The Specification and Description Language SDL

and what comes next ...

Dr. Christian Schröder
Professional Services, Germany
christian.schroeder@telelogic.de
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What's next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Who is Telelogic?

Advanced software and systems
Global business

• Presence in **18** countries
• Over **40** offices
• Over **1100** employees
• Local expertise **worldwide**
Who is Telelogic?

Customer focus

• Customers:
  – With large-scale, complex projects and distributed development teams

• Solutions:
  – Market leading products
  – Specialist services knowledge
  – Expertise developed through industry leadership
    • 3GPP, OMG, Bluetooth SIG, IEEE, INCOSE
  – Optimized to customer needs
Who is Telelogic?

Demonstrable success

© Telelogic AB
Who is Telelogic?

Telelogic Products

Telelogic DOORS
Requirements Management

Telelogic Tau
Visualization Software Engineering

Telelogic CM Synergy
Change Management

Analyze Design Test
Who is Telelogic?

 Tau SDL Suite

Telelogic DOORS
Requirements Management

Telelogic Tau

Visual Software Engineering

Telelogic CM Synergy
Change Management

Analyze, Design, Test
Who is Telelogic?

SDL Suite

Purpose
• SDL Suite is a real-time software development tool based on the object oriented design language **SDL** and the trace language **MSC**

Differentiators
• Mature and feature rich tool with many advanced features (validation, coverage viewer, type viewer, ... )
• High quality code generation
• Powerful simulation
• Strong telecom background
• Languages used in communication standards - SDL, MSC, ASN.1
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
What is the Specification and Description Language (SDL)?
Positioning – Application Area

- Algorithmically
- Very high real time demands
- Very high performance
- Assembler focus
- Extreme memory limits

- Communication intensive
- High real time demands
- Complex systems
- Development based on telecom standards
- C focus
- Small to large target environments

- Data intensive
- Limited real time focus
- Very high C++ or Java focus
SDL – Application Characteristics

- Real-time
- Communicating
- Event-driven
- Interactive
- Concurrent
- Complex
- Distributed
- Mission-critical
SDL – Application Areas

- intelligent networks
- datacoms
- network management
- switching
- access networks
- services
- terminals
- protocols
- GSM
- UMTS
- automotive
- aerospace
Specification and Description

A specification of a system is the description of its required behavior.

A description of a system is the description of its actual behavior.

(Section 1.1, Z.100)
The Specification and Description Language

- International Standard (ITU-T, Z.100)
- Formal description technique
- Object oriented language
- Independent of design paradigm (function or object oriented)
- Independent of implementation (language, operating system, and hardware)
SDL Objectives

• According to Z.100 SDL should be a language that is:
  
• easy to learn
• easy to use
• easy to interpret

„The graphical representation gives a language that is easy to understand both for creators (direct users) and viewers (“non constructors” of specifications).“
The representation technique is formal if the technique’s interpretation model is formally defined; guaranteeing that no ambiguities can occur.

i.e.

an ambiguity based on the interpretation model is an error and can automatically be detected.
SDL History

- **SDL-76**  First version. Only recommendations on how to draw process graph symbols (*Behavior*).
- **SDL-80**  Blocks are introduced (*Architecture*), PR-form (textual Phrase Representation) becomes a part of the language.
- **SDL-84**  The abstract data type concept is introduced (*Data*). Additional concepts are introduced.
- **SDL-88**  Only minor changes
- **SDL-92**  *Object-oriented* extensions to SDL
- **SDL-96**  Minor changes (e.g. external procedures)
- **SDL-2000**  Major changes regarding architecture and data model (towards a harmonization with UML).
## SDL Language Concepts - Four Components

<table>
<thead>
<tr>
<th>Attribute</th>
<th>SDL Construct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>system, block</td>
</tr>
<tr>
<td>Behavior</td>
<td>processes</td>
</tr>
<tr>
<td>Communication</td>
<td>signals and channels</td>
</tr>
<tr>
<td>Data</td>
<td>abstract data types</td>
</tr>
</tbody>
</table>

... plus object oriented concepts
Architecture - Hierarchy - Abstraction
Behavior – Application and Target Language Independent

AccessControl
- Opened
  - Close
    - ‘close door’
  - Closed
    - Closed

Production
- Stocks Empty
  - Material
    - ‘produce’
  - Ready
    - Stocks Full

Process
- State
  - Event
    - ‘Action’
  - Result
    - New State
Communication – Horizontal and Vertical Signal Flow
Object Oriented Concepts – promoting abstraction and reuse

Type

Door

Instances

EntranceA: Door
ExitB: Door
.... Door
Message Sequence Charts MSC – specifying and tracing scenarios
Aim
General Structural Concepts

• The architecture of SDL systems permits the description of different levels of abstraction:
  • The relation between the system and the ‘outside world‘.
  • Block interaction.
  • Process interaction.
  • Behavioral description in processes and procedures.
System Boundaries

– In the first step the relationship between the modeled system and its environment must be defined.
Architecture

- A complete SDL system must contain at least one block.
- A block must contain at least one process or block. Eventually, the “leaves” of this tree are the processes.
- *Blocks and processes must not be mixed in one block.*
- The behavior of the system should be described inside the processes.
Communication

The communication is defined by:

- Channels and signal routes
- Signals and signal parameters (data)

```plaintext
SIGNAL
  open,
  close,
  code, /* textual information for the user */
  display;
```
System Specification

– A complete system specification contains all levels of abstraction, and all properties of the objects defined (modeled) at each level.
Behavior

– Dynamic behavior is modeled with processes based on finite state machines (FSM)

• Abstract state machines that at any moment can be in a state, receive signals or send out signals.

  • In SDL the concept is extended with data processing
  • “Send no wait”, asynchronous signalling
  • Process instances can be created and terminated dynamically
Basic Process Symbols

- Start
- State
- Input
- Output
Example: UML state diagrams (state oriented)
SDL process diagram (transition oriented)
The Software Development Paradigm Shift

**Traditional** development

- “Labor intensive”
- Non-interworking tools
- Lower productivity
- Hard work!

**Visual** development

- “Brain intensive”
- Single environment
- High productivity
- Fun!
“The diagrams are the code”

typedef basic_istream<__E, __Tr> _Myis;
ios_base::iostate _St = ios_base::goodbit;
bool _Chg = false;
_X.erase();
const _Myis::sentry _Ok(_I);
if (_Ok)
  (const _Ctype& _Pac = _USE(_I.getloc(), _Ctype);
   _TRY_IO_BEGIN
   _A:;size_type _N = 0 < _I.width()
   && _I.width() < _X.max_size()
   ? _I.width() : _X.max_size();
   _Tr:;int_type _C = _I.rdbuf() -> sgetc();
   for (; 0 < _N; _C = _I.rdbuf() -> sgetc())
     if (_Tr:;eq_int_type(_Tr:;eof(), _C))
       (_St |= ios_base::eofbit;
        break;)
     else if (_Fac.is(_Ctype::space,
       _Tr:;to_char_type(_C))
        break;
     else
       (_X.append1(_I, _Tr:;to_char_type(_C));
        _Chg = true;)
   _CATCH_IO(_I, _I);)
   _I.width(0);
if (!_Chg)
  _St |= ios_base::failbit;
if (_Ok)
  (const _Ctype& _Pac = _USE(_I.getloc(), _Ctype);
   _TRY_IO_BEGIN
   _A:;size_type _N = 0 < _I.width()
   && _I.width() < _X.max_size()
   ? _I.width() : _X.max_size();
   _Tr:;int_type _C = _I.rdbuf() -> sgetc();
   for (; 0 < _N; _C = _I.rdbuf() -> sgetc())
     if (_Tr:;eq_int_type(_Tr:;eof(), _C))
       (_St |= ios_base::eofbit;
        break;)
     else if (_Fac.is(_Ctype::space,
       _Tr:;to_char_type(_C)))

Textual programming Visual programming
Hierarchical Architecture
Modeling Distributed Systems

1. Channel definitions
   - EstwEingabe, EstwFahrstrasse, EstwAusgabe, Sensorik

2. Signaling definitions
   - BueFreimeldung, BueFestlegung, StreckeBelegt, (EstwKommandos), (EstwMeldungen), ...
Modeling Concurrency

1. Encapsulation of parallel Automata:
   SsBueKern
   SsBue_Teilsystem

2. Refinement of interfaces:
   VonBuesa,
   VonBk,
   BkAnzeige,
   Anzeige,
   ZurBuesa,
   Estw,
   Fahrstrasse
Specifying Behavior, implementing functionality
**SDL = Communication Oriented Extended Finite State Machine**
A Formal Language

1. Syntactic Analysis
2. Semantic Analysis
3. Error correction
Verification & Validation

1. Bit State Exploration
2. Random Walk
3. Power Walk
4. Exhaustive
5. Tree Search
6. MSC-Verification
7. Navigator Walk
Simulation helps avoiding errors!
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• **SDL for ages 11 and up**
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Programming a LEGO Robot with SDL

- LEGO MINDSTORMS
- Developed for ages 11 and up
- RCX™ Microcomputer
- CD-ROM Software
- 717 pieces, including:
  - 2 Motors
  - 2 Touch Sensors
  - 1 Light Sensor
  - Infrared Transmitter
The RCX

- Hitachi H8300 micro controller
- 32kb off-chip RAM
- 3 input ports
- 3 output ports
- LCD display panel
- Control buttons
- Infrared communications port
First, the OS has to be downloaded via IR communication:
```
firmdl3 -s LegOS.srec
```
The targeting expert automatically compiles, links and downloads the SDL system onto the robot. For the download via IR communication the targeting expert uses dll.exe.
Agenda

- Who is Telelogic?
- SDL in a Nutshell
- SDL for ages 11 and up
- Wrap Up
- What’s next?
  - SDL 2000, UML 2.0
- Telelogic and the Standards
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Wrap Up – SDL

- industry proven language for the development of complex distributed and event driven systems.
- particularly suited for the specification of concurrent and communicating real-time processes based on an EFSM formalism.
- defined by a graphical as well as a textual representation.
- formal language and thus can be used for complete automatic code generation
- independent of any implementation language
- language Standard (currently SDL-2000, ITU-T, Z.100)
- intuitive to use and read
- Tool support for code generation, simulation and validation
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Agenda

- Who is Telelogic?
- SDL in a Nutshell
- SDL for ages 11 and up
- Wrap Up
- What’s next?
  - SDL 2000, UML 2.0
- Telelogic and the Standards
UML and SDL - Merging languages

Now

- Analysis and high-level design
- Real-time design and simulation

UML

2001/2002

- Analysis and Data
- Behavior and structure
- Seamless real-time development workflow

UML 2.0/SDL2000

MSC

SDL
What benefits did we get with UML 1.x?

- UML is a tool for communication!
- Programmers, customers, managers, domain experts will have a common understanding of the system functionality
What benefits did we get with UML 1.x? (cont.)

• Most systems being modeled are so **complex** that a single kind of diagram cannot clearly and completely describe the system.

• Because of this, UML provides different diagram types, each type offering a **different view or perspective** on the system.

(picture taken from: Douglas R. Hofstadter: Gödel, Escher, Bach: An Eternal Golden Braid)
What are the weaknesses of UML 1.x?

- UML 1.x is a fine language for what it was designed for
  - analysis and modeling of software.
- But there are a number of problems with UML 1.x:
  - excessive size and gratuitous complexity
  - imprecise semantics
  - non-standard implementations
  - limited customizability
  - lack of support for component-based development
  - inadequate semantics for executable models
  - missing diagram interchange capability
- As technology evolves, and the requirements for large, complex systems become more demanding, UML also has to evolve.
What will UML 2.0 improve?

- Better support for developing component-based software
- Better support for modeling architecture of software
- Better alignment with other popular standards, such as XML/XMI, SDL and MSC.
- More powerful expressions, including major parts of SDL and MSC.
- Better possibilities to build tools with simulation and code generation
- Better support for executable models and dynamic behavior
- Better diagram interchange between tools
- Better scalability

UML 2.0 is a great leap forward
How will UML 2.0 be created?

- The **U2 Partners** ([www.u2-partners.org](http://www.u2-partners.org)) group including Alcatel, CA, Ericsson, HP, IBM, Motorola, Oracle, Rational and Telelogic is submitting proposals for UML 2.0.
How will UML 2.0 be created (cont)?

- The first major proposal was submitted to **OMG** in August 2001.
- After that, OMG will ratify the new standard: **UML 2.0**.
UML 2.0* news preview: Architecture diagrams

- Defines **logical** components
- Hierarchical decomposition
  - divide and conquer (top down)
  - building blocks (bottom up)
- Provides encapsulation
  - black-box approach for just the right level of detail
- Enables component-based development
  - protect, re-use and share knowledge
- Maps to physical components
  - EJB, COM+, ...
- Equivalent to **SDL-2000** block diagrams.

*according to latest submissions to OMG*
UML 2.0* news preview:
Statecharts with graphical transitions

- Powerful definition of behavior.
- Graphical transitions visualize activities that take place during transitions.
- Precise specifications without use of implementation language ensures portability and reuse.
- Better scalability ensure sufficient support for industrial-scale complexity.
- Equivalent to SDL-2000 state diagrams.

*according to latest submissions to OMG
UML 2.0* news preview: Sequence diagrams

- Describes the sequence of events in an easy-to-understand format
- Modularity and scaleability support reuse of design.
- Better formal foundation enables analysis of complex applications and industrial large-scale testing.
- Support for both object-oriented and real-time development.
- Equivalent to MSC-2000 sequence charts.

*according to latest submissions to OMG
Why is UML 2.0 interesting?

- UML 2.0 is **more** than just a language.
- UML 2.0 is part of a **megatrend** in systems and software development that has been going on for 40 years:

  Closing the abstraction gap

  **What does this mean?**
Describing the problem

”I want to know exactly what everybody in my project is doing.”

“I want to automate our payroll system.”

”I want to build a nationwide cell phone system.”

“I want to build a car that every American can afford.”

”I want to put a man on the moon.”

These are some examples of problems and how they are described.
Solving the problem

• Once the problem is **described**, you have to **bridge** a certain **gap** to solve the problem.
  – **Requirements** have to be interpreted.
  – Systems need to be **decomposed**.
  – **Details** are assembled.
  – In reality, the problem is solved at a significantly **lower level**.

But the megatrend in all development is a constant movement upwards, to solve the problem at a higher and higher level...

...getting closer and closer to the level where it was described.
Software Development Megatrend – Closing the Abstraction Gap


Problem Description

Abstraction gap to bridge

Solution
The Abstraction gap to bridge

“I want to automate our payroll system.”

00110100110
00111000011
...

© Telelogic AB
How to bridge the gap?

“"I want to automate our payroll system.""

- The developer must establish the association between the machine model (solution space) and the model of the problem that is actually being solved (problem space).

- Unfortunately, this mapping is complicated because it is extrinsic to programming languages!
The origin of the gap

“I want to automate our payroll system.”

• Of course, all (programming) languages provide abstractions!

*BUT*

• the **complexity** of a problem one is able to solve is directly related to:
  • The kind and quality of abstraction
  • The ability to automate the solution creation
Advanced software engineering – real-time space

Problem

Solution:

Assembler

1960s

1970s

1980s

1990s

2000s

2010s

Solution:

C

Solution:

SDL

Solution:

UML 2.0

Solution:

Visual requirements expression?

Gap to bridge

Automation

Automation

Automation

Automation

Automation
Wrap Up – UML 2.0

• UML 2.0 is the next major revision of the Unified Modeling Language, a visual language for specifying, constructing and documenting software systems.
• UML 1.x was first standardized in 1997 by the Object Management Group (OMG), and it is currently at version 1.4.
• UML 2.0 is being defined right now, and is planned for release in 2002.
• UML 2.0 will be a merge of UML 1.x and SDL/MSC.
Agenda

- Who is Telelogic?
- SDL in a Nutshell
- SDL for ages 11 and up
- Wrap Up
- What’s next?
  - SDL 2000, UML 2.0
- Telelogic and the Standards
Agenda

• Who is Telelogic?
• SDL in a Nutshell
• SDL for ages 11 and up
• Wrap Up
• What’s next?
  – SDL 2000, UML 2.0
• Telelogic and the Standards
Telelogic today

• Telelogic is involved where the future is defined!
  – Telelogic’s Chief Technologist Cris Kobryn co-chairs the UML 2.0 work in OMG and co-authors the UML 2.0 proposal
  – Telelogic is strongly involved in the elaboration of new significant standards in ITU, ETSI and 3GPP
  – Telelogic is the leading tool provider for advanced wireless software development and 3G projects

• Telelogic has the leading position in the visual real-time development market
  – 46% market share of Software Modeling Tools (according to VDC 2000)

Thank you for your attention!