Domain-Specific Languages (DSL)

Software Systems (Computer & Embedded Systems Engineering)

Arjan Mooij

January 2023 (week 8)

An initiative of industry, academia and TNO

TU Delft Delft University of Technology

ESI

Objectives

At the end of the course, you should be able to:

- Explain the purpose of Domain-Specific Languages, including several application areas
- Explain the basics of grammars and parsing
- Create basic textual Domain-Specific Languages, including editor support, validation and generators

Assessment:

- Modeling assignment using Domain-Specific Languages (in groups of 2 students)
- Reflection document on Model-Based Development (individual)

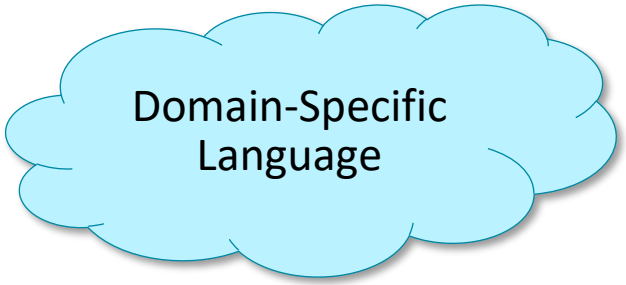
Motivation

Domain-Specific Languages (DSL)

What is Jargon?

Oxford dictionaries:

- **Special words or expressions**
  - used by a profession or group
  - that are difficult for others to understand

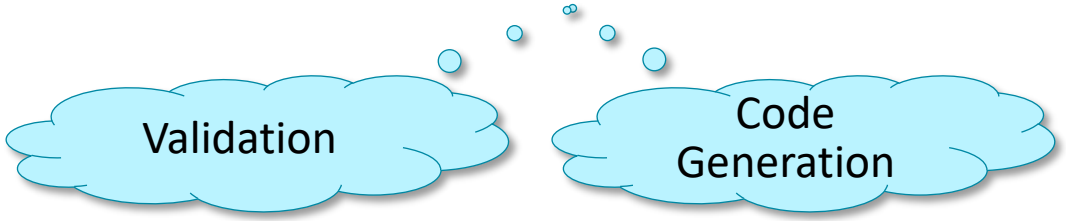


Wikipedia:

- **Terminology defined in relationship to**

  a specific activity, profession, group, or event
  - … a barrier to communication with those not familiar with the language

➢**A standard term may be given a more precise or unique usage**



What is a Domain-Specific Language?

- **What are your associations with the term Domain-Specific Language?**
- **Do you know any Domain-Specific Languages?**

Think → Pair → Share

What is a Domain-Specific Language?

General-purpose programming languages:

- C, C++, Java, Python, etc.

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

Horizontal Domain-specific languages:

- HTML for web pages
- SQL for relational database queries

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

```sql
SELECT *
 FROM Book
 WHERE price > 100
 ORDER BY title;
```

Vertical Domain-specific languages:

- Specifically designed for a specific application by a single company

Canon ASML VANDERLANDE PHILIPS THALES

What about the other model-based techniques from this course?

- **PlantUML** **for** **Unified Modeling Language (UML)**

- **YAKINDU Statechart Tools** **for** **Finite-State Machines (FSM)**

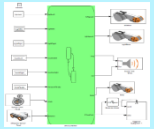
➔ **Horizontal DSLs**

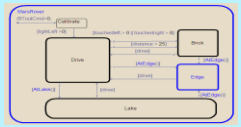
Generations of programming languages

4th generation

Higher abstraction &
Expressive power

Domain-Specific Language (DSL)





```
SELECT *
 FROM  Book
 WHERE price > 100.00
 ORDER BY title;
```

code generator

3rd generation

High-level programming languages
(Fortran, Cobol, C, Java, ..)

```
if (frameSize.height > screenSize.height) {
    frameSize.height = screenSize.height;
}
if (frameSize.width > screenSize.width) {
    frameSize.width = screenSize.width;
}
```

compiler

2nd generation

Assembly code

```
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
```

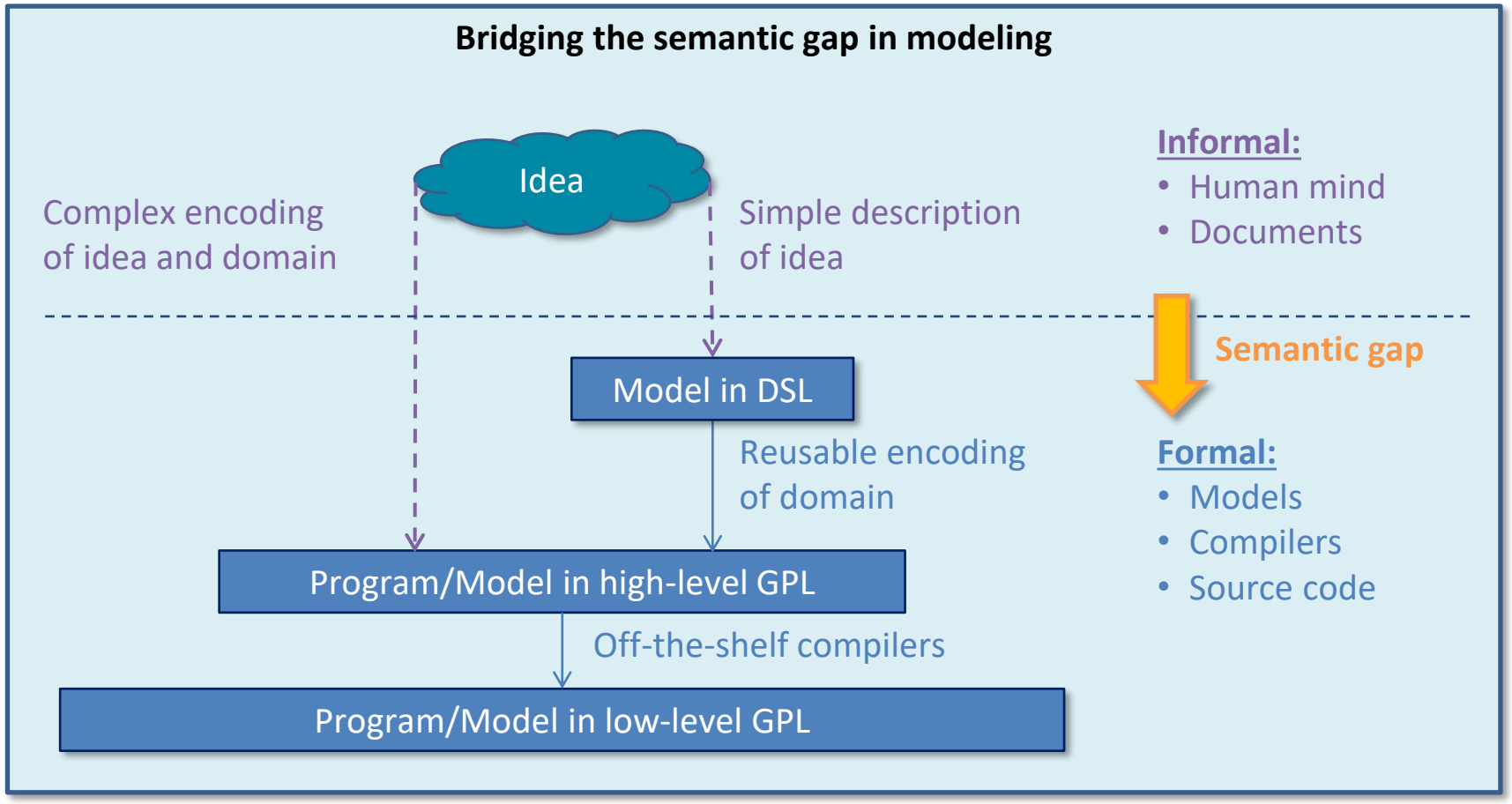
assembler

1st generation

Machine instructions

op	rs	rt	address
100011	00011	01000	00000 00001 000100

Modeling Perspective on DSLs



DSL as Central Artifact

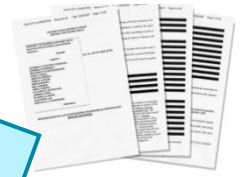
```
restriction VeryCloseTableTopAndBeam
    activation
          Distance(TableTop, Beam) < 20 mm
    effects
        userGuidance "TableTop and Beam very close"
        relative limit TableTop*[Rotation, Translation],
                       Beam*[Rotation, Translation]
            at 0


restriction ApproachingTableTopAndBeam
    activation
          Distance(TableTop, Beam) < 35 mm + 15 cm
    effects
        userGuidance "TableTop and Beam approaching"
        relative limit TableTop*[Rotation, Translation],
                       Beam*[Rotation, Translation]
            at (Distance(TableTop, Beam) - 35 mm) / 15 cm
```

Xtext

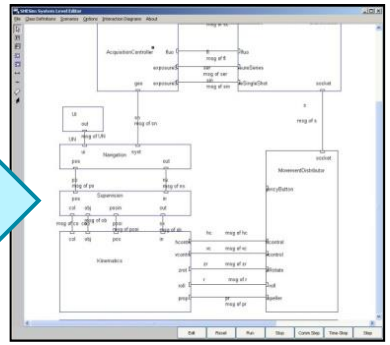
Xtend

Validation

Generate documentation

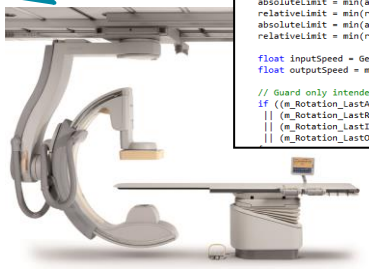


Generate analysis models



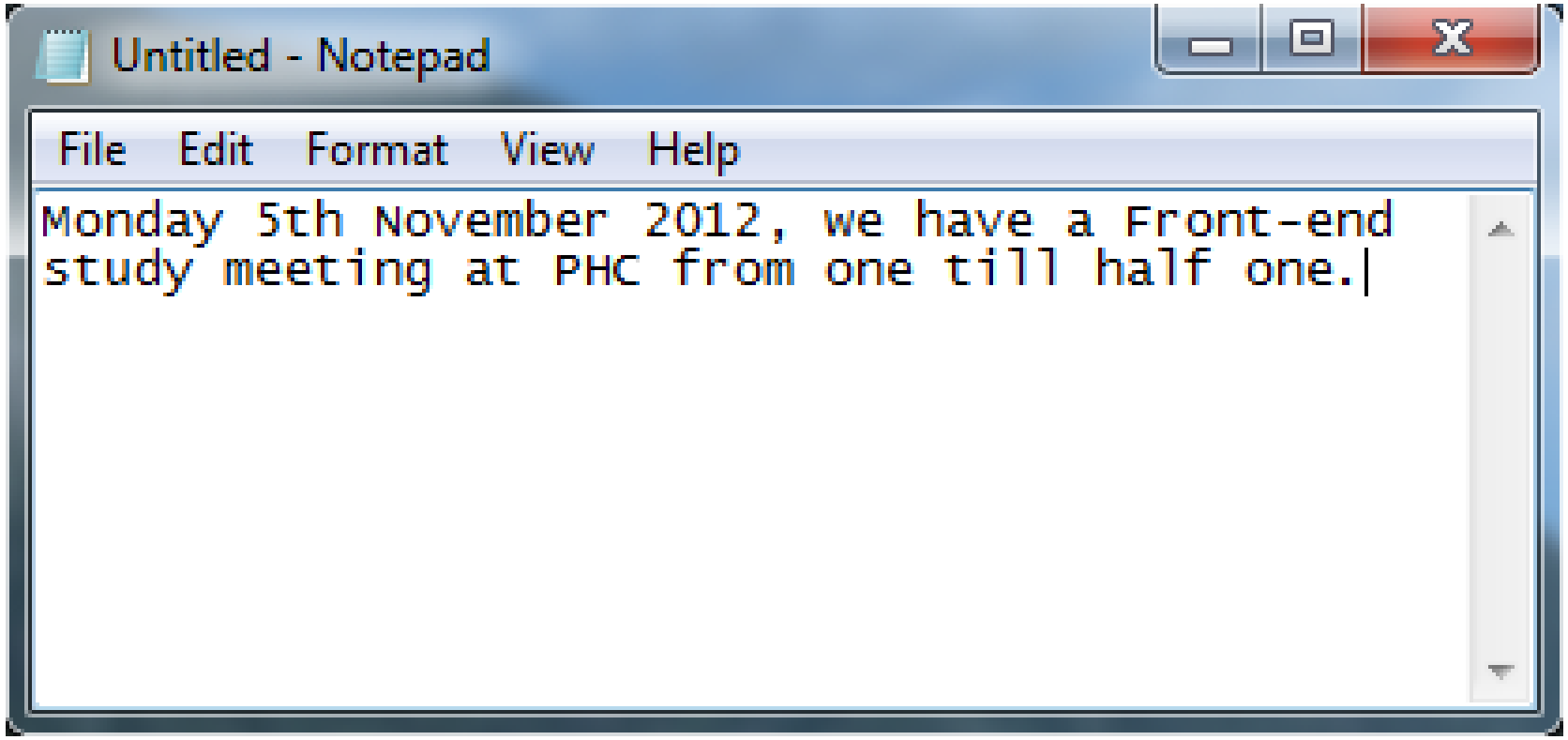
Generate source code

factor >30

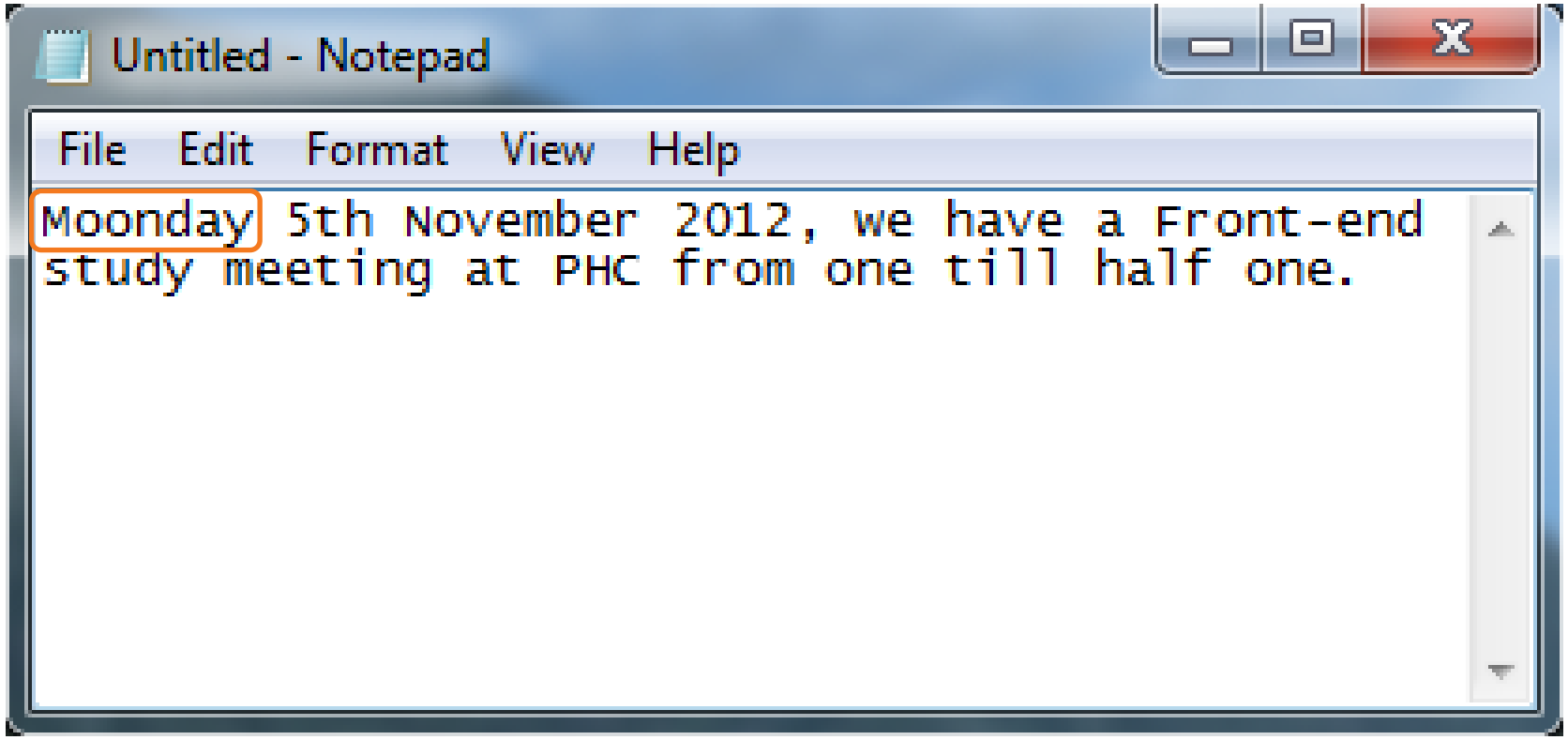


```
float CObject_Beam::GetRotationScaling()
{
  float absoluteLimit = GEN_t_float_max;
  float relativeLimit = 1.;

  absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_DetectorRotationInBetween);
  absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableTopAndBeam);
  relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableTopAndBeam);
  absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableTopAndDetector);
  relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableTopAndDetector);
  absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableBaseAndBeam);
  relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableBaseAndBeam);
  absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableBaseAndDetector);
  relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableBaseAndDetector);

  float inputSpeed = GetRequestedRotSpeed();
  float outputSpeed = min(inputSpeed * relativeLimit, absoluteLimit);

  // Guard only intended to reduce calls to logging
  if ((m_Rotation_LastAbsoluteLimit != absoluteLimit)
   || (m_Rotation_LastRelativeLimit != relativeLimit)
   || (m_Rotation_LastInputSpeed != inputSpeed)
   || (m_Rotation_LastOutputSpeed != outputSpeed))
```

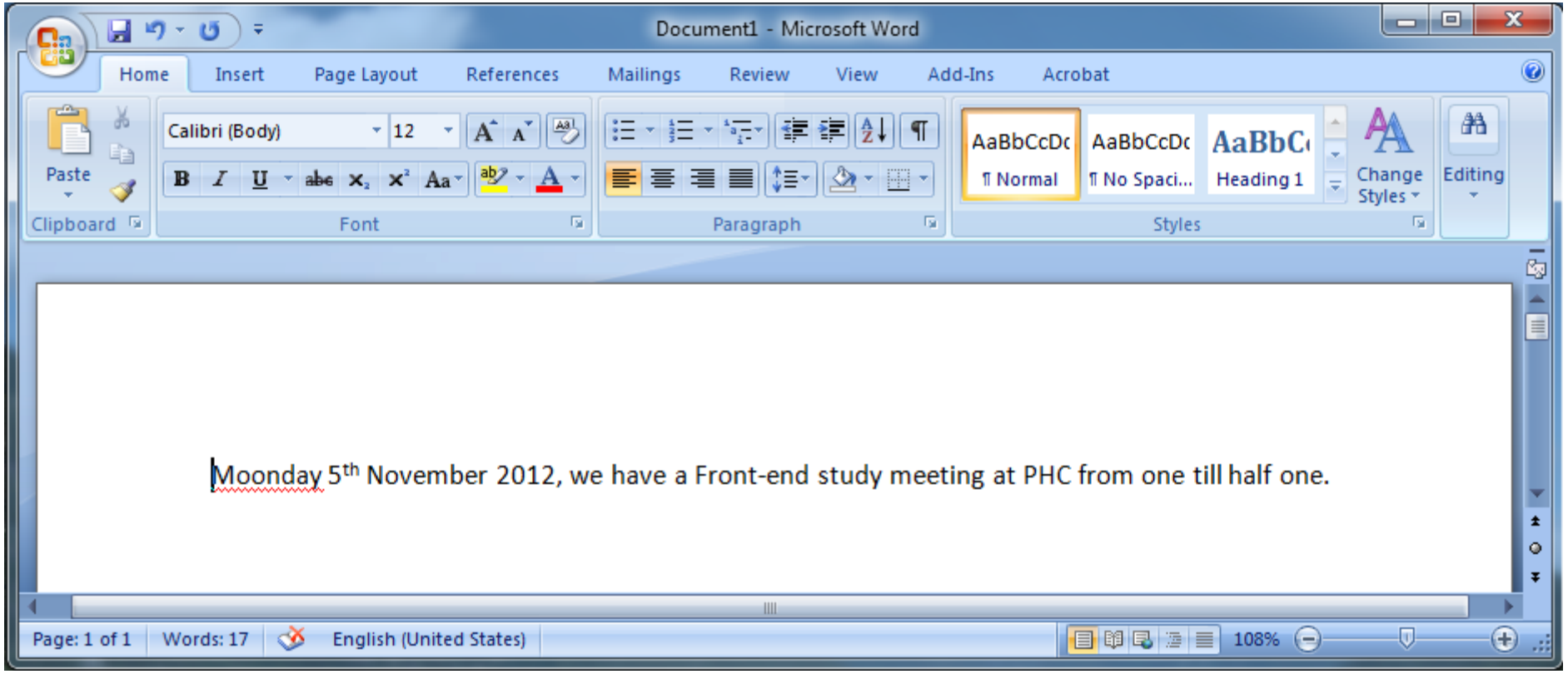
Microsoft Notepad: A Correct Sentence



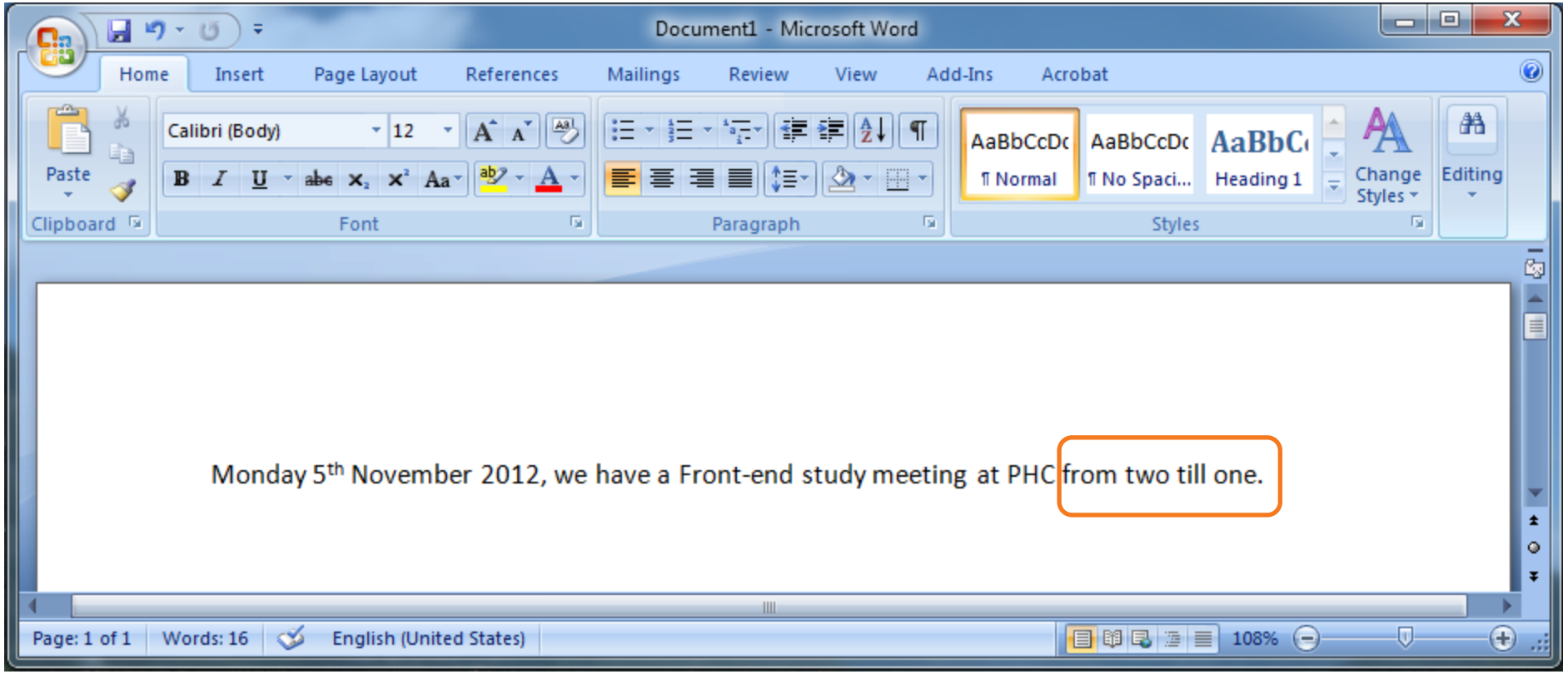
Microsoft Notepad: No Problem Detected, but …



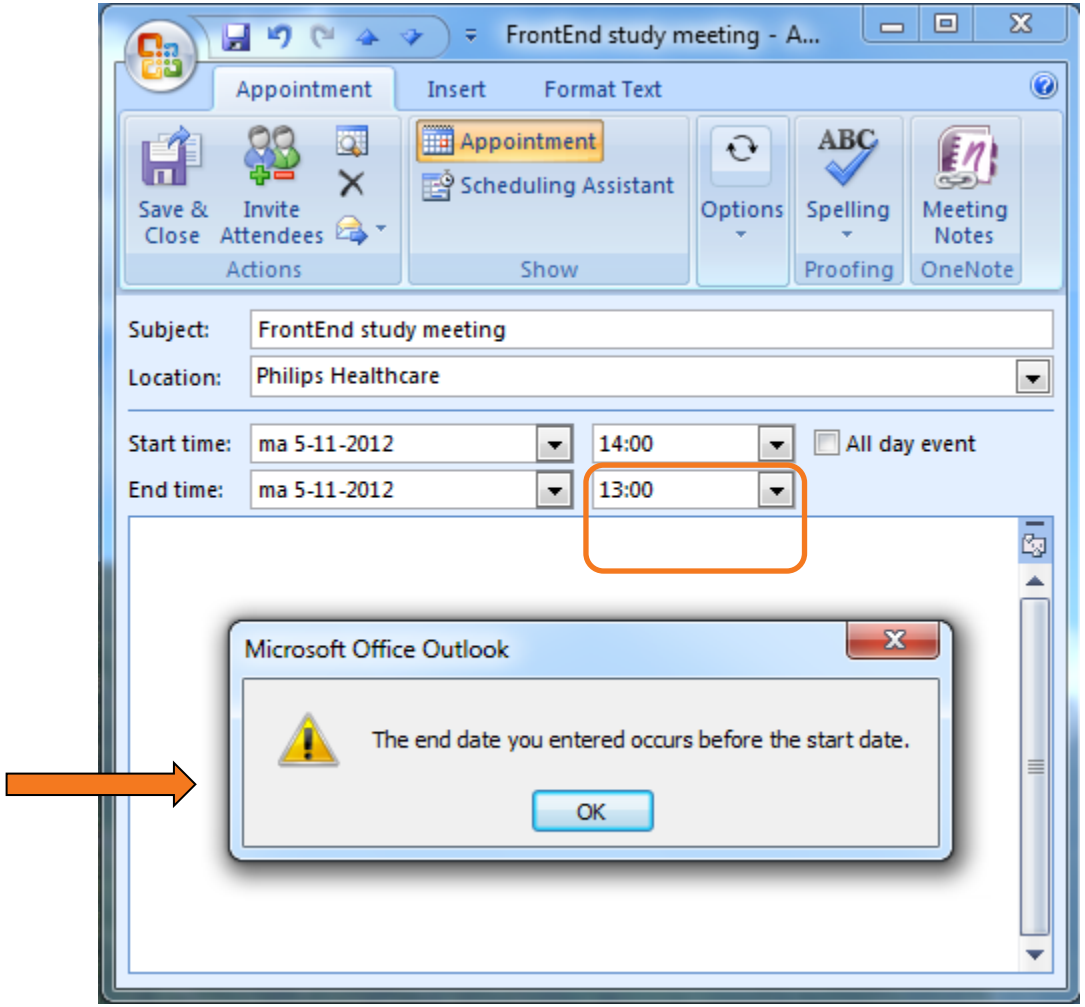
Microsoft Word: Spelling Error



Microsoft Word: No Problem Detected, but …



Microsoft Outlook: Wrong Times



DSL as Central Artifact

```
restriction VeryCloseTableTopAndBeam
    activation
          Distance(TableTop, Beam) < 20 mm
    effects
        userGuidance "TableTop and Beam very close"
        relative limit TableTop*[Rotation, Translation],
                       Beam*[Rotation, Translation]
            at 0


restriction ApproachingTableTopAndBeam
    activation
          Distance(TableTop, Beam) < 35 mm + 15 cm
    effects
        userGuidance "TableTop and Beam approaching"
        relative limit TableTop*[Rotation, Translation],
                       Beam*[Rotation, Translation]
            at (Distance(TableTop, Beam) - 35 mm) / 15 cm
```

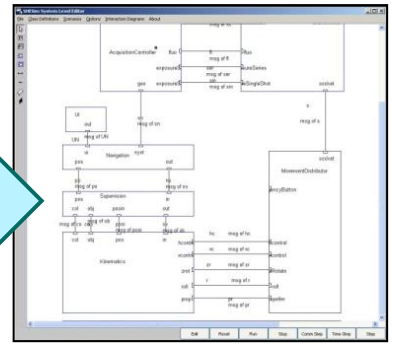
Xtext

Xtend

Validation

Generate documentation

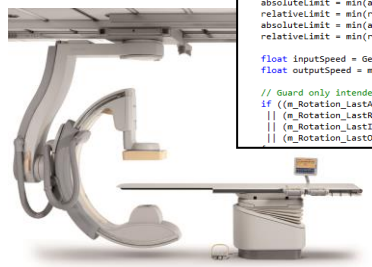
Generate analysis models



Generate source code

factor >30

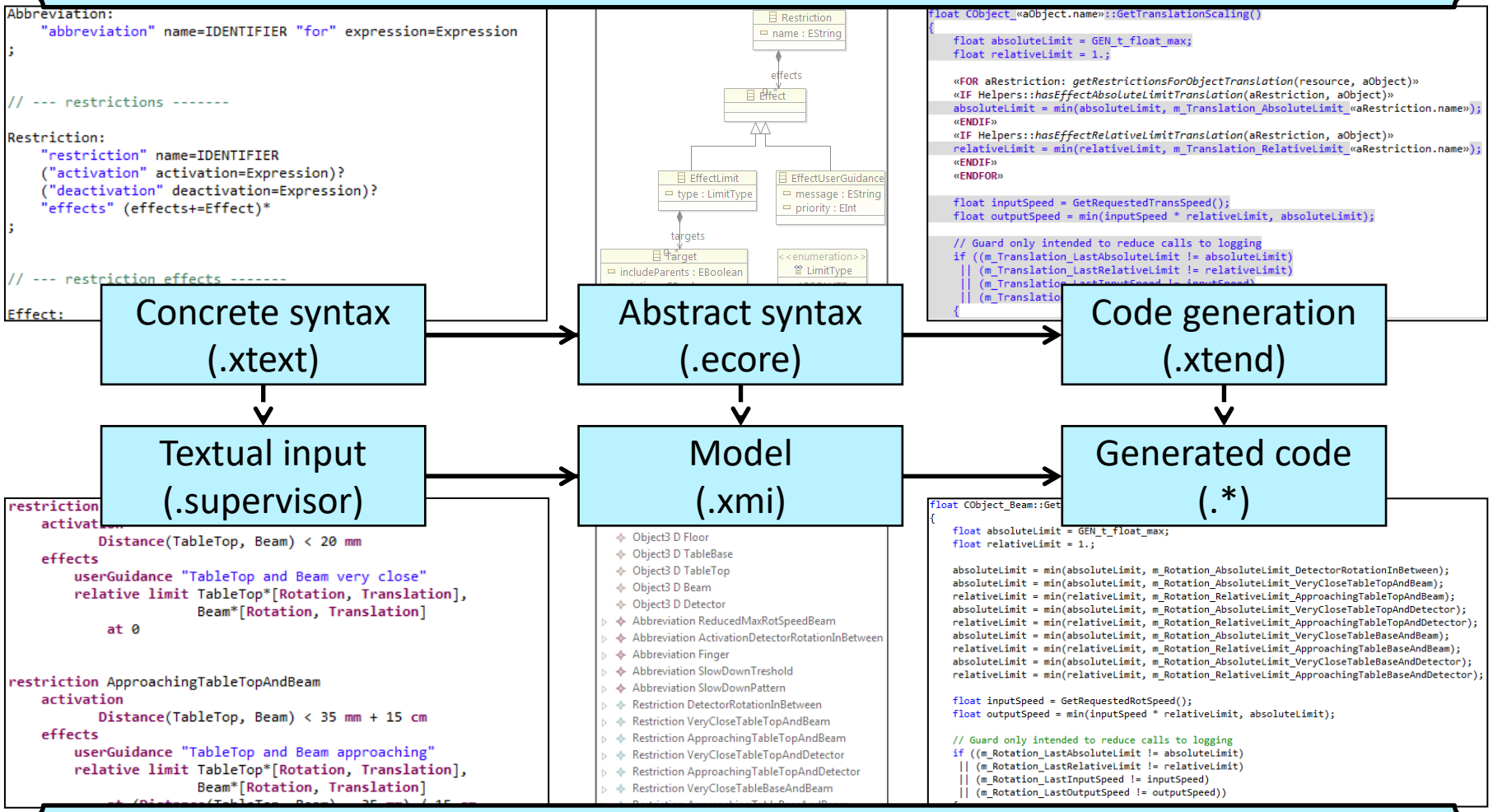
```
float CObject_Beam::GetRotationScaling()
{
    float absoluteLimit = GEN_t_float_max;
    float relativeLimit = 1.;

    absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_DetectorRotationInBetween);
    absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableTopAndBeam);
    relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableTopAndBeam);
    absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableTopAndDetector);
    relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableTopAndDetector);
    absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableBaseAndBeam);
    relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableBaseAndBeam);
    absoluteLimit = min(absoluteLimit, m_Rotation_AbsoluteLimit_VeryCloseTableBaseAndDetector);
    relativeLimit = min(relativeLimit, m_Rotation_RelativeLimit_ApproachingTableBaseAndDetector);

    float inputSpeed = GetRequestedRotSpeed();
    float outputSpeed = min(inputSpeed * relativeLimit, absoluteLimit);

    // Guard only intended to reduce calls to logging
    if ((m_Rotation_LastAbsoluteLimit != absoluteLimit)
     || (m_Rotation_LastRelativeLimit != relativeLimit)
     || (m_Rotation_LastInputSpeed != inputSpeed)
     || (m_Rotation_LastOutputSpeed != outputSpeed))
```



Metamodels and grammars

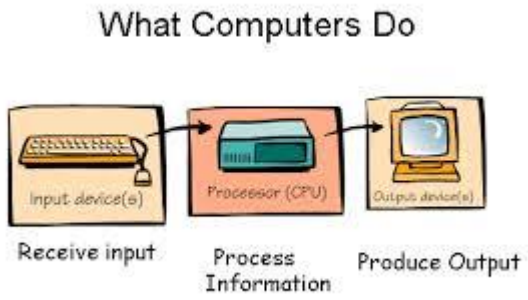
Domain-Specific Languages (DSL)

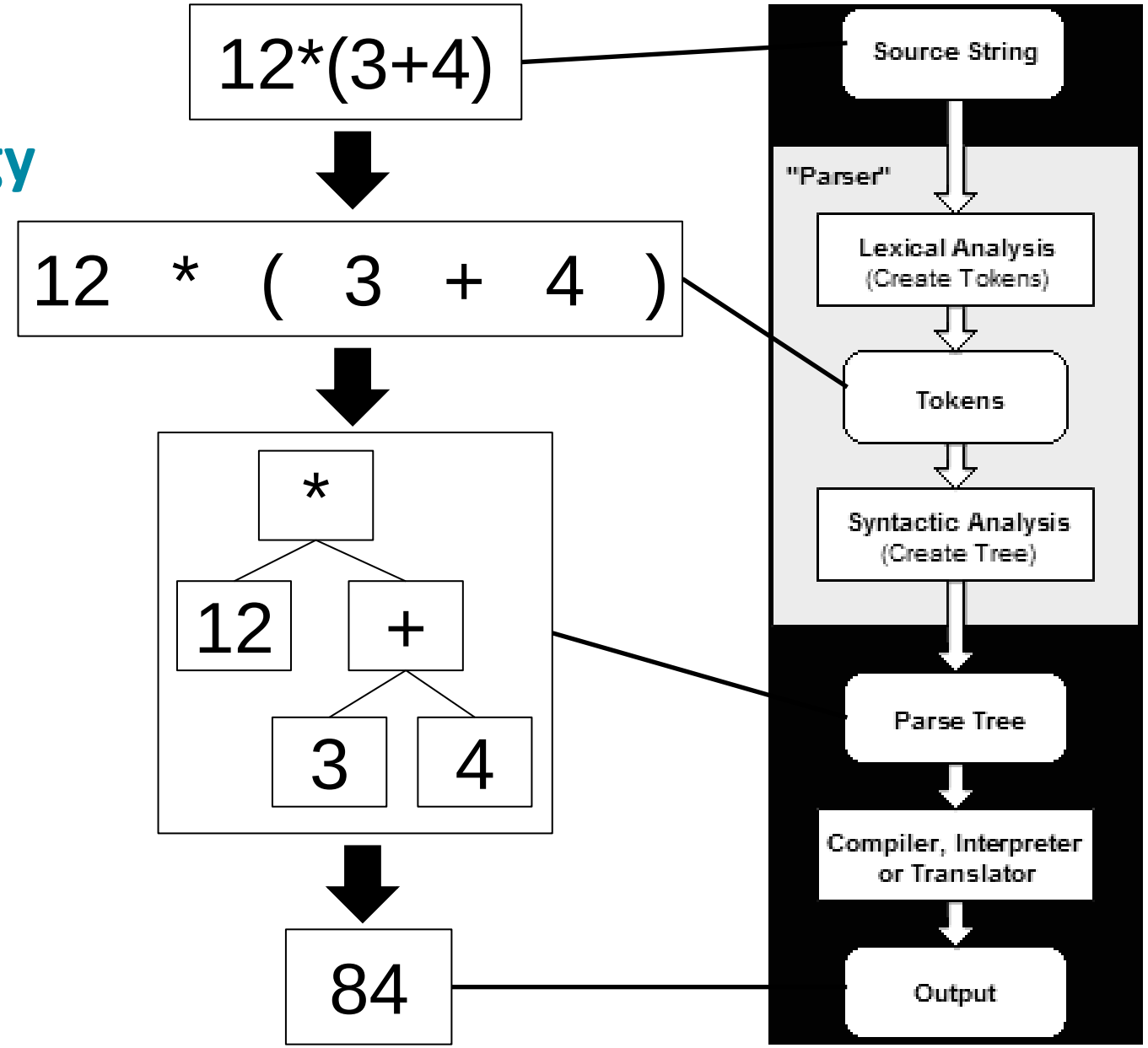
Meta level, for developing the general infrastructure



Instance level, for developing a specific system

Compiler Technology





Lexical Analysis

|	choice
?	optional
*	zero or more times
+	one or more times

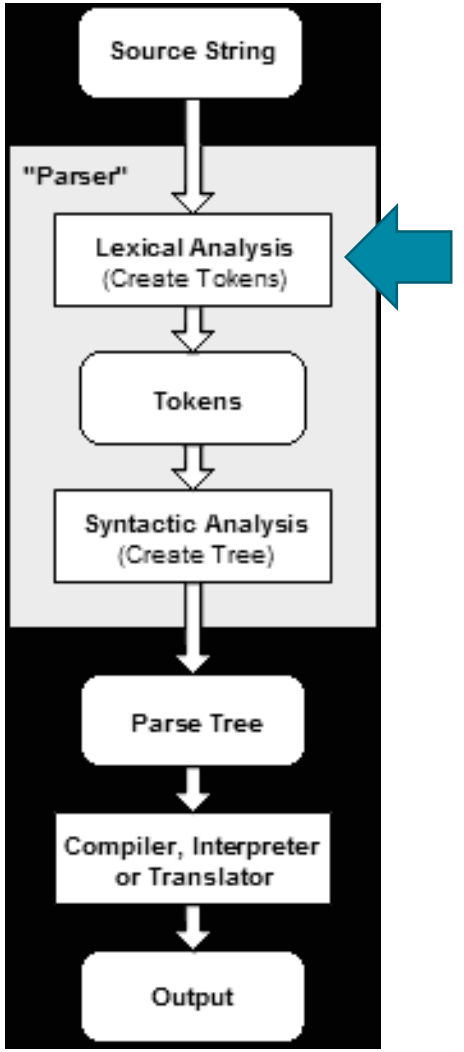
- **Regular expressions using Extended Backus-Naur Form (EBNF)**
  - Literal character sequences

    'while'

    '('
  - Custom terminal definitions

```
terminal INT: ('0'..'9')+;
terminal ID: '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;
```

  - Terminals that are imported by default in Xtext:
    - ID
    - INT
    - STRING
    - ML_COMMENT (= multi-line comment) ← Hidden by default
    - SL_COMMENT (= single-line comment) ← Hidden by default
    - WS (= whitespace, tab, newline) ← Hidden by default



Syntactic Analysis

|	choice
?	optional
*	zero or more times
+	one or more times

Context-free grammars in terms of tokens

- More expressive than regular expressions, e.g., recursion to parse nested brackets

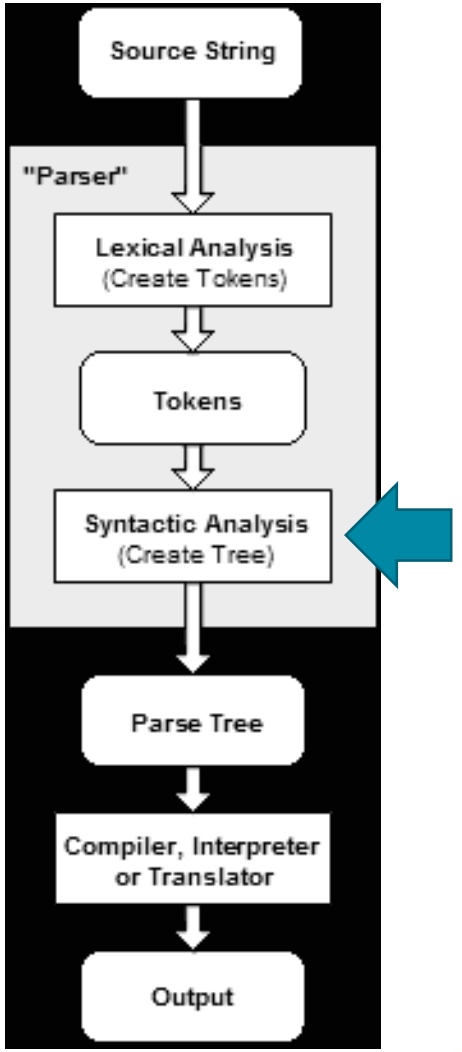
```
Task name AvoidBorder Task freq 100
    Task moves
        Move name Turn Start border End noBorder Speed 15 Rotation 90
        Move name Back Speed 30
```

Robot: (Message | Task)*

Message: 'Display' STRING

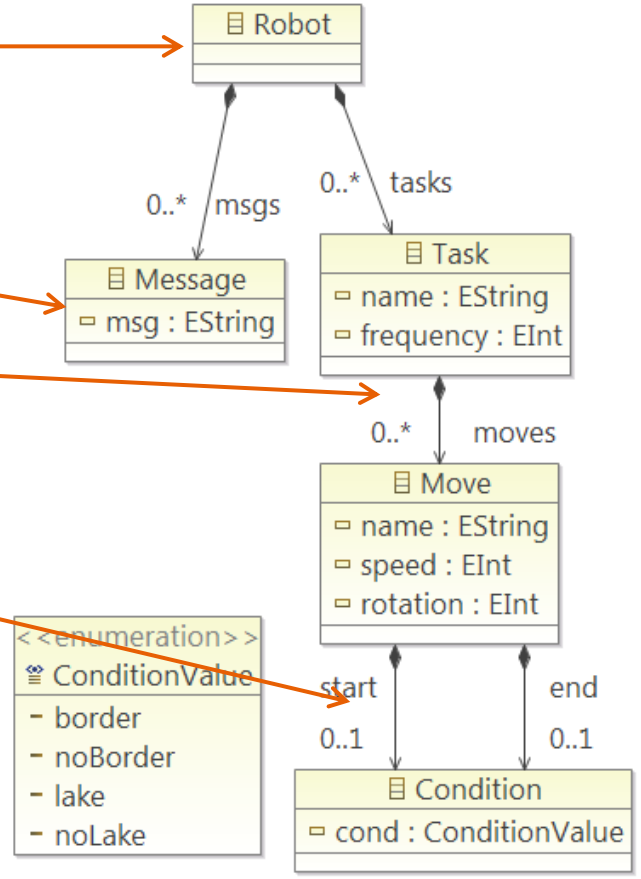
Task: 'Task' 'name' ID ('Task' 'freq' INT)?
'Task' 'moves' (Move)+

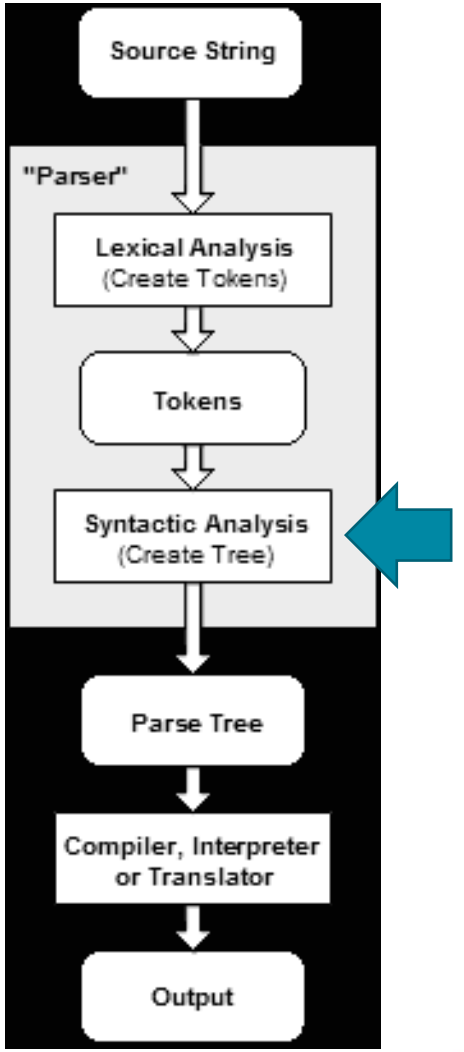
Move: 'Move' 'name' ID
('Start' Condition)? ('End' Condition)?
'Speed' INT ('Rotation' INT)?



Abstract Syntax

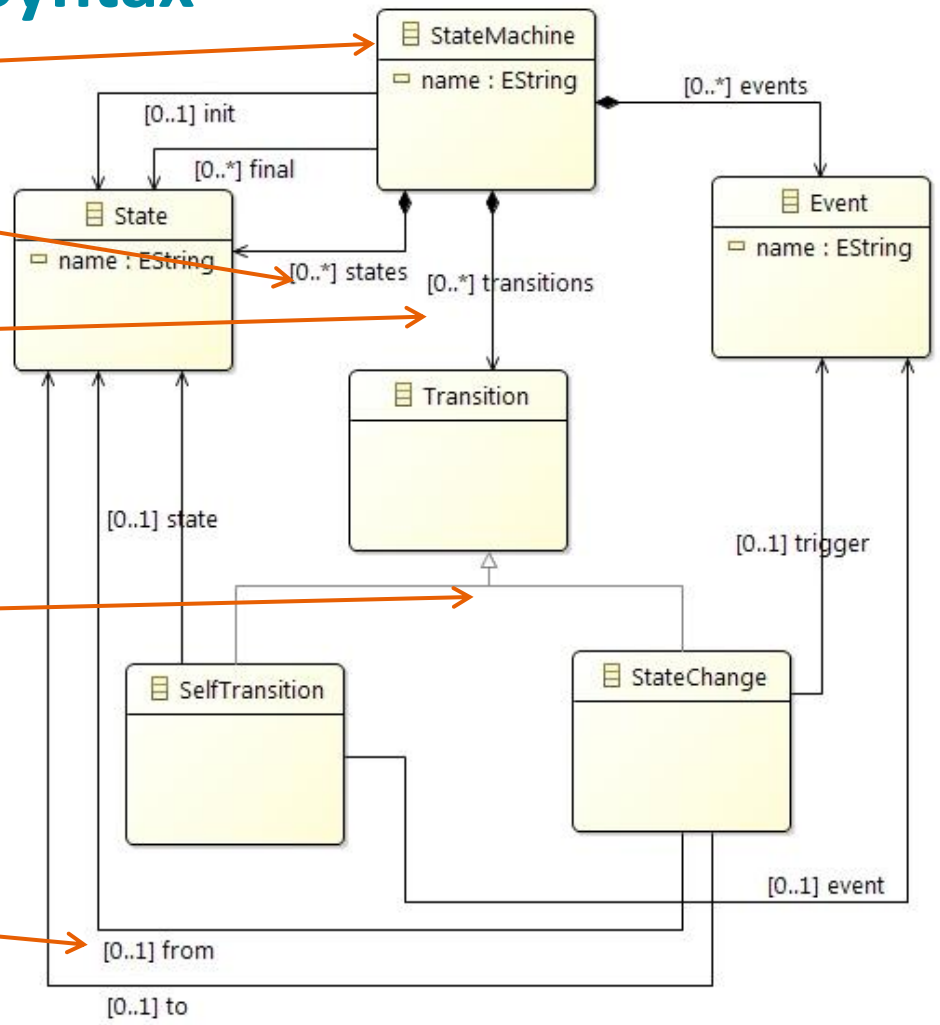
```
Robot:
    (msgs += Message  | tasks += Task)*
;
Message:
    'Display' msg = STRING
;
Task:
    'Task name' name = ID ('Task freq' frequency = INT)?
    'Task moves' (moves += Move)+
;
Move:
    'Move name' name = ID
    ('Start' start = Condition)? ('End' end = Condition)?
    'Speed' speed = INT ('Rotation' rotation = INT)?
;

Condition :
    cond = ConditionValue
;
enum ConditionValue:
    border | noBorder | lake | noLake
;
```

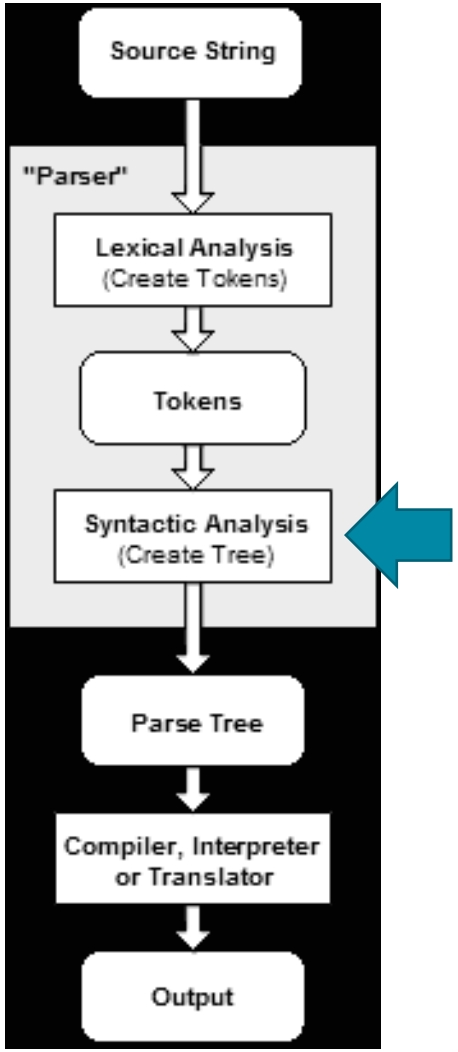




Concrete and Abstract Syntax

```
StateMachine:
    'StateMachine' name = ID
    'Init' init = [State]
    'States' states += State+
    'Events' events += Event+
    'Transitions'
       transitions += Transition*
    'Final' final += [State];

State: name=ID;
Event: name =ID;

Transition:
    SelfTransition | StateChange;

SelfTransition:
    'FromTo' state = [State]
    'Event' event = [Event]
   ;
StateChange:
    'From' from = [State]
    'Trigger' trigger = [Event]
    'To' to = [State];
```

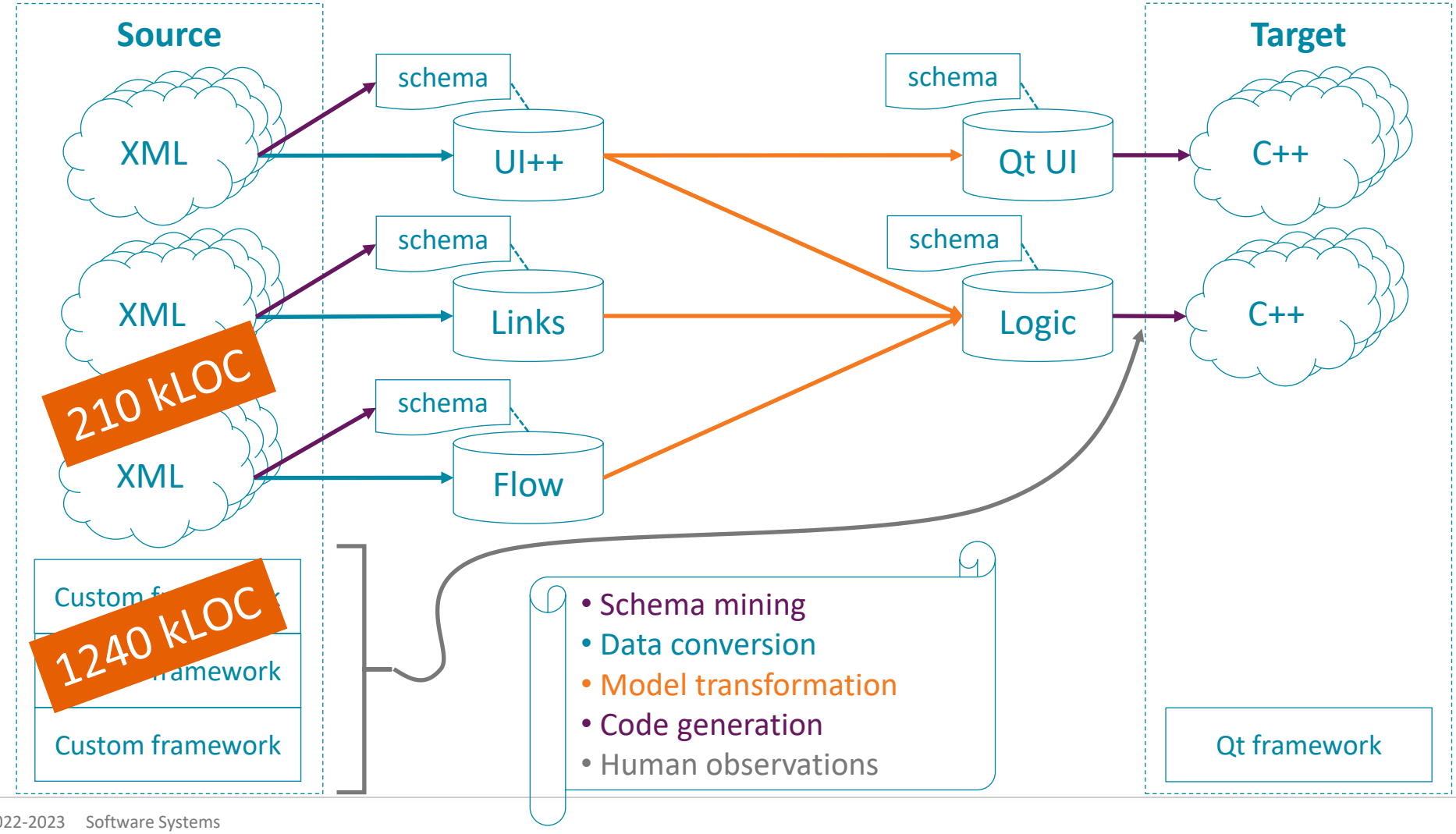




Field-service procedures at Philips IGT

Domain-Specific Languages (DSL)

Application: Migration



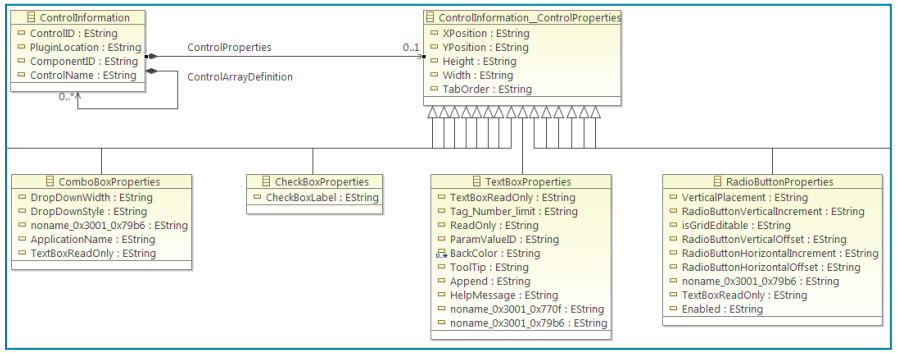
Application: Model extraction

```
<DataObject ObjectType="PanelProperties">
    <Attribute Group="0x3001" Element="0x652c" PMSVR="IInt16">5</Attribute>
    <Attribute Name="XPosition" Group="0x3001" Element="0x7600" PMSVR="IInt16">0</Attribute>
    <Attribute Name="YPosition" Group="0x3001" Element="0x7601" PMSVR="IInt16">145</Attribute>
    <Attribute Name="Height" Group="0x3001" Element="0x7602" PMSVR="IInt16">35</Attribute>
    <Attribute Name="Width" Group="0x3001" Element="0x7603" PMSVR="IInt16">60</Attribute>
    <Attribute Name="TabOrder" Group="0x3001" Element="0x7609" PMSVR="IInt16">2</Attribute>
    <Attribute Name="LayerType" Group="0x3001" Element="0x77d1" PMSVR="IString">"Data"</Attribute>
</DataObject>
```

XML data

↙ ↘

Schema



Converted data

```
ControlProperties = PanelProperties {
    Dock = 5
    XPosition = 0
    YPosition = 145
    Height = 35
    Width = 60
    TabOrder = 2
    LayerType = "Data"
}
```

Application: Code generation

```
// --- node FULLAUTO (stage FullAutomatic) -------
node
    name = "FULLAUTO"
    transitions
        OK -> "STATIC"
        Cancel -> "FINALIZATION"
    stage = FullAutomatic
        part = ""
    UIResource
        assembly = "FSCGeneratorUIDefinitions"
        id = "FSCXGNTubeAdaptationAdjustmentFullAutoUIDef"
    mapping
        event id 21: element "C24"
        event id 20: element "C25"
        event id 10: element "C33" ScrollToLatest
        event id 22: element "C23"
        event id 23: element "ProgressBarCtrl" conversion "TubeLoadPowerFactorToStyleSheet"
        event id 11: element "C31"
        event id 100: element "C9" Append
        event name "ExecutionBegun": element "C35_ProcedureEvents"
        event name "PartCompleted": element "C35_ProcedureEvents"
```

DSL model

```
#include "stdafx.h"
#include "«fileName».h"

#include <QDoubleValidator>
#include <QMessageBox>

#include "Utility.h"

«FOR node : procedure.nodesUI»
«FOR mapping : node.mappingsId»
const ULONG «constant(node, mapping)» = «mapping.id»L;
«ENDFOR»
«FOR mapping : node.mappingsIdEvent»
const ULONG «constant(node, mapping)» = «mapping.id»L;
«ENDFOR»
«FOR action: node.actionsId»
const ULONG «constant(node, action)» = «action.id»L;
«ENDFOR»
«ENDFOR»

«fileName»::«fileName»(QString qstrProcedureName, CLSID component, QWidget *parent, bool bStubbed)
    : «NewFlowGeneratorHeader::parentClass(procedure)»(qstrProcedureName, component, parent, bStubbed)
    «FOR node: procedure.nodesUI»
    , m_qwidget_node«node.name»(NULL)
    «ENDFOR»
```

Template

Generated code

```
#include "stdafx.h"
#include "FlowFSCXGNTubeAdaptationAdjustment.h"

#include <QDoubleValidator>
#include <QMessageBox>

#include "Utility.h"

const ULONG FlowFSCXGNTubeAdaptationAdjustment_Preset_getset_C46 = 1L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C24 = 21L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C25 = 20L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C33 = 10L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C23 = 22L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_ProgressBarCtrl = 23L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C31 = 11L;
const ULONG FlowFSCXGNTubeAdaptationAdjustment_FULLAUTO_event_C9 = 100L;

FlowFSCXGNTubeAdaptationAdjustment::FlowFSCXGNTubeAdaptationAdjustment(QString qstrProcedureName, CLSID component, QWidget *parent, bool bStubbed
    : BaseAdjustment(qstrProcedureName, component, parent, bStubbed)
    , m_qwidget_nodePreset(NULL)
    , m_qwidget_nodeFULLAUTO(NULL)
    , m_qwidget_nodeSTATIC(NULL)
```

Conclusion: It pays off!

Main challenges:

- **Extract the domain knowledge from the legacy software**
- **Avoid that the rejuvenated software resembles the legacy too much**

Industrial results:

- **Redesigned software: 20 times smaller code base than legacy software**
- **Model-based migration: 6 times less effort than manual**

More time for **innovation**!


Dirk Jan Swagerman


Gernot Eggen


Hans van Wezep


Martien van der Meij


Aron van Beurden

PHILIPS

Embedded Systems Innovation BY TNO


Arjan Mooij


Jozef Hooman

Closing remarks

Domain-Specific Languages (DSL)

General Tips and Tricks

- **It may help to first create an example instance, and afterwards create a grammar.**
  - "test-driven"
- **Look at the abstract syntax!**
  - E.g., check missing attribute names
- **Don't be too restrictive in the grammar; validation can be used for extra checks.**
- **Focus on specification (not on execution)**
- **A DSL is not a general-purpose programming language**
- **Use enumeration types when appropriate:**

```
enum ChangeKind :
        ADD     = 'add'
      | MOVE    = 'move'
;
```

Closing remarks

- **Lab session work:**
  - Follow the manual “Creating a Domain Specific Language (DSL) with Xtext” up to section 3.5
    - http://www.cs.ru.nl/J.Hooman/DSL/Creating_a_Domain_Specific_Language_(DSL)_with_Xtext.pdf
  - Modeling assignment
    - Create an Xtext grammar (+ some example instances)
    - Create validation and code generators