Languages and Models of Computation (Revisted)

Peter Marwedel
TU Dortmund
Informatik 12
Germany
Other languages

- **Estelle**: Designed to describe communication protocols; scope similar to SDL; unification of both failed.
- **LOTOS, Z**: Algebraic specification languages
- **Silage**: functional language for digital signal processing.
- **Rosetta**: Efforts on new system design language
- **Esterel**: reactive language; synchronous; all reactions are assumed to be in 0 time; communication based on ("instantaneous") broadcast; //www.esterel-technologies.com
- **IEC 60848, STEP 7**: Process control languages using graphical elements
Levels covered by the different languages

- Requirements
- Architecture
- HW/SW
- Behavior
- Functional Verification
- Test bench
- RTL
- Gates
- Transistors

Languages:
- Verilog
- VHDL
- SystemVerilog
- Vera
- Sugar
- Jeda
- System C
- Matlab
## Language Comparison

<table>
<thead>
<tr>
<th>Language</th>
<th>Behavioral Hierarchy</th>
<th>Structural Hierarchy</th>
<th>Programming Language Elements</th>
<th>Exceptions Supported</th>
<th>Dynamic Process Creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateCharts</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>VHDL</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SpecCharts</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>SDL</td>
<td>+-</td>
<td>+-</td>
<td>+-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Petri nets</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Java</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SpecC</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>SystemC</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>- (2.0)</td>
<td>- (2.0)</td>
</tr>
<tr>
<td>ADA</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Comparison of languages demonstrated

<table>
<thead>
<tr>
<th>Communication/local computations</th>
<th>Shared memory</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Synchronous</td>
</tr>
<tr>
<td>Communicating finite state machines</td>
<td>StateCharts</td>
<td>SDL</td>
</tr>
<tr>
<td>Data flow model</td>
<td>Not useful</td>
<td>Kahn process networks</td>
</tr>
<tr>
<td>Von Neumann model</td>
<td>C, C++, Java</td>
<td>C, C++, Java with libraries CSP, ADA</td>
</tr>
<tr>
<td>Discrete event (DE) model</td>
<td>VHDL, Verilog, SystemC</td>
<td>Only experimental systems, e.g. distributed DE in Ptolemy</td>
</tr>
</tbody>
</table>
Observations

- Useful to consider model of computation matching with the class of applications considered
- Nevertheless, feasible to simulate one model in another model (e.g. most models are simulated with imperative computing Ptolemy)
- Commercial products available:
  - DSpace (Paderborn): Synthesis of control software for the power train in cars
  - Esterel Technologies (Toulouse): Synthesis from synchronous language SCADE (used by Airbus)
How to cope with language problems in practice?

Mixed approaches:

- (RT-) UML or equivalent
- SDL
- C-programs
- Assembly programs
- Objectcode

- (RT-) UML or equivalent
- VHDL
- Net list
- hardware
- Objectcode

- (RT-) Java
Models of computation in Ptolemy

1. Finite state machines
2. Communicating sequential processes
3. Discrete event model
4. Distributed discrete event model
5. Process networks, including Kahn process networks
6. Synchronous dataflow (SDF)
7. Continous time
8. Synchronous/reactive models
UML and Model Driven Design

Peter Marwedel
TU Dortmund
Informatik 12
Germany
Higher levels of abstraction

- For many applications, it is desirable to start modeling at a level higher than the (Java-, ADA-, SDL-code level)
- Special modeling mechanisms have been designed for these cases
- In particular, the unified modeling language (UML) – building on concepts already presented - is designed to cover this requirement

Elements
(Message) Sequence Charts (MSC)

Graphical means for representing schedules; time used vertically, geographical distribution horizontally.

No distinction between accidental overlap and synchronization.
Time/distance diagrams as a special case

© www.opentrack.ch
Example 1: establishing an (old-fashioned) modem connection

According to Stallings
Example 2: setting up a TCP connection

TCP - Transmission Control Protocol (TCP Basic Handling)

Client Node | Internet | Server Node
-------------|----------|------------------
Client       | Net      | Server           
Client App   | Client Socket | Network | Server Socket | Server App

LEG: About TCP

TCP (Transmission Control Protocol) provides a reliable end to end service that delivers packets over the Internet. Packets are delivered in sequence without loss or duplication.

Server Application creates a Socket
The Socket is created in Closed state
Server sets the initial sequence number to 100
Server application has initiated a passive open.
In this mode, the socket does not attempt to establish a TCP connection. The socket listens for TCP connection request from clients
Socket transitions to the Listen state

Client Application creates Socket
The socket is created in the Closed state
Initial sequence number is set to 0
Application wishes to communicate with a destination server using a TCP connection. The application opens a socket for the connection in active mode. In this mode, a TCP connection will be attempted with the server.
Typically, the client will use a well known port number to communicate with the remote Server. For example, HTTP uses port 80.

LEG: Client initiates TCP connection

Client initiated three way handshake to establish a TCP connection

Client sets the SYN bit in the TCP header to request a TCP connection. The sequence number field is set to 0. Since the SYN bit is set, this sequence number is used as the initial sequence.
(Message) Sequence Charts

PROs:
- Appropriate for visualizing schedules,
- Proven method for representing schedules in transportation.

CONS:
- describes just one case, no timing tolerances: "What does an MSC specification mean: does it describe all behaviors of a system, or does it describe a set of sample behaviors of a system?"

Use in UML

Heavy usage of MSCs in UML (known as sequence diagram);

No precise timing.

Many kinds of additional elements

UML diagrams

From: www.sdmagazine.com/documents/s=815/SDM0012C/

© p. marwedel, informatik 12, 2008
State machine diagrams (UML 2.x)
State diagrams (UML 1.x)

State machine diagrams/State diagrams:
UML includes variant of StateCharts
Activity diagram

Extended Petri nets. Include decisions (like in flow charts). Graphical notation similar to SDL.

© Cris Kobryn: UML
2001: A Standardization Odyssey, CACM, October, 1999
Activity diagram (2)

Parent service of SOAP connector

[send]

SOAP Connector

- transcode [yes] → transcode
- transcode [no]
- encrypt [yes] → encrypt
- encrypt [no]
- Create SOAP header
- Fill in SOAP body
- Send message & receive result
- decrypt [yes] → decrypt
- decrypt [no]
- transcode [yes] → transcode
- transcode [no]
Activity diagram (3)
Deployment diagram

Example including some details:

Describe execution architecture of systems (HW or SW). Important for embedded systems.

Deployment diagram
- More concise example -

© Scott Ambler,
Agile Modeling,
Use case diagram

Captures typical application scenarios

Correct

Customer

open account
deposit funds
withdraw funds
close account


//sds.hss.cmu.edu/courses/Syllabi/ids/271/umlfaq.asp#ucdefinition
Package diagram

Represents the partitioning into packages. Introduces hierarchy.

Example: Use case package diagram.

Class diagrams

Describe object classes. Example:

Timing diagrams

Can be used to show the change of the state of an object over time.

.. model the business software architecture, the technical software architecture, .... Physical architecture issues, in particular hardware issues, are better addressed via UML deployment diagrams..

Represent components used in applications:

Additional diagrams

- **Communication diagram**
  (called collaboration diagram in UML 1.x)
- **Object diagrams**
- **Interaction overview diagrams**
- **Composite structure diagrams**

Less frequently used
Evaluation

Precise specification of semantics?
Typically combined with some other “precise” language
UML for real-time?

Initially not designed for real-time.
Lacking features (1998):

- Partitioning of software into tasks and processes
- specifying timing
- specification of hardware components
- Projects on defining real-time UML based on previous work
- ROOM [Selic] is an object-oriented methodology for real-time systems developed originally at Bell-Northern Research.
## UML Profiles Relevant for SoC

<table>
<thead>
<tr>
<th>Existing (OMG)</th>
<th>Upcoming (OMG)</th>
<th>non-OMG</th>
</tr>
</thead>
</table>
| ▪ SPT (Schedulability, Performance, and Timing Analysis)  
▪ Testing Profile  
▪ QoS and Fault Tolerance  
▪ SysML (System Modelling Language)  
▪ UML Profile for SoC | ▪ MARTE (Modeling and Analysis of Real-Time Embedded Systems) | ▪ UML/SystemC (STMicroelectronics)  
▪ SPRINT Profile (ST, NXP, Infineon, …) |

SoC = system on a chip
Example: Activity diagram with annotations

See also W. Müller et al.: UML for SoC, http://jerry.c-lab.de/uml-soc/

Figure 8-10  Details of the “send video” subactivity with performance annotations
UML Profile Summary

- UML Profile comes as class diagrams with constraints, textual outlines (semantics), icons, diagram symbols, ...
  Constraints and behavioral semantics typically leave several issues open (variation points)

- Different OMG profiles of related domains may not be compatible!

- Current OMG UML Profiles are mainly for modelling

- UML Profiles do not come with a formal semantics
- … but Hardware Design is not just modelling
- HW verification and synthesis requires a well-defined and precise behavioral semantics

- Several UML tools already support UML profile definition
Summary

- Message sequence charts
  - Original (ITU) version, Time-Distance diagrams
  - Sequence diagram in UML
- UML
  - State machine diagram (StateChart-like)
  - Activity diagram (extended Petri nets)
  - Deployment diagram (exec. arch.),
  - Use case diagram
  - Package diagram (hierarchy), Class diagrams,
  - Timing diagrams (UML 2.0), UML for real-time?