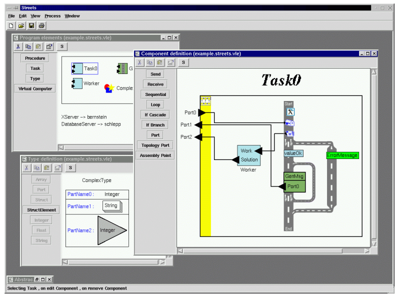
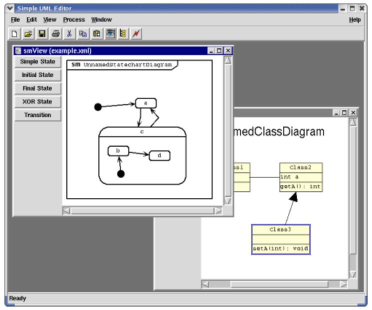
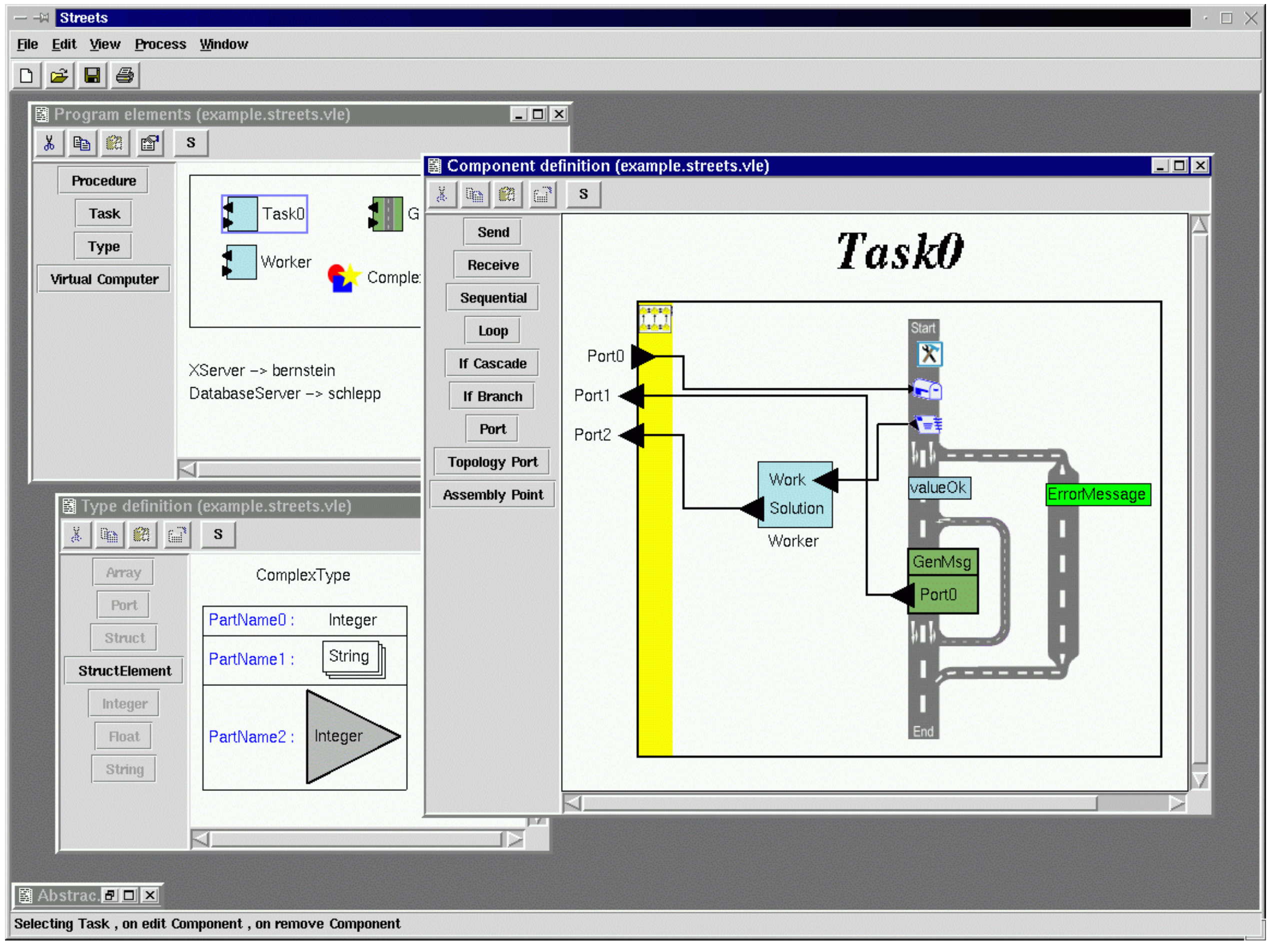
Domain-Specific Visual Languages: Design and Implemenation



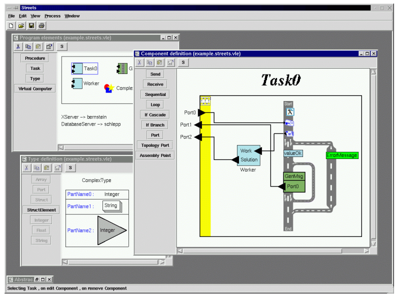


Uwe Kastens, 6. July 2007
CoRTA

Streets: A Visual Programming Language



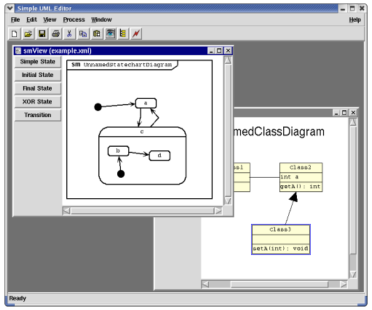
Outline

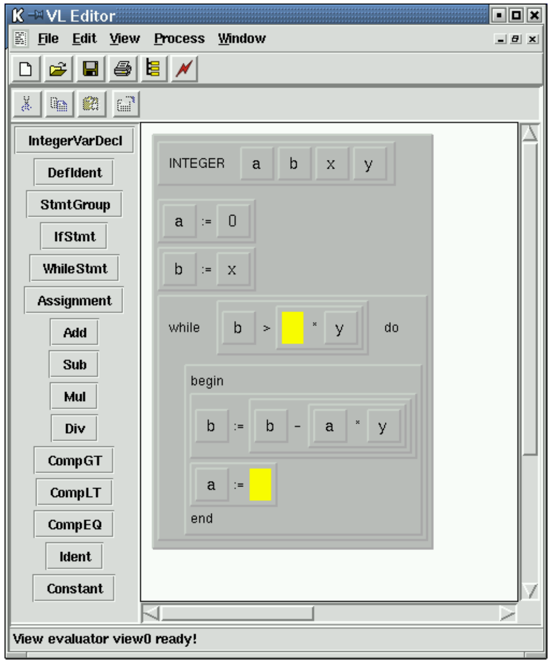


1. **What are visual languages?**
2. Domain-specific visual languages
3. Ingredients for Language design
4. A Development Environment for Visual Languages
5. Pattern-Based Specifications in DEViL

Visual Language

Formal language:
Set of sentences, each
- a sequence of tokens
- composed according to a syntax
- obeys static constraints
- has certain semantics

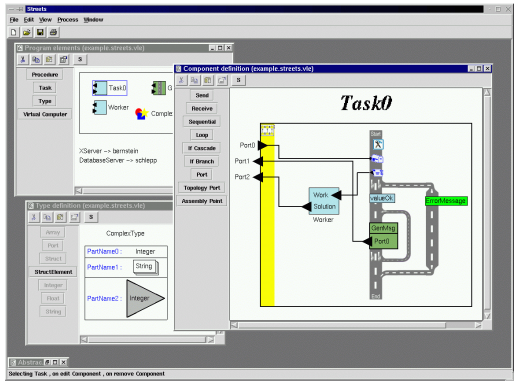


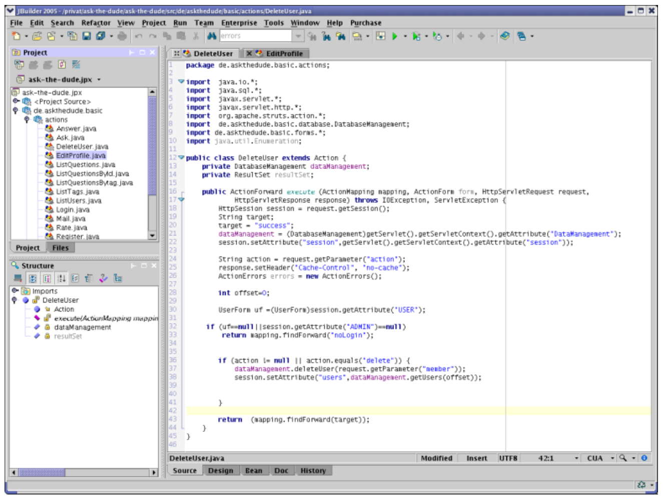


[Schiffer 1998]
Visual:
At least one essential property of an object is recognized by **visual perception**.

Visual language:
has a visual syntax or visual semantics

Visual Language vs. Graphical User Interface




JBuilder 2005 - /privat/ask-the-dude/ask-the-dude/src/de/askthedude/basic/actions/DeleteUser.java
File Edit Search Refactor View Project Run Team Enterprise Tools Window Help Purchase
Project
ask-the-dude.jpx
DeleteUser
EditProfile
Structure
DeleteUser.java
Modified
Insert
UTF8
42:1
CUA
Source
Design
Bean
Doc
History

```
package de.askthedude.basic.actions;

import  java.io.*;
import  java.sql.*;
import  javax.servlet.*;
import  javax.servlet.http.*;
import  org.apache.struts.action.*;
import  de.askthedude.basic.database.DatabaseManagement;
import de.askthedude.basic.forms.*;
import java.util.Enumeration;

public class DeleteUser extends Action {
    private DatabaseManagement dataManagement;
    private ResultSet resultSet;

    public ActionForward execute (ActionMapping mapping, ActionForm form, HttpServletRequest request,
            HttpServletResponse response) throws IOException, ServletException {
        HttpSession session = request.getSession();
        String target;
        target = "success";
        dataManagement = (DatabaseManagement)getServlet().getServletContext().getAttribute("DataManagement");
        session.setAttribute("session",getServlet().getServletContext().getAttribute("session"));

        String action = request.getParameter("action");
        response.setHeader("Cache-Control", "no-cache");
        ActionErrors errors = new ActionErrors();

        int offset=0;

        UserForm uf =(UserForm)session.getAttribute("USER");

     if (uf==null||session.getAttribute("ADMIN")==null)
      return mapping.findForward("noLogin");


        if (action != null || action.equals("delete")) {
            dataManagement.deleteUser(request.getParameter("member"));
            session.setAttribute("users",dataManagement.getUsers(offset));


        }

        return  (mapping.findForward(target));
    }
}
```

User
- **composes** a sentence

Designer
- designs language **constructs** and composition **rules**

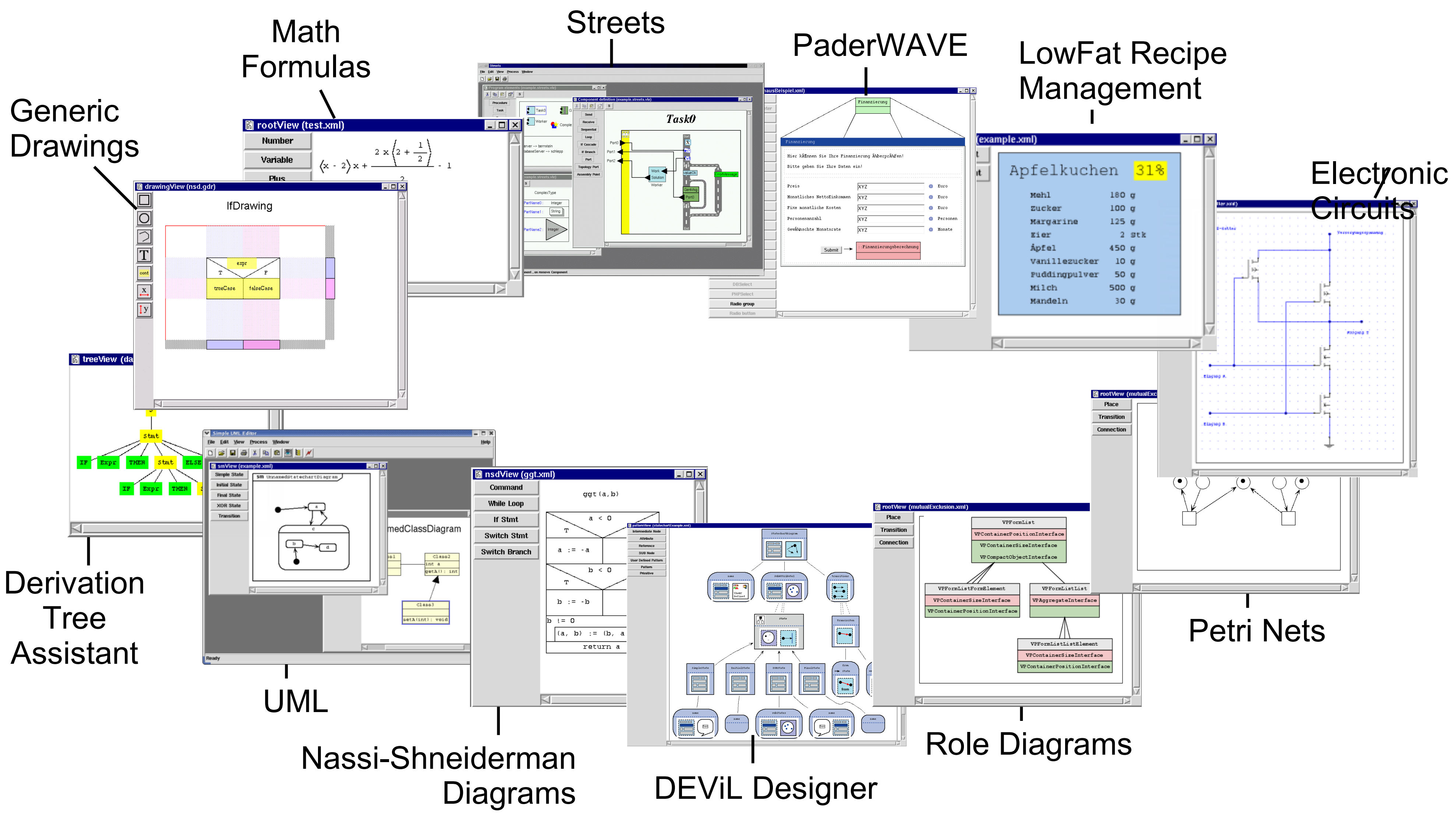
Tool
- provides language design support

User
- **interacts** with a GUI

Designer
- **composes** a GUI

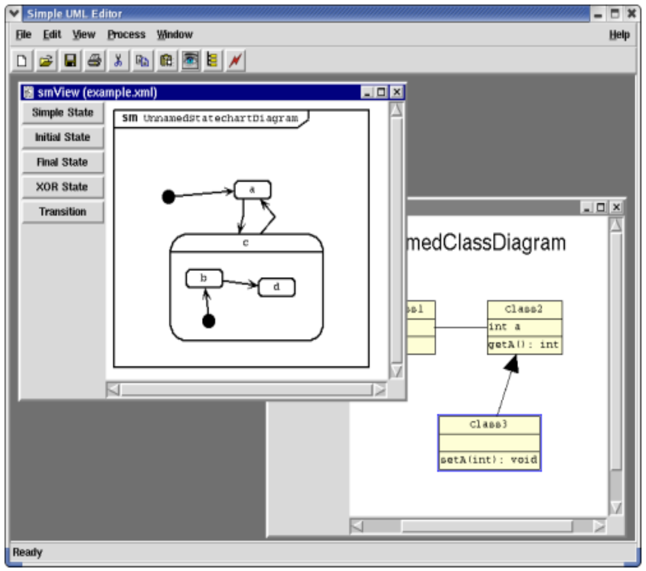
Tool
- provides GUI **components** and composition **rules**

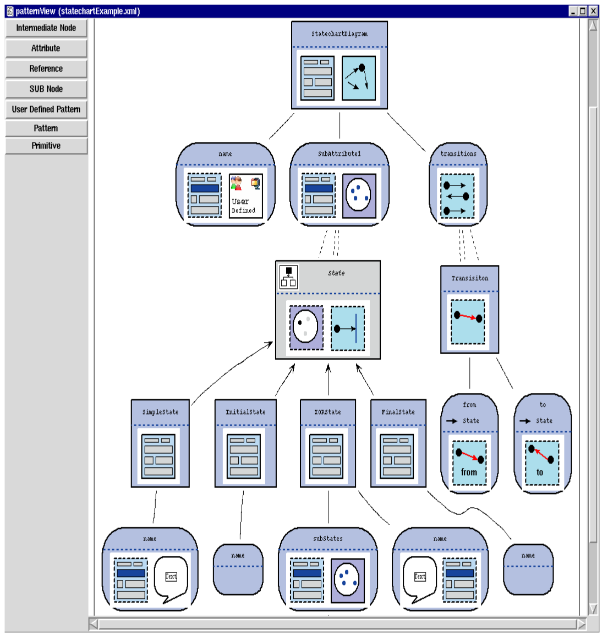
Variety of Language Constructs and Styles

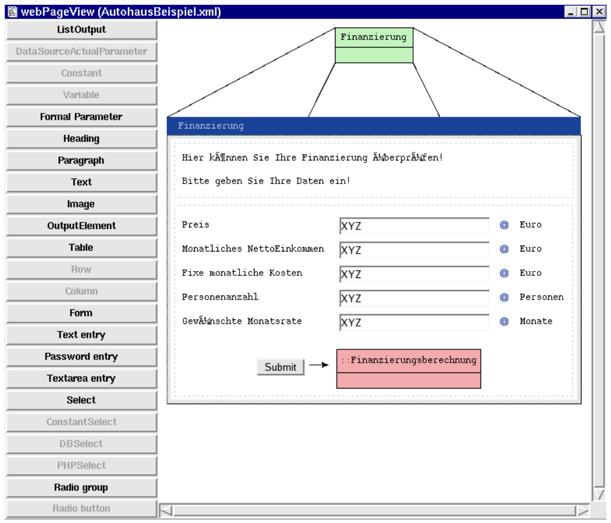


Classification of Visual Representations

Diagrammatic **Iconic** **Table-, Form-based**







Finanzierung

Hier kÃ¶nnen Sie Ihre Finanzierung Ã¼berprÃ¼fen!

Bitte geben Sie Ihre Daten ein!

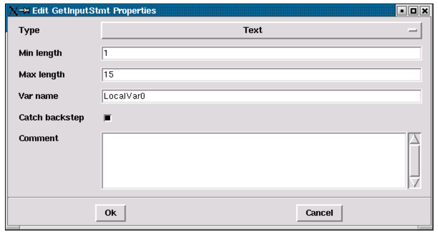
Preis	XYZ	Euro
Monatliches NettoEinkommen	XYZ	Euro
Fixe monatliche Kosten	XYZ	Euro
Personenanzahl	XYZ	Personen
GewÃ¼nschte Monatsrate	XYZ	Monate

According [Shu 1982]

Classification dimensions like paradigms (imperative, object-oriented, ...) apply for visual languages as well as for textual ones

Pros for Visual Languages

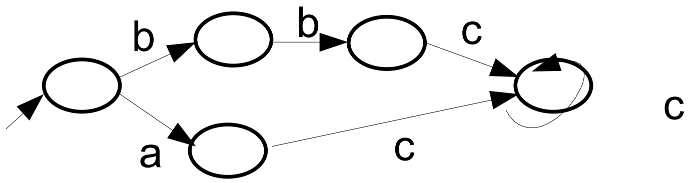
- **Intuitively usable**
  even complex language constructs


Edit GetInputStmt Properties

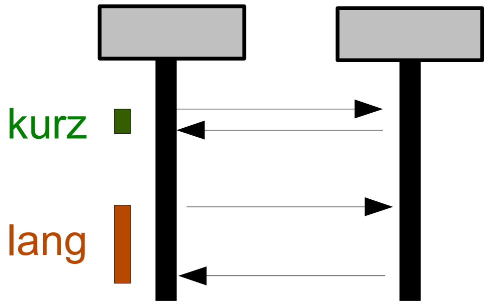
Type	Text
Min length	1
Max length	15
Var name	LocalVar0
Catch backstep	■
Comment	

Ok Cancel

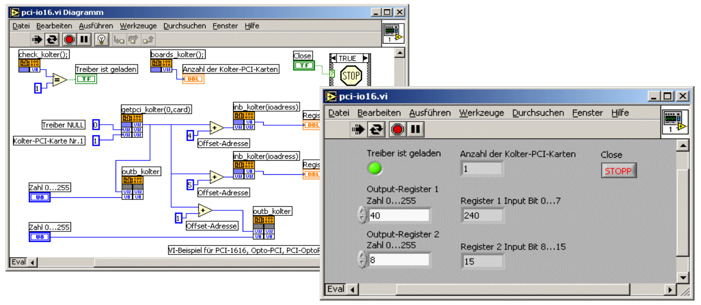
- Structures and relations can be represented for **easy recognition**



- **Quantitative** properties are representable



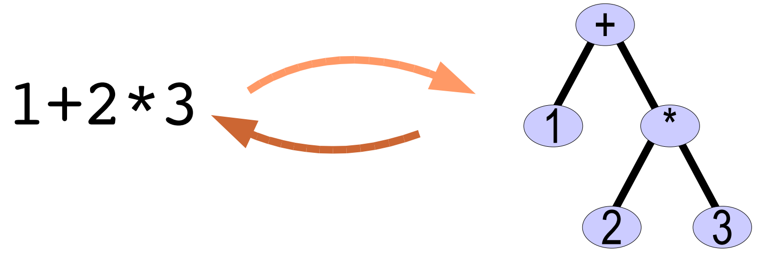
- **Different views** can show different aspects at the same time

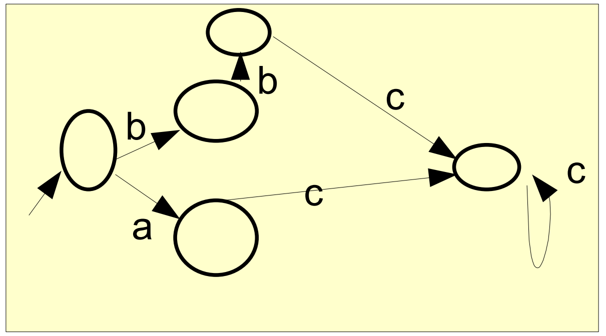


Treiber ist geladen
Anzahl der Kolter-PCI-Karten: 1
Close
Output-Register 1 Zahl 0...255: 40
Register 1 Input Bit 0...7: 240
Output-Register 2 Zahl 0...255: 8
Register 2 Input Bit 8...15: 15

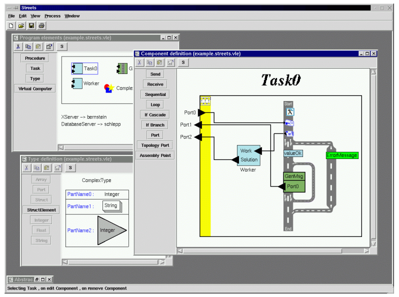
Cons for Visual Languages

- **Structural editing** may be cumbersome
- **Maintaining** a visual layout may be cumbersome
- **Space needed**
- **General purpose graphical editor**: Drawing may be cumbersome
- **Language specific editor** may not be available





Outline



1. What are visual languages?
2. **Domain-specific visual languages**
3. Ingredients for Language design
4. A Development Environment for Visual Languages
5. Pattern-Based Specifications in DEViL

Domain-Specific vs. General Purpose

A **task**: „Implement a program to store collections of words, that describe animals“

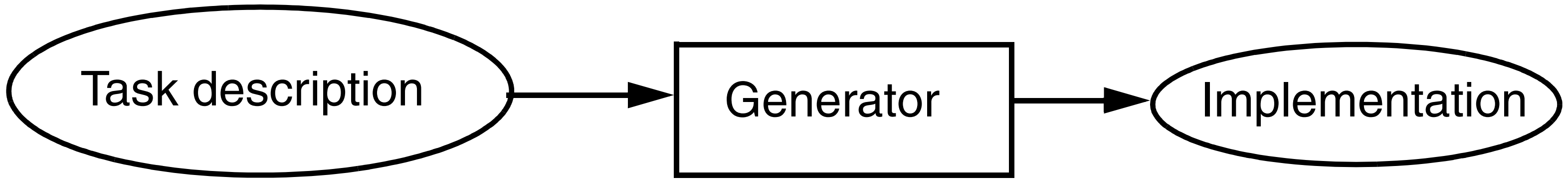
Categories of knowledge required to carry out a task:

General:	knowledge applicable to a wide variety of tasks e.g. English words; program in C
Domain-specific:	knowledge applicable to all tasks of this type e.g. group word in sets; implement arbitrary numbers of sets of strings in C
Task-specific:	knowledge about the particular task at hand e.g. sets of words to characterize animals

A domain-specific language is used to describe the particular task

A domain-specific generator creates a C program that stores the particular set of strings.

The Generator Principle



Application generator: the most effective reuse method

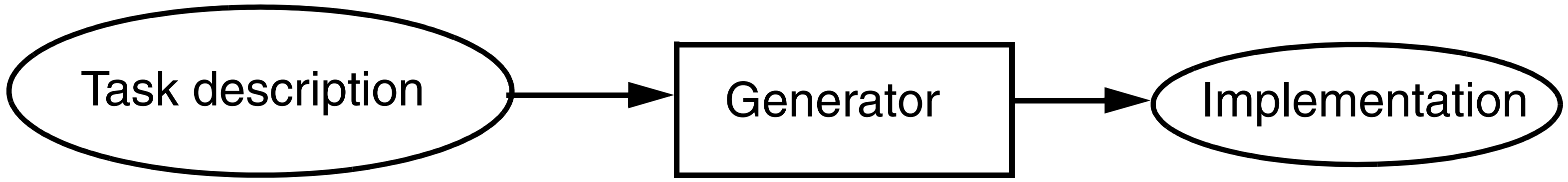
[Ch. W. Kruger: Software Reuse]

narrow, specific application domain	completely understood Implementation automatically generated
Abstractions on a high level (using domain knowledge)	transformed into executable software
User understands **abstractions** of the application domain	**Generator expert** understands **implementation methods**

wide cognitive distance
generator makes expert knowledge available

Examples: Data base report generator
GUI generator
Parser generator

Generators for DSLs



Domain-specific languages (DSL)

Domains outside of informatics
- Robot control
- Stock exchange
- Control of production lines
- Music scores

Software engineering domains
- Data base reports
- User interfaces
- Test descriptions
- Representation of data structures (XML)

Language implementation as domain
- Scanner specified by regular expressions
- Parser specified by a context-free grammar
- Language implementation specified for *Eli*

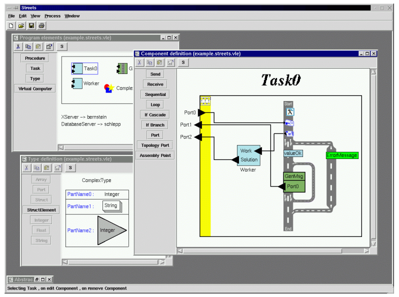
Some student projects:

Party organization
Soccer teams
Tutorial organization
Shopping lists
Train tracks layout

LED descriptions to VHDL
SimpleUML to XMI
Rule-based XML transformation

Generator:
transforms a DSL program into an
executable program or/and **into data,**
applies domain-specific methods and techniques

Outline



1. What are visual languages?
2. Domain-specific visual languages
3. **Ingredients for Language design**
4. A Development Environment for Visual Languages
5. Pattern-Based Specifications in DEViL

Textual Languages

Tokens

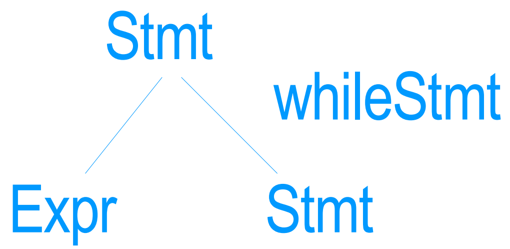
identifiers	names of entities
literals	values of operands
keywords, operators, delimiters	characterize language constructs

Concrete syntax representation of constructs

CFG Stmt ::= while (Expr) Stmt

Abstract syntax information structure

CFG



Contextual constraints

e.g. scope rules

relate names to entities

```
float x;
      x=x*2;
```

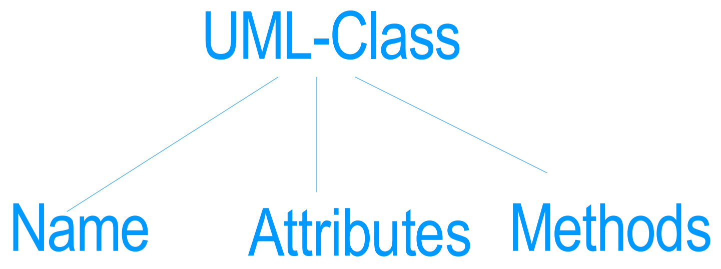
Semantics of constructs and entities

determined by language domain

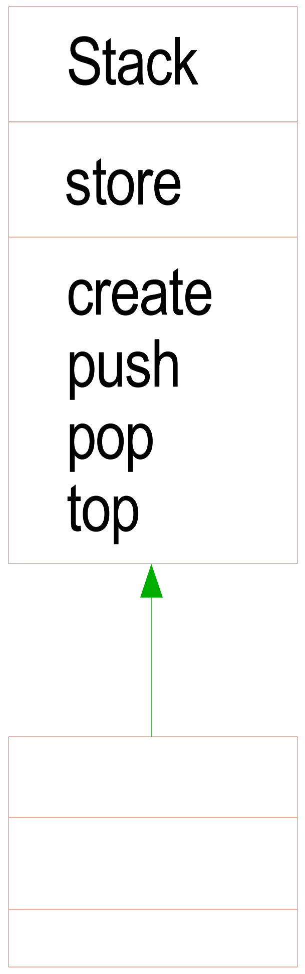
related to abstract syntax, supported by representation

Visual Languages

Abstract syntax



Representation



Graphics, icons, visual annotations

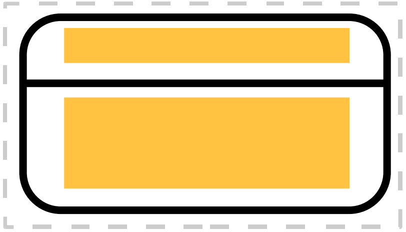
Spacial relations, nesting

connection

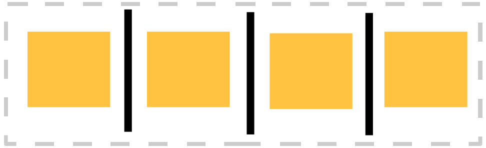
Contextual constraints
Semantics of constructs and entities
... based on abstract syntax

Compositional Visual Patterns

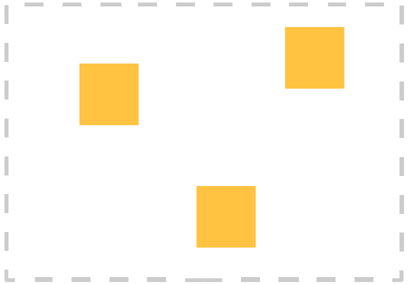
Form



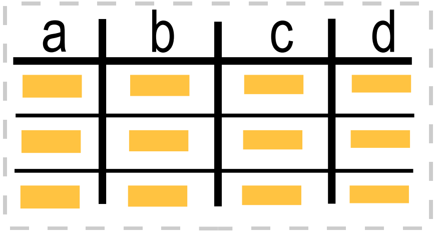
List



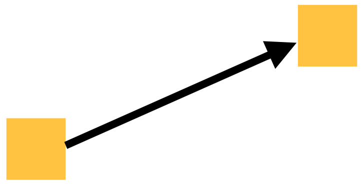
Set



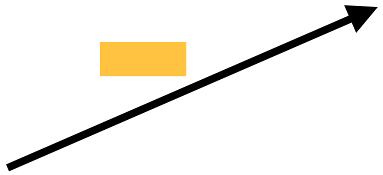
Table



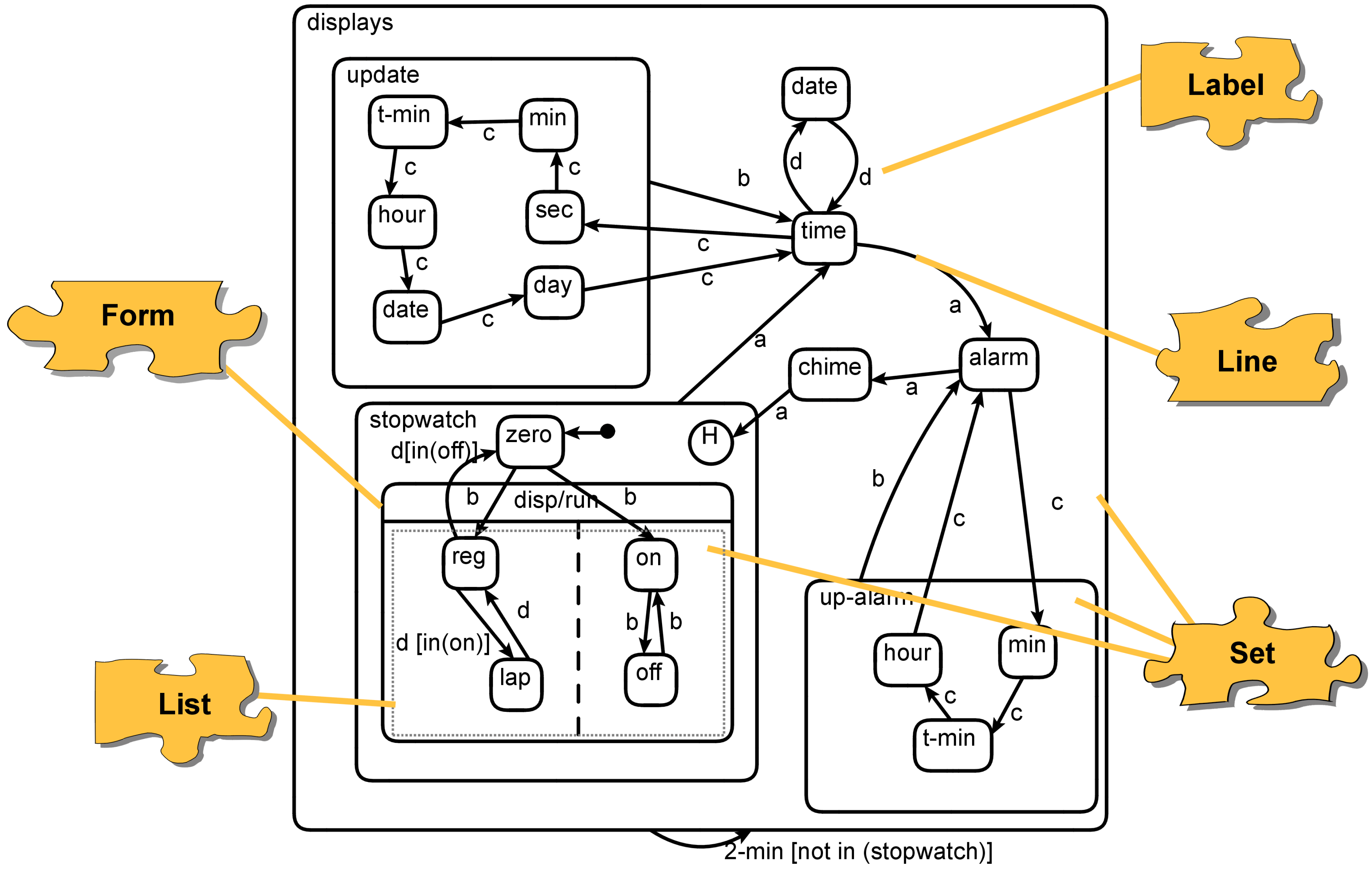
Line



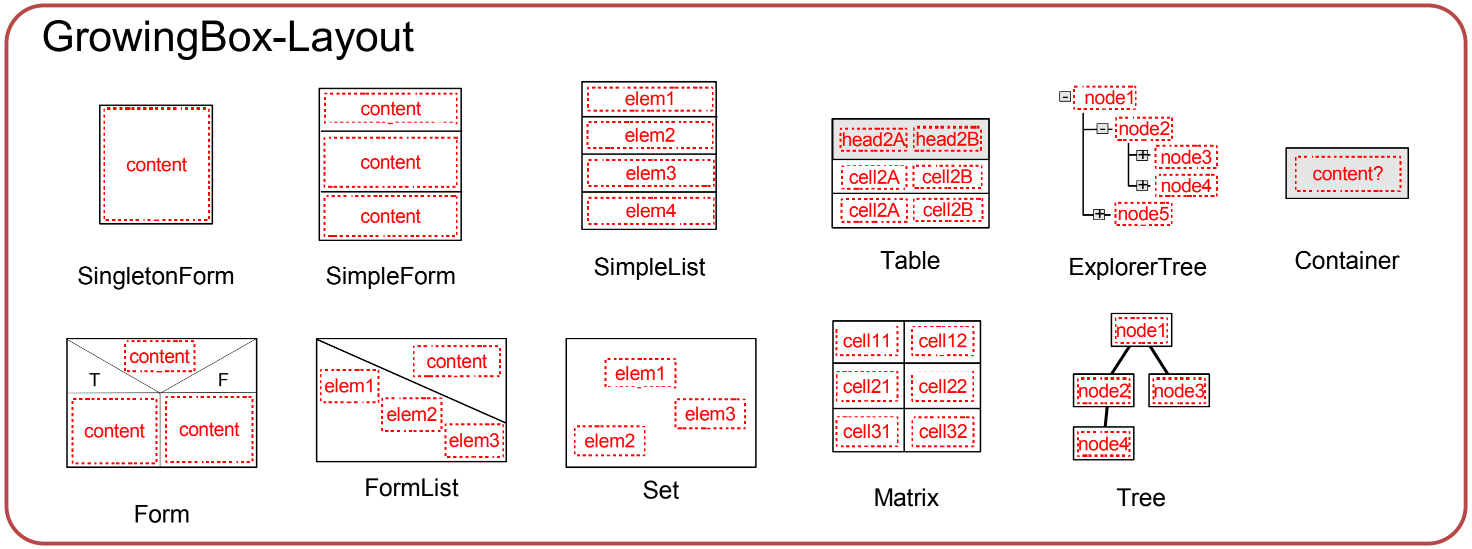
Label

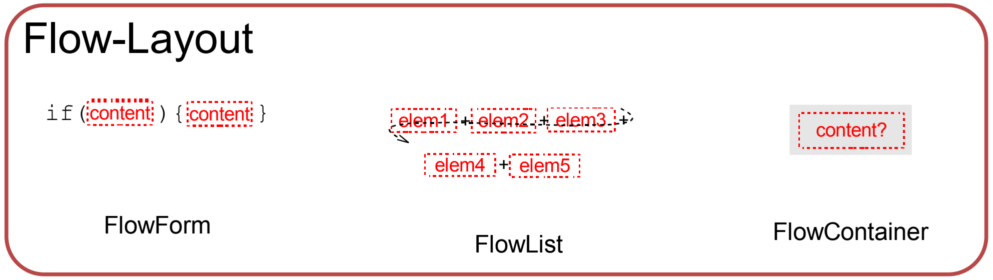


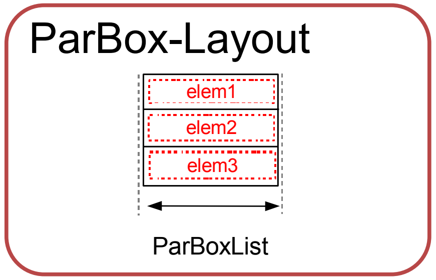
Occurrences of Compositional Concepts

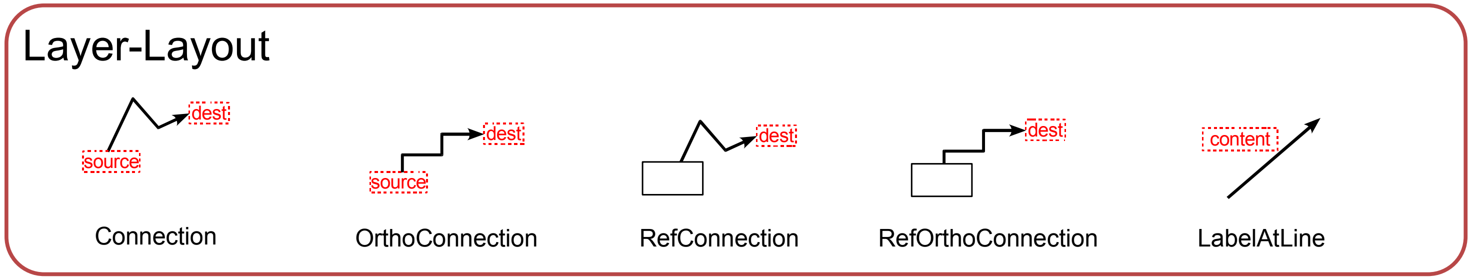


Pattern Variants and Layout Strategies

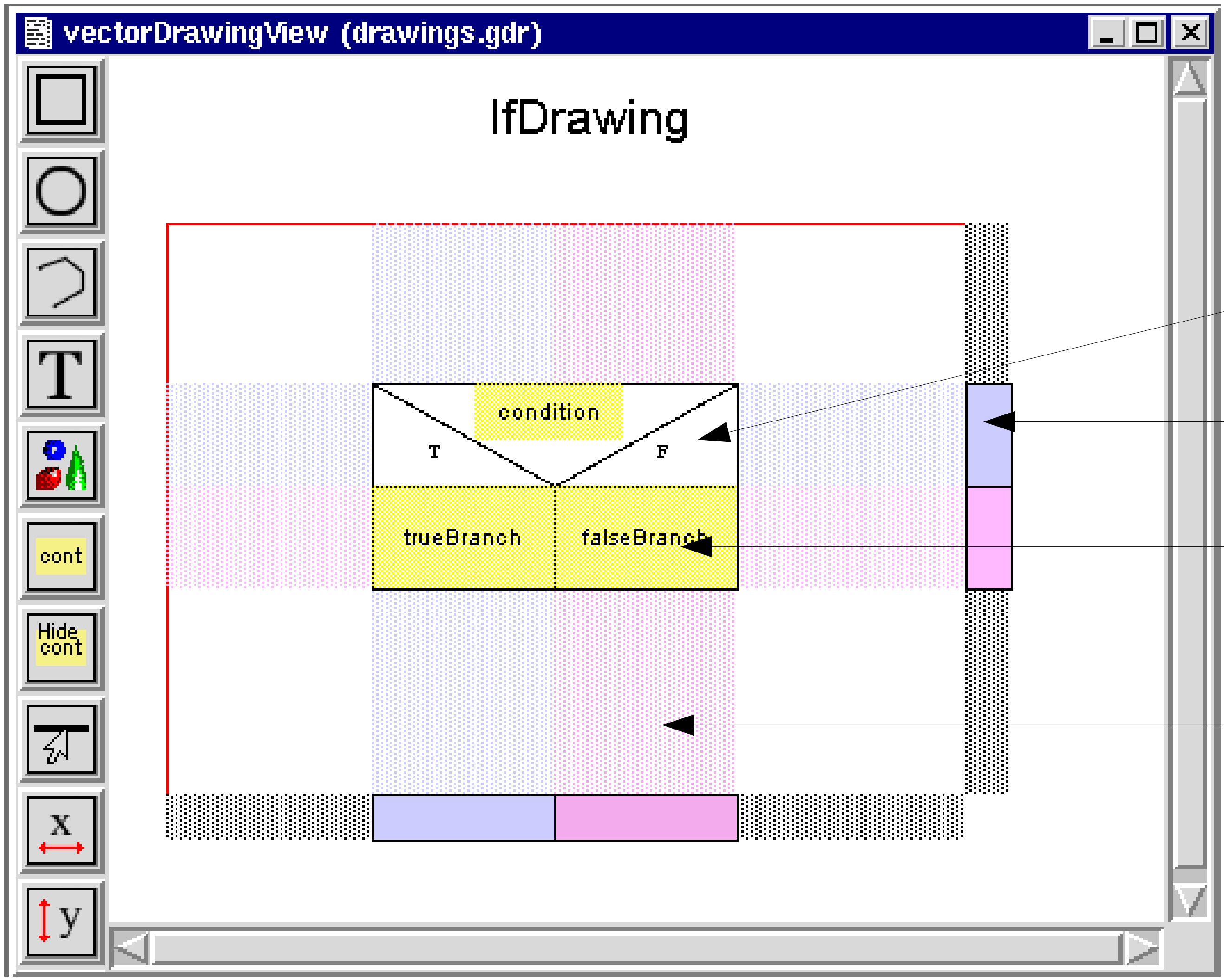








Generic Drawings: Design Graphics with Flexible Containers



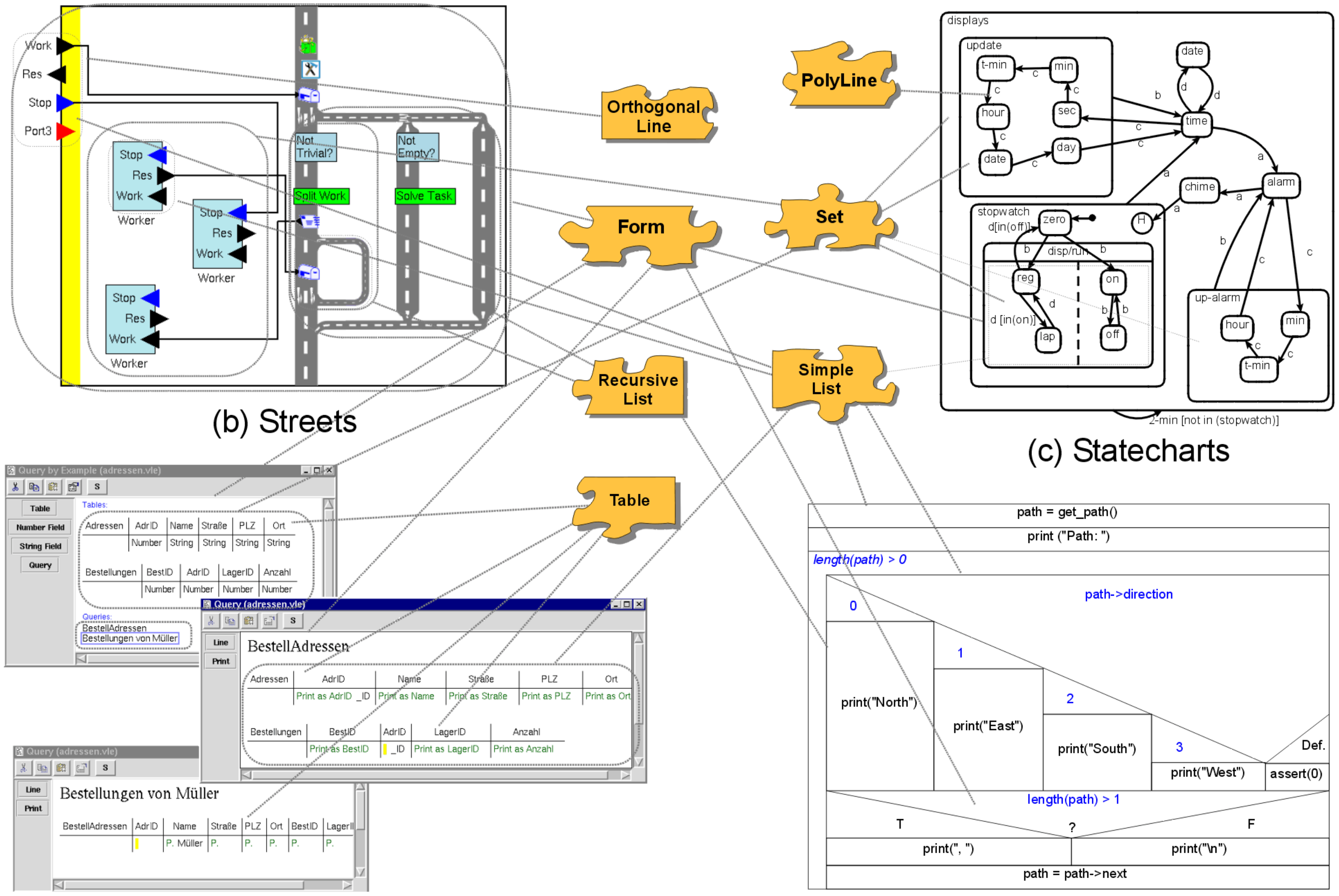
Gaphic primitive

Extension interval

Container

Indication of extension interval

Instances of Visual Patterns



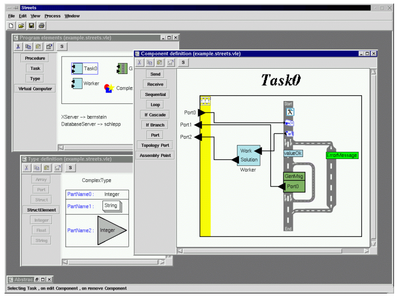
(b) Streets

(c) Statecharts

(d) Query by Example

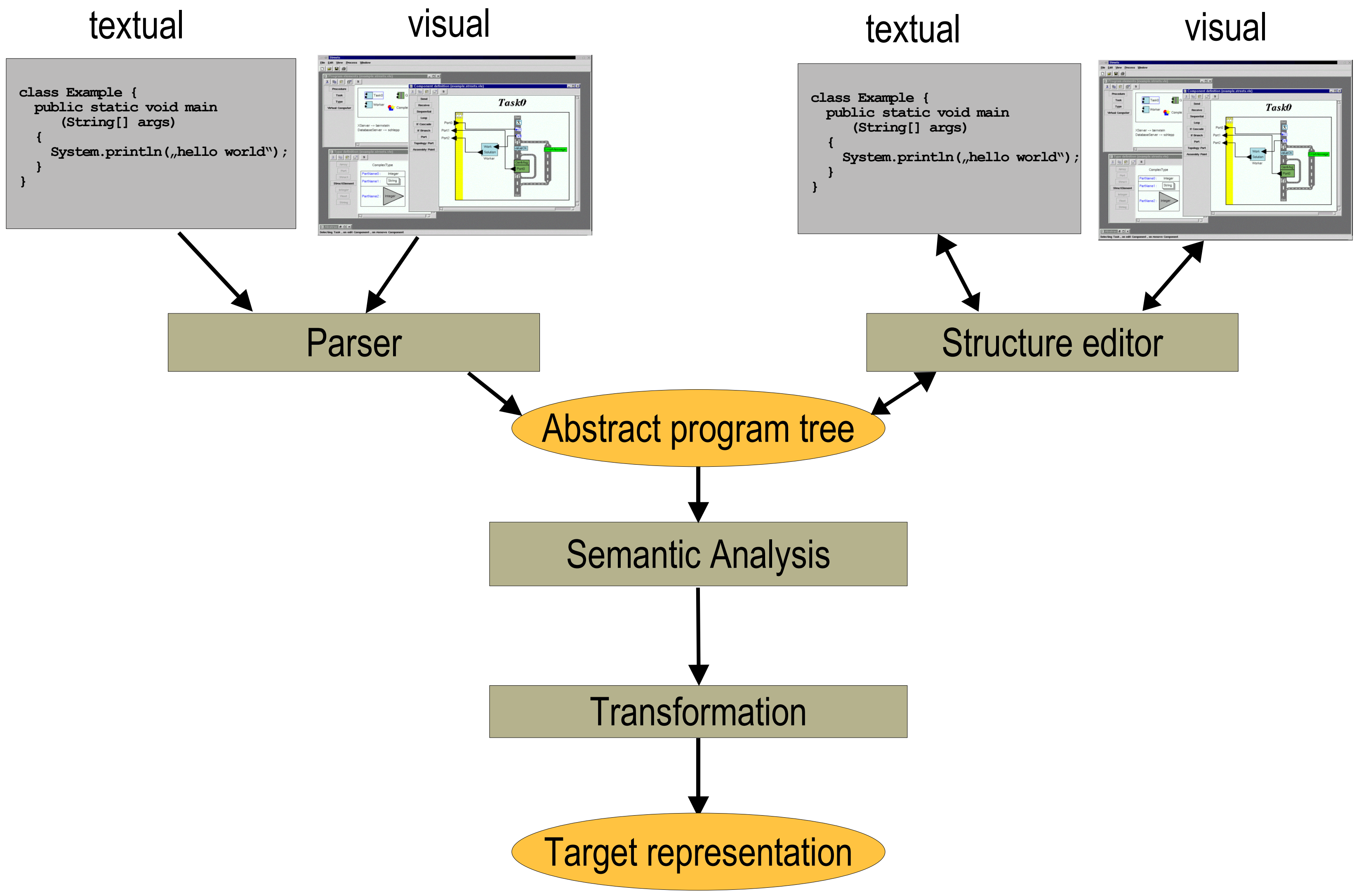
(e) Nassi-Shneiderman-Diagrams

Outline



1. What are visual languages?
2. Domain-specific visual languages
3. Ingredients for Language design
4. **A Development Environment for Visual Languages**
5. Pattern-Based Specifications in DEViL

Language Implementation Variants



```
class Example {
  public static void main
      (String[] args)
  {
    System.println(„hello world“);
  }
}
```

```
class Example {
  public static void main
      (String[] args)
  {
    System.println(„hello world“);
  }
}
```

Visual Language Implementation Tools

DiaGen

Based on hypergraph grammars. It generates editors, which allow free and structured editing.

GenGEd

Is based on graph grammars to describe syntax. Algebraic specifications are used to describe graphical symbols, relations and layout constraints. Structure editors are generated.

MetaEdit+

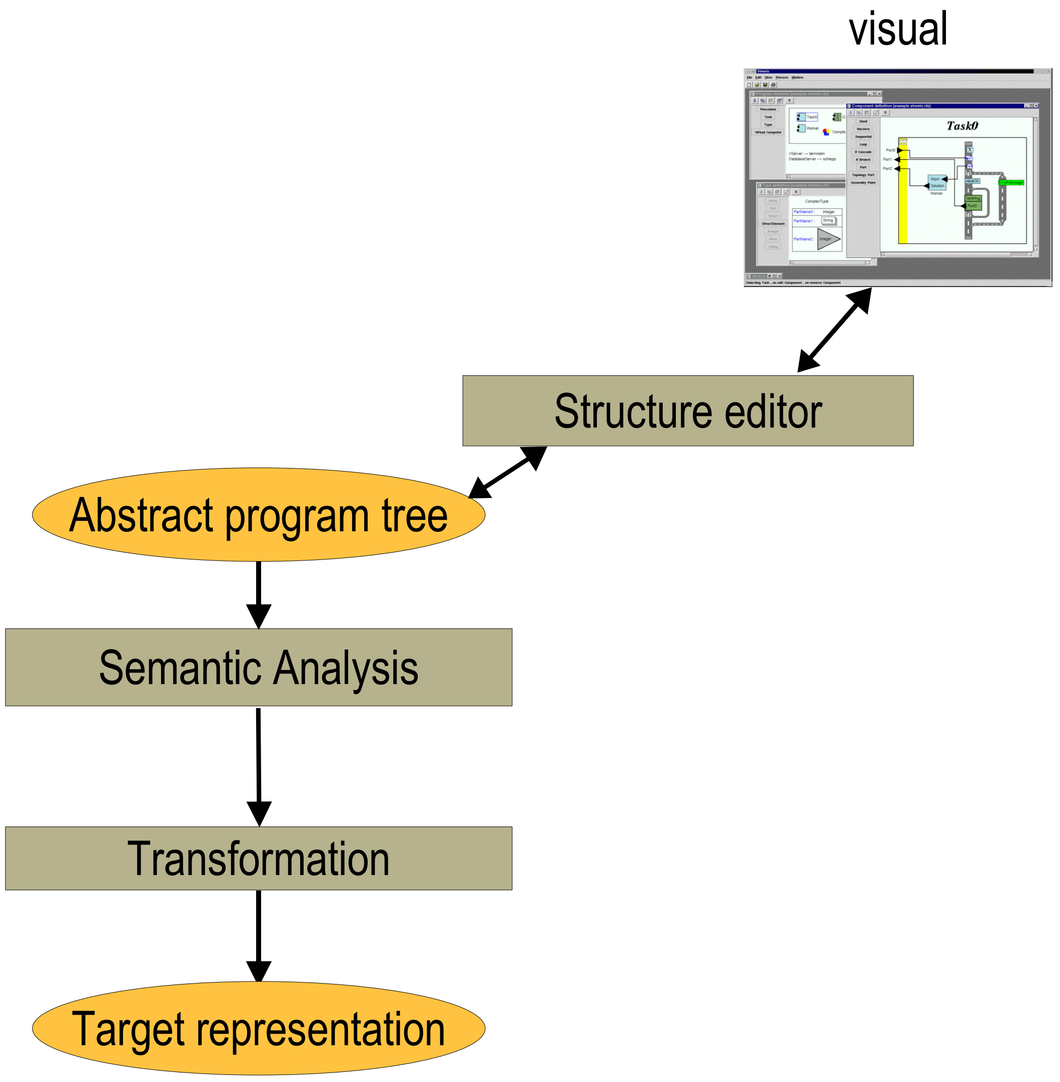
Used to implement domain-specific modeling languages. A model of the language structure is specified. The graphical representation of instances are derived using graphical tools.

DEViL

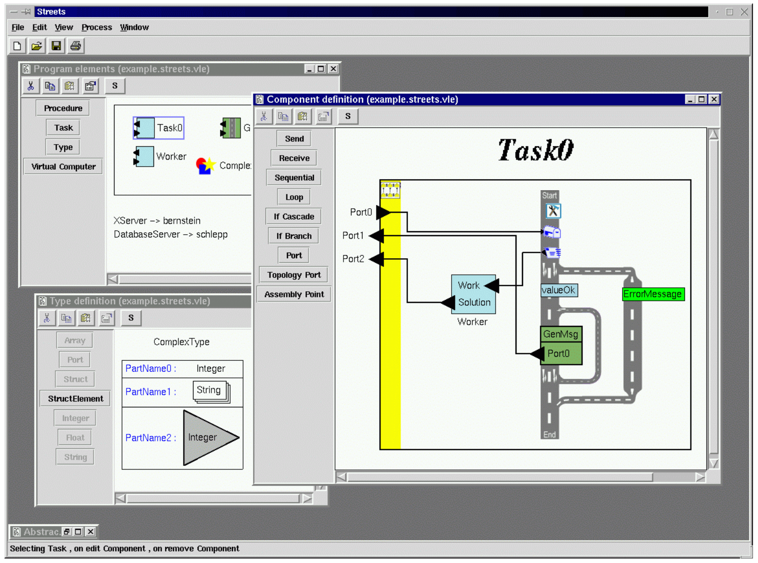
DEViL generates implemenations of visual languages from specifications of the abstract structure, visual representation, and of analysis and transformation. DEViL generates a complete language processor including a visual structure editor.

... For more and URLs see DEViL's HomePage

Frontend: Visual Structure Editor



Tasks of Visual Structure Editors



- Draw the visual representation
- Determine the layout of constructs
- Show several views

- Create and insert constructs
- Delete, move, copy, cut, paste constructs
- Update the abstract representation

- Load, store the visual representation

Structure of the DEViL System

DEViL: *Development Environment for Visual Languages*

DEViL Designer *visual Specifications*

Library of Visual Patterns
Reusable specifications of visual representations

VL-Generator
Generators and libraries for implementation of visual languages

BasicTypes
Data types for abstract Structur and attribute computations

ActiveCanvas
Interface for drawing graphic representations

MDIFrame
Implementation of the Multiple-Document-Interface (MDI)

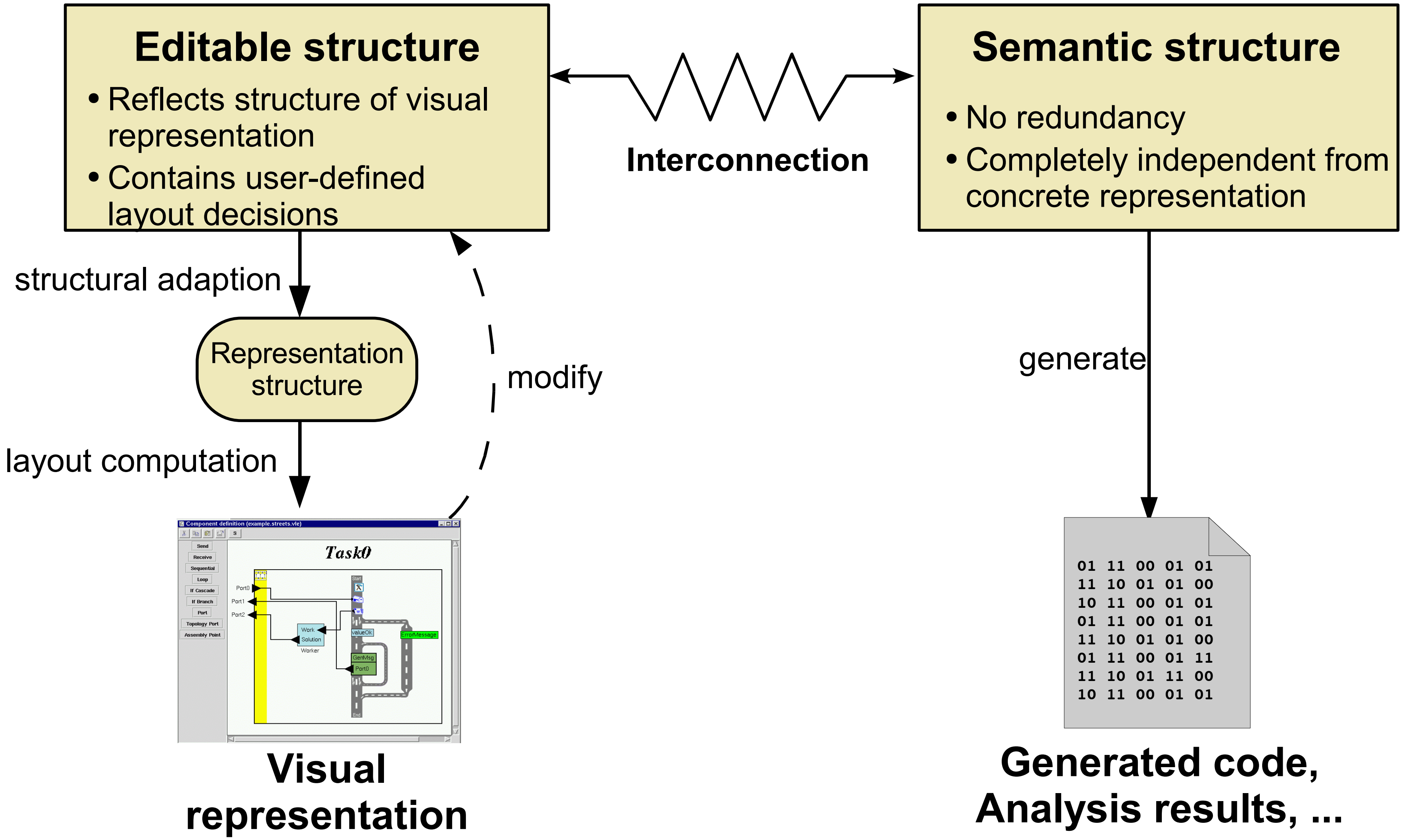
Wodan
Implementation of build processes

Eli
Analysis and transformation

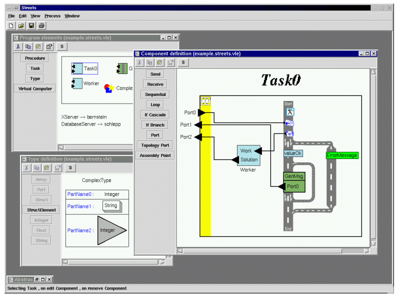
Tcl/Tk + Tkzinc
Graphic representation, visualenvironment

Dot
Graph layout

Model of Visual Structure Editors

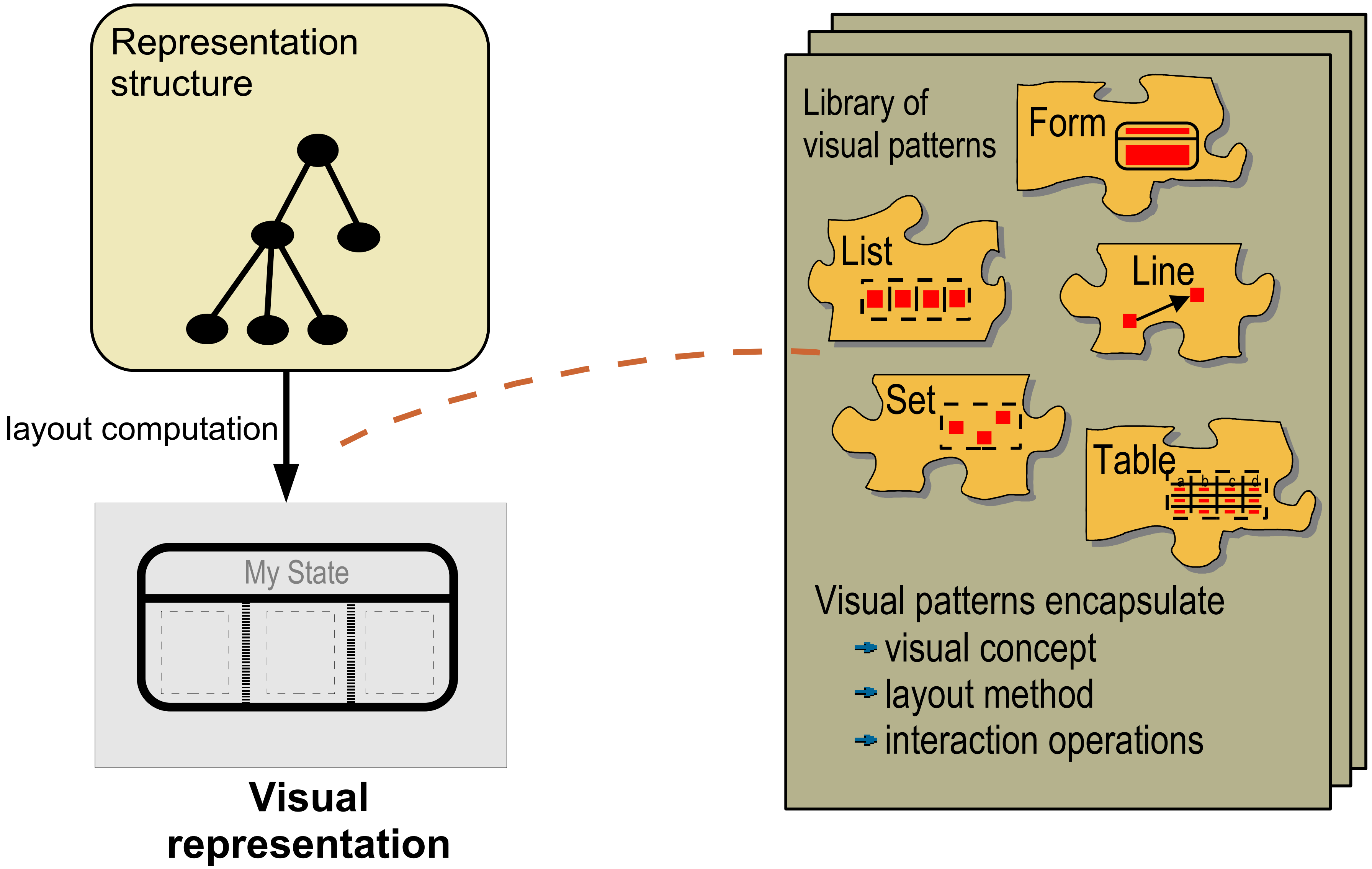


Outline



1. What are visual languages?
2. Domain-specific visual languages
3. Ingredients for Language design
4. A Development Environment for Visual Languages
5. **Pattern-Based Specifications in DEViL**

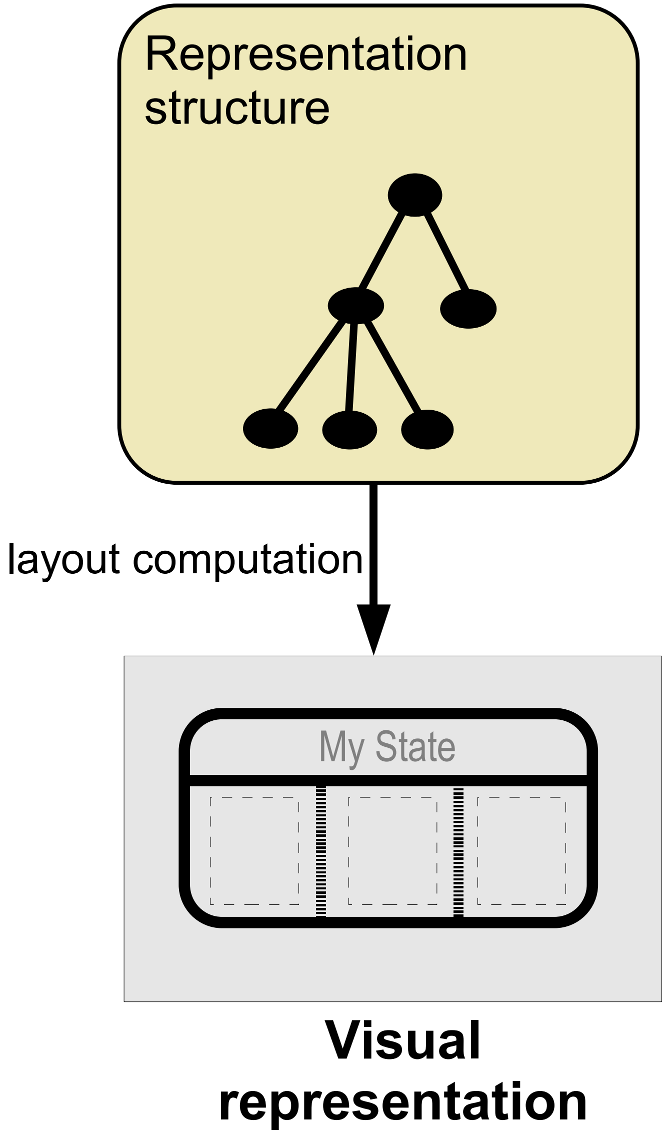
Pattern-based Specification



Visual patterns encapsulate

- visual concept
- layout method
- interaction operations

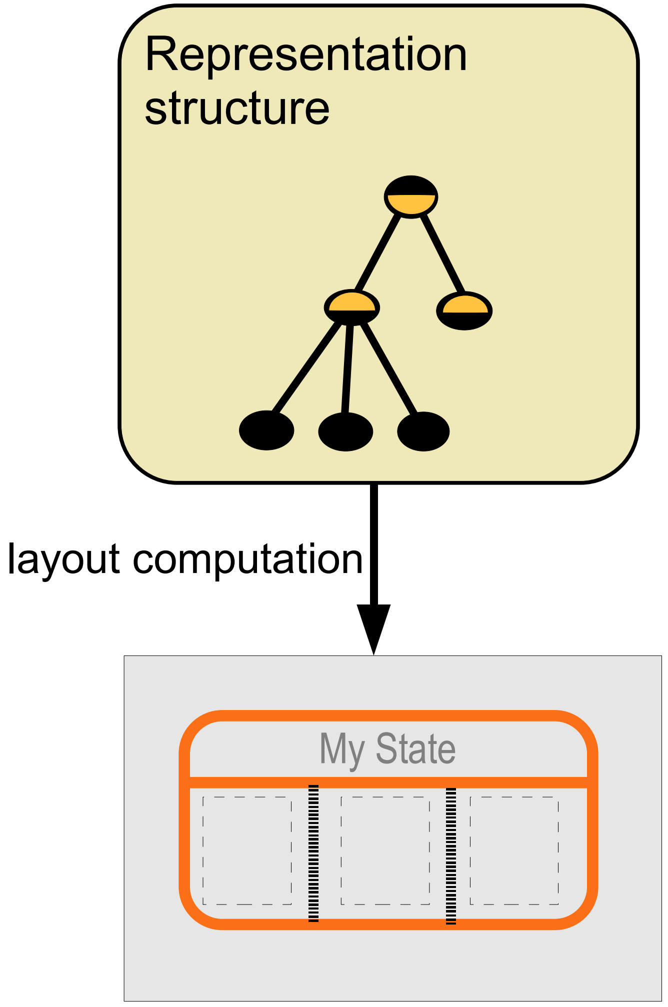
Pattern-based Specification



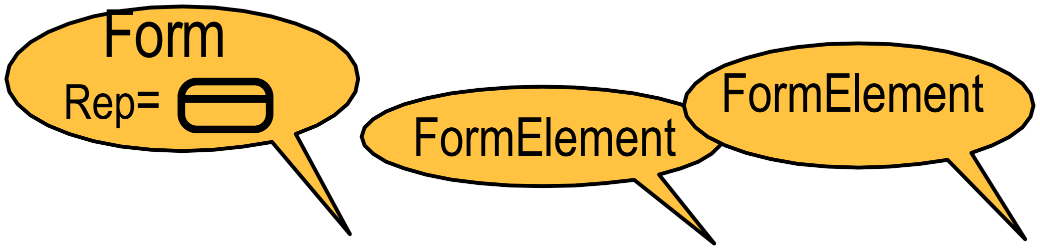
ANDSuperstate ::= ASName ASRegionList

ASRegionList ::= ASRegion*

Pattern-based Specification



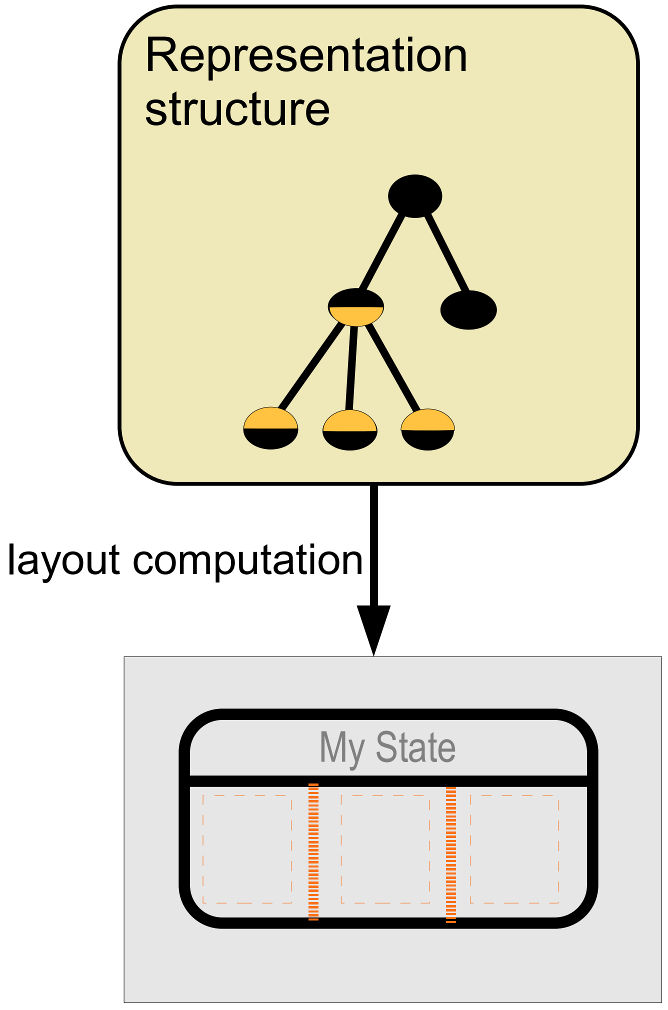
Visual representation



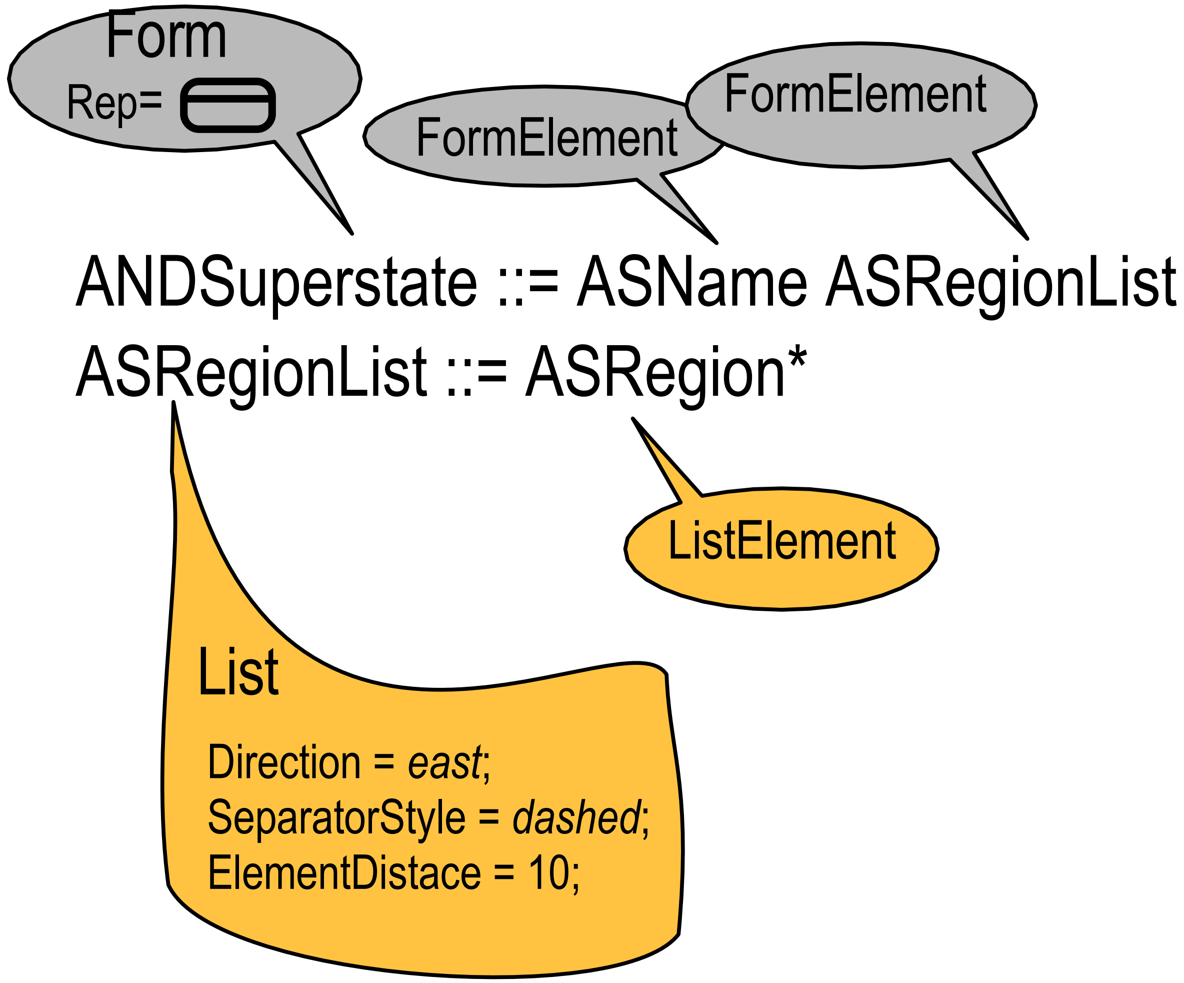
ANDSuperstate ::= ASName ASRegionList

ASRegionList ::= ASRegion*

Pattern-based View Specification



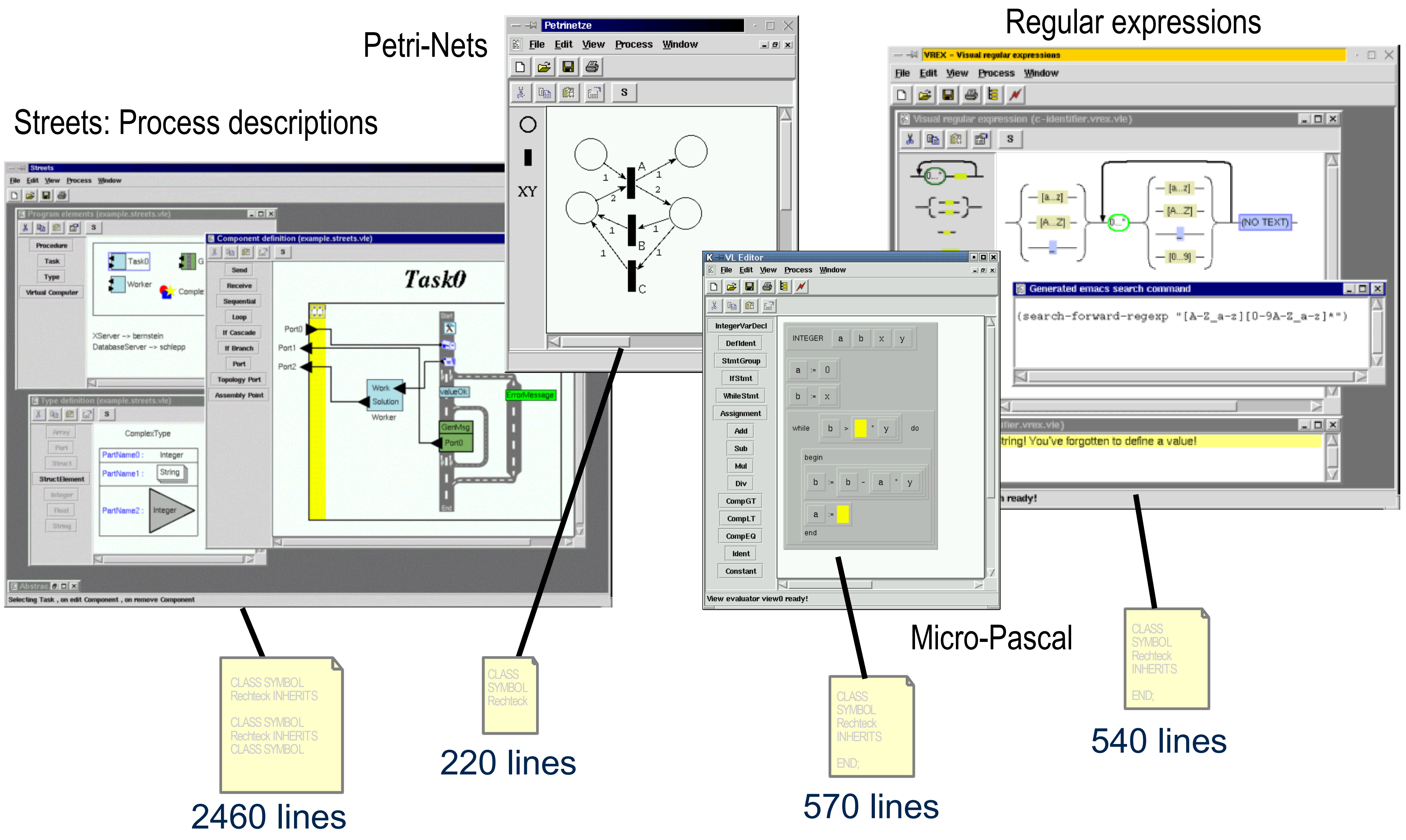
Visual representation



ANDSuperstate ::= ASName ASRegionList
ASRegionList ::= ASRegion*

Direction = *east*;
SeparatorStyle = *dashed*;
ElementDistace = 10;

Sizes of Language Specifications


Petri-Nets
Regular expressions
Streets: Process descriptions
Micro-Pascal
2460 lines
220 lines
570 lines
540 lines
CLASS SYMBOL Rechteck INHERITS
CLASS SYMBOL Rechteck
END;
Petrinetze
VREX - Visual regular expressions
Visual regular expression (c-identifier.vrex.vle)
Generated emacs search command
VL Editor
Streets
Program elements (example.streets.vle)
Component definition (example.streets.vle)
Type definition (example.streets.vle)
Task0

```
(search-forward-regexp "[A-Z_a-z][0-9A-Z_a-z]*")
```

tring! You've forgotten to define a value!

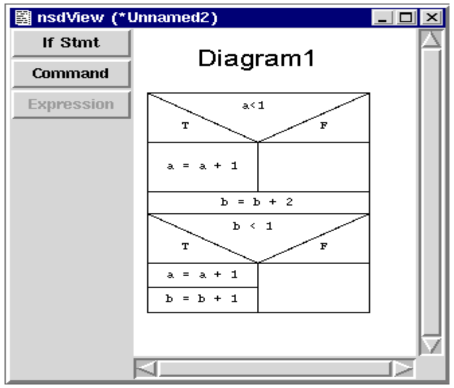
Selecting Task , on edit Component , on remove Component

View evaluator view0 ready!

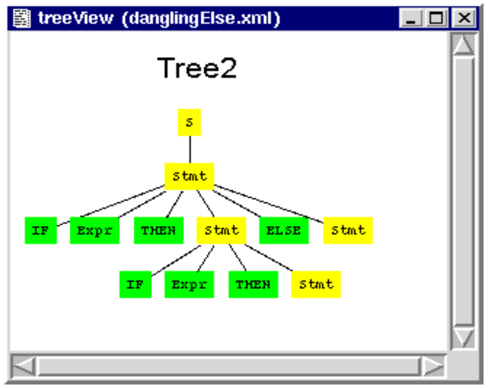
XServer --> bernstein
DatabaseServer --> schlepp

Analysis of Example Specifications

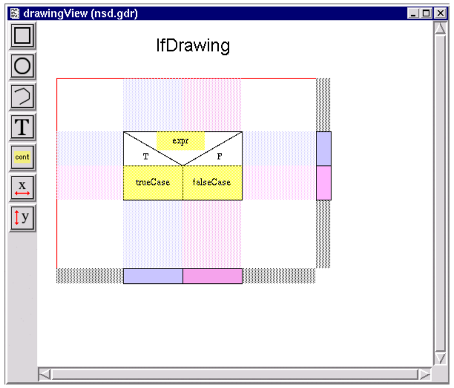
Sprache	**Number of struct.classes**	LOC structure	LOC attr. comp.	LOC Gen. drawings	LOC sync.	**LOC total**	≈ Spec. effort
Simplified NSD diagrams	**5**	19	34	22	0	**75**	1,6 h
Derivation Tree Assistant	**12**	73	134	24	44	**275**	5,9 h
Generic Drawings	**27**	184	518	157	231	**1090**	23 h
PaderWAVE	**93**	414	1322	527	450	**2713**	58 h



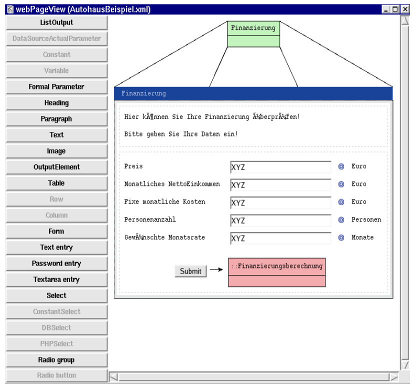
Simplified NSD diagrams



Derivation Tree Assistant

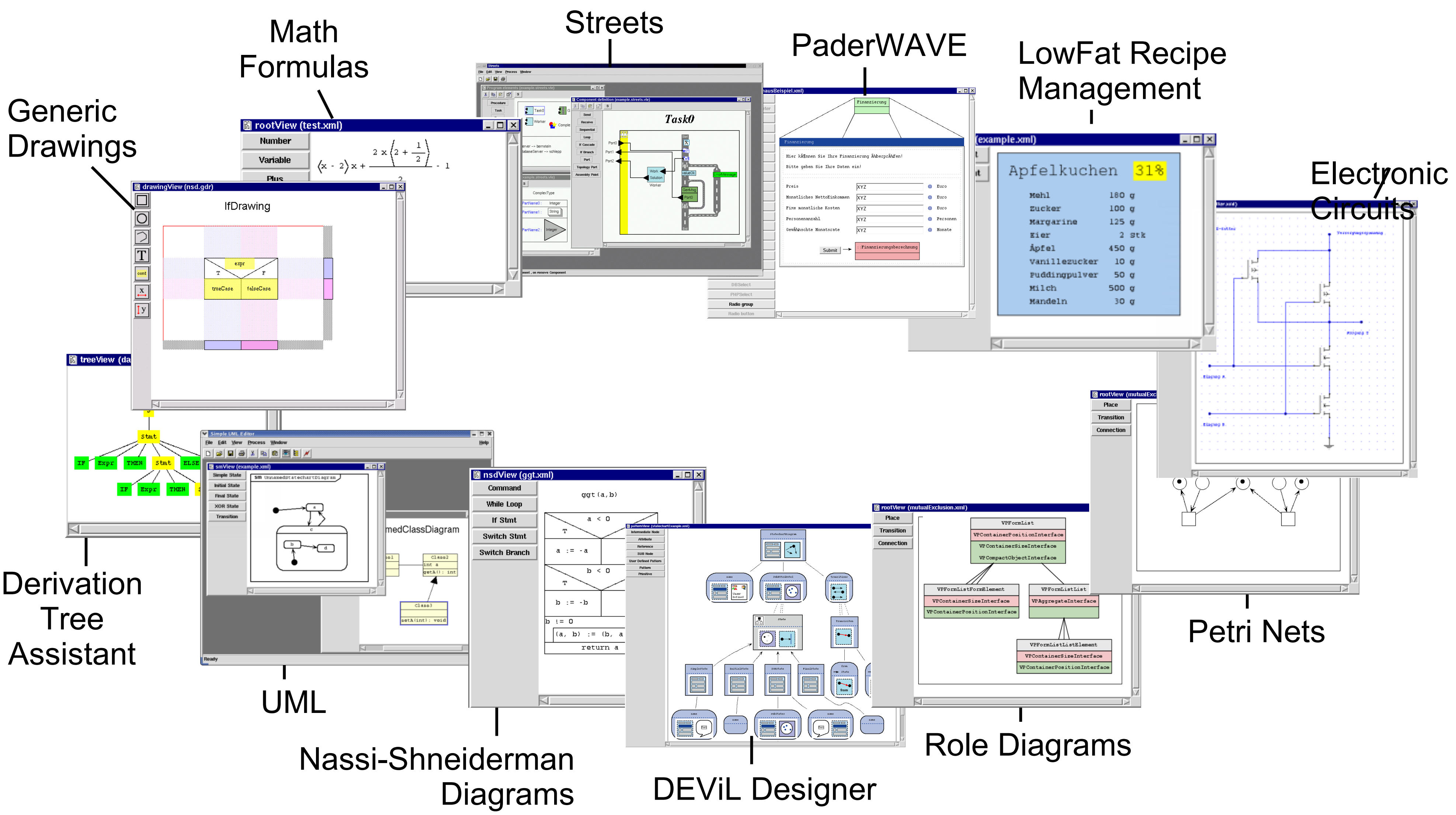


Generic drawings

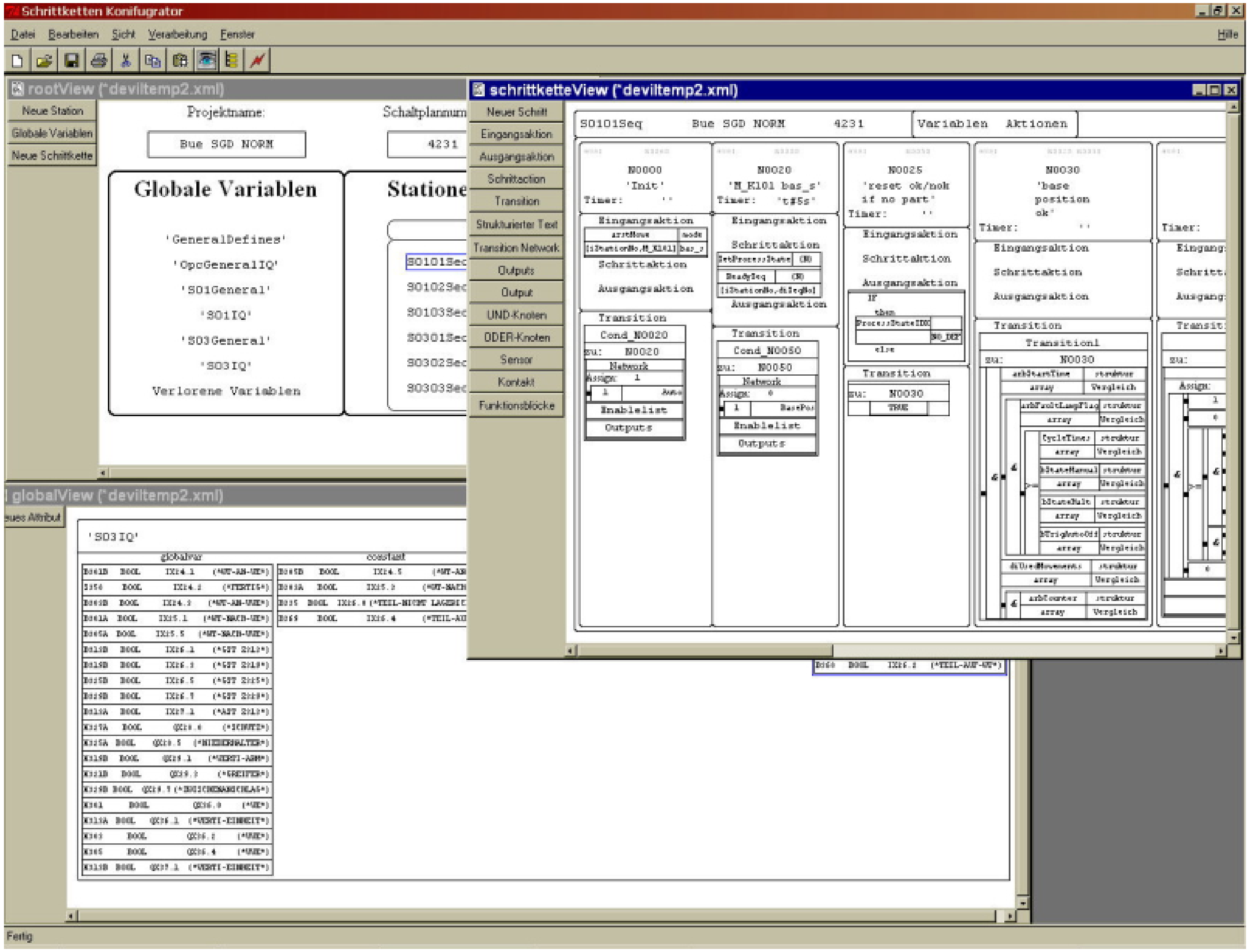


PaderWAVE

DEViL Generated Structure Editors



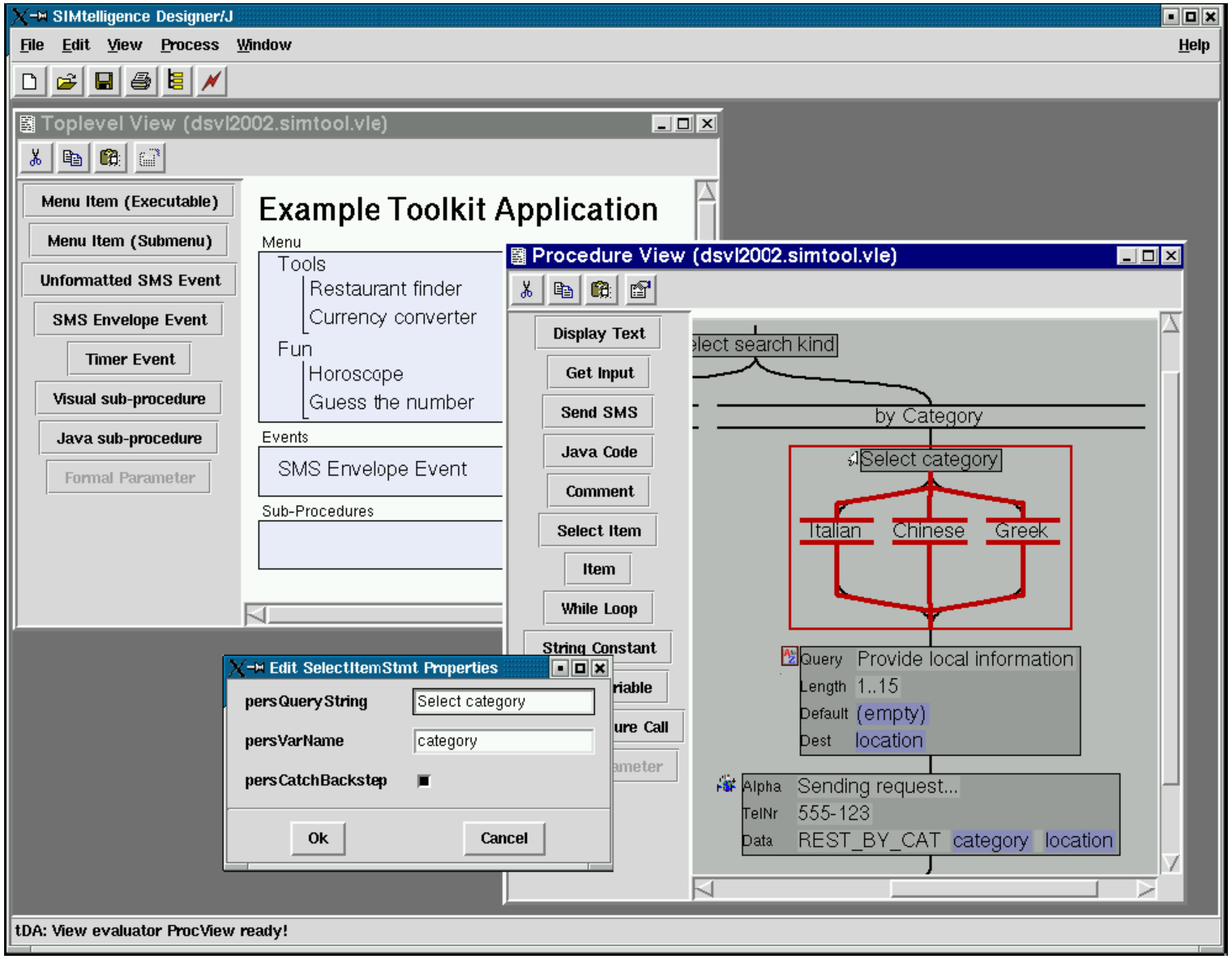
Industrial Project with Bosch: Robot Control for Motor Production



globalvar			
B061B	BOOL	IX14.1	(*WT-AN-VE*)
B354	BOOL	IX14.2	(*FERTIG*)
B060B	BOOL	IX14.3	(*WT-AN-VRE*)
B061A	BOOL	IX15.1	(*WT-NACH-VE*)
B060A	BOOL	IX15.5	(*WT-NACH-VRE*)
B013B	BOOL	IX16.1	(*GST ZYL1*)
B015B	BOOL	IX16.3	(*GST ZYL3*)
B015B	BOOL	IX16.5	(*GST ZYL5*)
B016B	BOOL	IX16.7	(*GST ZYL6*)
B013A	BOOL	IX17.1	(*AST ZYL1*)
K017A	BOOL	QX18.6	(*SCHUTZ*)
K015A	BOOL	QX18.5	(*NIEDERHALTER*)
K018B	BOOL	QX19.1	(*VERTI-ABM*)
K011B	BOOL	QX19.3	(*GREIFER*)
K019B	BOOL	QX19.7	(*EINSCHRAUBSCHLAG*)
K061	BOOL	QX16.0	(*VE*)
K013A	BOOL	QX16.1	(*VERTI-EINHEIT*)
K063	BOOL	QX16.2	(*VRE*)
K065	BOOL	QX16.4	(*VRE*)
K013B	BOOL	QX17.1	(*VERTI-EINHEIT*)

constant			
B065B	BOOL	IX14.5	(*WT-AN
B063A	BOOL	IX15.3	(*WT-NACH
B035	BOOL	IX16.0	(*TEIL-NICHT LAGERIC
B069	BOOL	IX16.4	(*TEIL-AU
B060	BOOL	IX16.2	(*TEIL-AUF-WT*)

Industrial Project with Sagem Orga: SIMtelligence Designer/J



persQueryString: Select category
persVarName: category
persCatchBackstep: ■

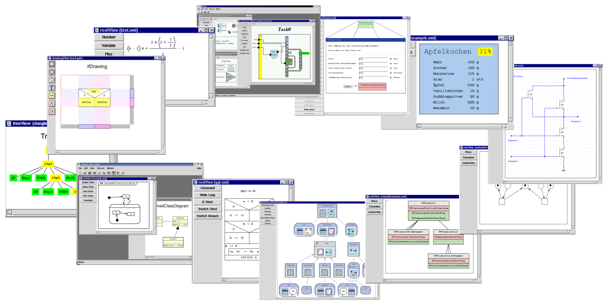
Query: Provide local information
Length: 1..15
Default: (empty)
Dest: location

Alpha: Sending request...
TelNr: 555-123
Data: REST_BY_CAT category location

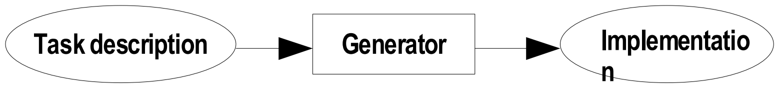
tDA: View evaluator ProcView ready!

Conclusion

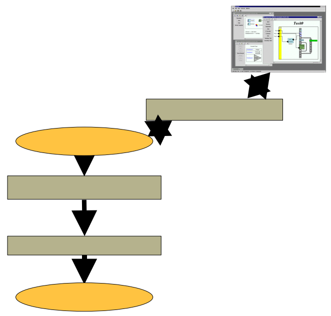
- Wide spectrum of visual language constructs



- Well suited for DSL generators



- Tool support



- Visual structure editor & analysis & translation

- Visual Patterns attached to syntax