Internal addressing External addressing Instruction encoding

Waite, Carter: An Introduction to Compiler Construction, Harper Collins, 1992

application program (HLL)

Eli provides solution methods

generating tools

reusable precoined solutions for common tasks

extensible by domain specific routines

Lexical analysis	**input processing** **token sets** **scanner generator** **encoding & conversion**
Syntactic analysis	**parser generator** **tree construction generator**
Semantic analysis **attribute evaluator**	**name analysis modules** **type analysis modules** **definition table generator**
generator **Transformation**	**pattern based text generator**

Integrated tool set Eli

Eli knows how to

- drive the generators
- obtain the precoined solutions
- compose the translator
- relate the user specifications

User specifications for source-to-source processors:

	token descriptions select non-literal tokens
	context-free grammar
associate roles to tree grammar compute in tree contexts	use name analysis module use type analysis module specify object properties
compose target structure	specify target text patterns

Examples for Specification Fragments

- context-free grammar

```
OutputItems:    OutputItem // ','.
```

- select non-literal token specification

```
Ident:          C_IDENTIFIER
```

- select name analysis module

```
$/Name/CScope.gnrc:inst
```

- associate roles to tree grammar symbols

```
SYMBOL DefIdent INHERITS IdDefScope END;
SYMBOL UseIdent INHERITS IdUseEnv END;
```

- specify object properties

```
Kind: DefTableKey;
```

- compute in tree context

```
RULE: Statement ::= UseIdent '=' Expr ';'
COMPUTE
IF (NE (GetKind (UseIdent.Key, VariableKind),
        VariableKind),
  message (ERROR, "variable required",
           0, COORDREF));
END;
```

- specify target text patterns

```
InpCall: "(" $1 string "(" $2 string "))"
```

- compose target structure

```
RULE: Expr ::= 'InString' StringLit COMPUTE
  Expr.CCode =
    PTGInpCall ("InString",
                StringTable(StringLit));
END;
```

FunnelWeb: Spezifikationsfragmente zusammenfassen und dokumentieren

```
@=~
Umschaltzeichen von @ auf Tilde umgestellt
~p maximum_input_line_length = infinity
beliebig lange Zeilen
Die Ausgabedatei ArrayType.con erzeugen:
~O~<ArrayType.con~>~{
Type: 'array' Index 'of' Type.
~}
Noch eine Ausgabedatei:
~O~<ArrayAccess.con~>~{
Variable: Variable '[' Expr ']'.
~}
Eli fügt alle .con Dateien zu einer zusammen.
Dies kann man auch mit fw-Makros tun:
Produktionen für Array-Typen:
~$~<Array type~>==~{
Type: 'array' Index 'of' Type.
~}
Produktionen für Array-Zugriffe:
~$~<Array access~>==~{
Variable: Variable '[' Expr ']'.
~}
Die zusammengesetzte Ausgabedatei:
~O~<Array.con~>~{
~<Array type~>
~<Array access~>
~}
Der Eli-Ausdruck X.fw : fwGen liefert eine
Directory mit den Ausgabedateien.
```