Growing an Equation-Based Object-Oriented Modeling Language

4th MODPROD Workshop on Model-Based Product Development
February 10, 2010

David Broman
Department of Computer and Information Science
Linköping University, Sweden
davbr@ida.liu.se

Problem

- Differential Algebraic Equations
- Object-Oriented concepts
- Multi-domain modeling
- Declarative acausal modeling
- Hybrid modeling

Problems

- Large and complex languages → hard to implement
- Many constructs - → hard to reason about
- Backwards compatibility

How should one design a modeling language over time?
Contribution

How should you grow an equation-based object-oriented language, such as Modelica, in a sound manner?

Main contributions of this work
• A systematic way of categorizing different ways of growth of an equation-based object-oriented language.
• Analyzed tradeoffs from different stakeholder’s perspective

Agenda

Part I
Introduction and concepts of syntax and semantics

Part II
Different ways of growing a language

Part III
The right way to grow
Part I

The concepts of syntax and semantics

What does \textit{syntax} of a language mean?

The syntax describes the \textit{structure} of a model

\begin{align*}
\text{if } x > 1 \text{ then } x \text{ else } 0
\end{align*}

\begin{itemize}
\item \textbf{Parse Tree} (Concrete Syntax)
\item \textbf{Abstract Syntax Tree (AST)}
\end{itemize}
What does \textit{semantics} of a language mean?

The semantics describes the \textbf{meaning} of a model.

\begin{verbatim}
if x > 1 then x else 0
\end{verbatim}

Different syntax, but could have the same semantics, e.g. meaning

\begin{verbatim}
if(x > 1){return x;} else {return 0;}
\end{verbatim}

Part I
The concepts of syntax and semantics

Part II
Different ways of growing a language

Part III
The right way to grow

Part II
Different ways of growing a language
Ways of growth matrix

Extending the Semantics

<table>
<thead>
<tr>
<th>Extending the Syntax</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>“growth by adding new language features”</td>
<td>“growth by adding syntactic sugar”</td>
</tr>
<tr>
<td>no</td>
<td>“growth by new meanings of annotations or built-in functions”</td>
<td>“growth by new user defined abstractions”</td>
</tr>
</tbody>
</table>

Extending syntax and semantics

Growth by adding new language features

```
model M
  outer Real x;
  ...
  end M;
model N
  inner Real x;
  M m1, m2;
  ...
  end N;
```

New semantics...

“An element declared with the prefix outer references an element instance with the same name but using the prefix inner which is nearest in the enclosing instance hierarchy of the outer element declaration.”

Components need to have access to other components in the instance hierarchy

New syntax
Inner and outer
Ways of growth matrix

Extending the Semantics

<table>
<thead>
<tr>
<th></th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“growth by adding new language features”</td>
<td>“growth by adding syntactic sugar”</td>
</tr>
<tr>
<td>Extending the Syntax</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>“growth by new meanings of annotations or built-in functions”</td>
<td>“growth by new user defined abstractions”</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

Extending only the syntax

Growth by adding syntactic sugar

**model** M1a
Real x(start=5);
equation
  der(x) = -x + 2;
end M1a;

**model** M1b
Real x;
equation
  der(x) = -x + 2;
initial equation
  x = 5;
end M1b;

**model** M1c
Real x(start = 5,
  fixed=true);
equation
  der(x) = -x + 2;
end M1c;

**model** M1d
Real x(fixed =
  true);
equation
  der(x) = -x + 2;
initial equation
  x = 5;
end M1d;

Are they all describing the same model?

Start attribute

Concrete Syntax

Initial equation section

Concrete Syntax

Do not compile. Over-determined x=0 and x=5 at time 0.
Ways of growth matrix

Extending the Semantics

<table>
<thead>
<tr>
<th>Extending the Syntax</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>“growth by adding new language features”</td>
<td>“growth by adding syntactic sugar”</td>
</tr>
<tr>
<td>no</td>
<td>“growth by new meanings of annotations or built-in functions”</td>
<td>“growth by new user defined abstractions”</td>
</tr>
</tbody>
</table>

Extending only the semantics

Growth by new meanings of annotations and built-in functions

New built-in functions
E.g. cos(x), floor(x), delay(expr, delaytime) etc.

Annotations
Store extra information about models, e.g., graphics, version info, documentation, etc.

Annotation

```
annotation (
  Icon(coordinateSystem(extent=[-100,-100], {100,100})),
  graphics={__NameOfVendor(Circle(center={0,0}, radius=10))})
```
## Ways of growth matrix

<table>
<thead>
<tr>
<th>Extending the Syntax</th>
<th>Extending the Semantics</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>“growth by adding new language features”</td>
<td>“growth by adding syntactic sugar”</td>
<td></td>
</tr>
<tr>
<td>no</td>
<td>“growth by new meanings of annotations or built-in functions”</td>
<td>“growth by new user defined abstractions”</td>
<td></td>
</tr>
</tbody>
</table>

### Part I
The concepts of syntax and semantics

### Part II
Different ways of growing a language

### Part III
The right way to grow

---

**Extending neither syntax nor semantics**

**Growth by new user defined abstractions**

**User defined abstractions**
Functions, models, blocks etc. Encapsulated into reusable libraries

**Important!**
New abstractions should look like primitive constructs in the language

Neither changing the language definition nor the tool implementation.
Ways of growth matrix

Part I
The concepts of syntax and semantics

Part II
Different ways of growing a language

Part III
The right way to grow

Extending the Semantics

<table>
<thead>
<tr>
<th>Extending the Syntax</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>“growth by adding new language features”</td>
<td>“growth by adding syntactic sugar”</td>
</tr>
<tr>
<td>no</td>
<td>“growth by new meanings of annotations or built-in functions”</td>
<td>“growth by new user defined abstractions”</td>
</tr>
</tbody>
</table>

Restricting the language (syntax, semantics or both)

Part III
The right way to grow
Part I
The concepts of syntax and semantics

Part II
Different ways of growing a language

Part III
The right way to grow

**Different stakeholders’ perspective**

**Large group** – continuously adding features

**Small group** – kernel, syntactic sugar, missing user input

**Language Designers**

**End Users**

**Library Users**

**Tool Vendor**

Handle everything – what to leave out?

Easy to use and understand

Clear semantics, e.g. Initial equations

Detect and locate errors. Restrictions necessary.

Interpret the specification unambiguously.

Commercial perspective. Conflict of interest with language design. Not always want to be 100% compatible.

**Expressiveness...**

Ideal, grow without changing syntax or semantics

**Concluding Summary**

Minimize or avoid, especially with mature and widely used languages

Most drastic change

Library developers are pushing, language designers and tool vendors negatively affected

Extending the Syntax

Extending the Semantics

Yes

“growth by adding new language features”

“growth by adding syntactic sugar”

No

“growth by new meanings of annotations or built-in functions”

“growth by new user defined abstractions”

A preferable approach. Clearer semantics for users and easier to implement correctly for tool vendors.

Hard to execute in a large design group

The preferable approach.

Very hard for young and new kind of languages, e.g. EOO language Modelica

Dangerous. Incompatible models which are tool dependent.

**Part I**
The concepts of syntax and semantics

**Part II**
Different ways of growing a language

**Part III**
The right way to grow

Available from: www.ida.liu.se/~davbr