Introduction of SDL

Chi-I Hung

CCL/ITRI
• Collected by the United States Department of Defence from nine large software projects
Network Experiment

Software Costs

![Graph showing the increase in software costs from 1955 to 1985. The graph indicates a significant rise in software costs compared to hardware and maintenance costs. The y-axis represents the cost percentage, ranging from 0 to 100%.

Chi-I Hung
Specification and Description Language

★ Specification
-- A specification of a system is the description of its required behaviour

★ Description
-- A description of a system is the description of its actual behaviour

★ The Purpose of SDL
Provide a language for unambiguous specification and description of the behaviour of telecommunications systems.
Objective of SDL

- Formal description technique
- Easy to understand both for creators (direct users) and viewers (“non constructors” of specifications) (graphical representation)
- Object oriented language
- Independent of design paradigm (e.g. function or object oriented)
- Independent of implementation (language, operating system, and hardware)
Usage of SDL

a) facility requirements;
b) system specifications;
c) ITU-T Recommendations, or other similar Standards (international, regional or national);
d) system design specifications;
e) detailed specifications;
f) system design descriptions (both high level and detailed enough to directly produce implementations);
g) system testing descriptions (in particular in combination with MSC and TTCN).

(From ITU-T Z.100)
Improvement of Error Rate

Source: Software Process Improvement with SDL, K. Kimbler, G. Opsahl, SDL Forum 1995
Process Dimension - CMMI

Process Management
- Organizational Process Focus
- Organizational Process Definition
- Organizational Training
- Organizational Process Performance
- Organizational Innovation and Deployment

Support
- Configuration Management
- Process and Product Quality Assurance
- Measurement and Analysis
- Decision Analysis and Resolution
- Causal Analysis and Resolution
- Organizational Environment for Integration

Project Management
- Project Planning
- Project Monitoring and Control
- Supplier Agreement Management
- Integrated Project Management
- Integrated Teaming
- Risk Management
- Quantitative Project Management

Engineering
- Requirements Management
- Requirements Development
- Technical Solution
- Product Integration
- Verification
- Validation

Chi-I Hung
**SDL History**

- **International Standard (ITU-T, Z.100)**

  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 Changes regarding architecture and data model.

*Chi-I Hung*
Representations of SDL

- **Graphical**

![SDL-GR Diagram]

- **Textual**

```plaintext
SYSTEM A;
SIGNAL
  sig1,
  sig2(real);
CHANNEL c1
  FROM B TO ENV WITH sig2;
  FROM ENV TO B WITH sig1, sig2;
ENDCHANNEL c1;
BLOCK B REFERENCED;
ENDSYSTEM A;
```

Chi-I Hung
Application of SDL

- call and connection processing
  (for example, call handling, telephony signalling, metering)
  in switching systems
- maintenance and fault treatment
  (for example alarms, automatic fault clearance, routine tests)
  in general telecommunications systems
- system control
  (for example, overload control, modification
  and extension procedures)
- operation and maintenance functions, network management
- data communication protocols
- telecommunications services
SDL Design Methodology

- Flow of Software Development
- Basic Concept of SDL Design Flow
- Basic Component of SDL
Flow of Software Development

- Software Requirement Analysis
  - UML
- Design
  - SDL Editor
  - MSC Editor
- Coding
  - SDL Code Generator
- Testing
  - SDL Simulator
  - TTCN Simulator

Chi-I Hung
Network Experiment

Protocol Development Process

Protocol Specification

Software Requirement

Test Suite
- TTCN-2
- TTCN-3

Host Conformance Test

Formal Description Language
- SDL
- UML-2

Conformance Test

Implementation Under Test (IUT)

Product

Code-Generation

Execution Test Suite

Chi-I Hung
Basic Concept of SDL Design Flow

- System Architecture --SDL Editor
- Use Cases --MSC Editor
- SDL Coding
- Code Analyze
- Code Gen
- Simulation
- SDL Code-Gen
  - C/C++ Source Code
  - Other Source Code or Object Code
  - Compile & Link
  - Execution File

Chi-I Hung
Basic Component of SDL

- **Architecture**
  - system, block
- **Behavior**
  - processes
- **Communication**
  - signals and channels
- **Data**
  - abstract data types
**SDL Architecture -- System**

- **SDL System**
  A SDL-system is the outermost agent that communicates with the environment.

- **Environment**:
  The environment of the system is everything in the surroundings that communicates with the system in an SDL-like way.
A SDL System must define the interfaces to communicate with other components

A complete SDL system must contain at least one block

A block must contain at least one process or block, blocks and processes must not be mixed in one block.

A process must contain an finite state machine
Network Experiment

SDL-System Structural Concepts (2)

Hierarchical Structure

Chi-I Hung
A SDL System can contains:

- **System Name**
- **SDL Blocks**
- **Channels**: The relation between the system and the ‘outside world’.
- **Signal Definition**
- **Data Type Definition**
A block can be refined either by a set of processes or by a set of block substructures.

Blocks are static entities in SDL.
SDL-System Example (1)
Network Experiment

SDL-System Example (2)

Chi-I Hung
Network Experiment

SDL-System Example (3)

![SDL Editor - System UE_Protocol](image1)

![SDL Editor - Block UE_AS](image2)

Chi-I Hung
Network Experiment

SDL-System Example (4)

process RLC

State_0

Input_Signal

RLC_Data_Req

State_0

State_1

Chi-I Hung
Processes are the actors of SDL systems.

An SDL process is the active unit of an SDL system.

An SDL process represents an extended finite state machine.

Processes may be defined with an initial and a maximum number of instances.

SDL processes can communicate by means of asynchronous message passing with other processes.
Elements of SDL Process

• The SDL Process may contain:

- Process name
- Formal Parameters
- Local Variables
- Time and timers
- The graphical representation of a FSM
**Behavior of SDL**

✧ *SDL Process as Finite State Machine*

- In SDL, the FSM is extended with data processing (EFSM)
- “Send no wait”, asynchronous signalling
- Process instances can be created and terminated dynamically
- A process is activated by means of an arriving signal. At the end of a transition the next state is entered.
Process Symbols

- start
- state
- input
- priority input
- output
- save
- condition

- task
- procedure call
- process reference
- decision
- create request
- stop
- text

Chi-I Hung
The RLC will send \textit{RLC\_Data\_Req} to MAC via \textit{RLC\_MAC\_Queue}.

Chi-I Hung
Each process instance has a FIFO queue of its own

The port allows received signals (and timeouts) to be queued until they are consumed (or discarded) by the owning process instance.

The queues have an infinite capacity.

The FIFO order in processing the queue is NOT always preserved (because of priority/save).
Each SDL process maintains a set of intrinsic variables which are of the predefined data type \texttt{PId} (Process Identity)

- \texttt{self}: Denotes the PId of the process instance itself
- \texttt{sender}: Denotes the PId of the process instance from which the most recent signal has been consumed
- \texttt{parent}: Denotes the PId of the process instance that has created the instance
- \texttt{offspring}: Denotes the PId of the most recent process instance created by this process instance.
Dynamic Process Creation

- Created processes must lie in the same block.
- Termination of a process can only be made by itself.
 SDL Procedure

- Procedures can have IN or IN/OUT parameters.
- Procedures can return a value.
- SDL procedures can contain states and variable declarations.
- Procedure states are disjoint from process state space.
- Procedures can see the variables of the owning process.
- Procedures can only use timers declared in the owning process.
Chi-I Hung

SDS Procedure

MyProcedure

Ret_val:=call MyProcedure(1,my_arg)

procedure MyProcedure

:FPAR
IN arg1 integer,
IN/OUT arg2 integer;
RETURNS ret_val integer;

DCL Total Integer;

Total:=arg1+arg2;

Total
SDL Channel

- Communication path between blocks itself and between blocks and the environment
- Channels can be unidirectional or bidirectional communication devices.
A signal route defines a communication path between a process and its environment or another process.
Signals

- Signals are the primary communication mechanism in SDL
- Signals may carry data by means of parameters.
- All signals to and from the environment are declared at system level.
- Signals can be defined inside a block (thus only visible in the block)
If there are several possible receivers of a signal, the signal will be delivered to an arbitrary receiver.

Example:
SDL provide two way for addressing:

- **VIA**: Specifies a signal path
- **TO**: Specifies a receiving process

**Example 1:** A Send \( \text{Sig1} \) to B

- Sig1 via C1  **OR**  Sig1 to B

**Example 2:** A Send \( \text{Sig1} \) to C

- Sig1 to C
SDL Data

Based on abstract data types
- An abstract data type defines a type of data object by its functional properties, i.e. by a set of operations applied to it.

In SDL, data types are called “sorts”

Variables can only be declared in processes
- No global variables.

In SDL inheritance and generics are supported

A new data type is defined by:
- constants and a set of values
- operators
- axioms
SDL Predefined Data Sorts

- Integer
- Natural
- Real
- Boolean
- Character
- Charstring
- PId
- Duration
- Time
Boolean is a typical example of a data type in SDL

NEWTYPE Boolean
LITERALS True, False;
OPERATORS
“NOT”:Boolean ->Boolean;
“=” :Boolean, Boolean ->Boolean;
“/=”:Boolean, Boolean ->Boolean;
“AND”:Boolean, Boolean ->Boolean;
“OR”:Boolean, Boolean ->Boolean;
“XOR”:Boolean, Boolean ->Boolean;
“=>”:Boolean, Boolean ->Boolean;
AXIOMS
“NOT”(True)== False;
“NOT”(False)== True;
...
ENDNEWTYPE Boolean;
Variable Declaration and Use

process RLC

DCL Count Integer

State_0
Input_Signal
Count:=1
Output_Signal
State_1

State_1
Input_Signal
Count

20
30

40
The “Task” symbol is used for assignments

A decision is much like an if or case statement.

\[i := i + 1;\]
\[k := i \text{ mod } 7;\]
With Synonym, constants are declared in SDL

- Can be used in SDL-System or Blocks
- If a constant is declared as EXTERNAL it means that the constant will be assigned first at system start-up time (i.e. it can be assigned different values each time).

```plaintext
SYNONYM Zero Integer = 0;
SYNONYM One Integer = 1;
SYNONYM NrOfDoors Natural = EXTERNAL;
```
\textbf{Newtype}

\textbf{Newtype creates a new data type}

\begin{itemize}
  \item \textbf{Newtype} \texttt{SeqNum}
  \begin{itemize}
    \item \textit{literals} \texttt{zero, one};
    \item \textit{operators} \texttt{succ : SeqNum -> SeqNum;}
    \item \textit{axioms}
    \begin{itemize}
      \item \texttt{succ(zero) == one;}
      \item \texttt{succ(one) == zero;}
    \end{itemize}
  \end{itemize}
  \texttt{Endnewtype} \texttt{SeqNum;}
\end{itemize}

\begin{itemize}
  \item \textbf{Newtype} \texttt{EyeColorType}
  \begin{itemize}
    \item \texttt{LITERALS Blue, Brown}
    \item \texttt{DEFAULT}
    \begin{itemize}
      \item \texttt{Blue}
    \end{itemize}
    \item \texttt{Endnewtype} \texttt{EyeColorType;}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{DCL} \texttt{MySeqNum SeqNum;}
  \item \texttt{DCL} \texttt{MyEye EyeColorType;}
\end{itemize}

\begin{itemize}
  \item \texttt{SeqNum := zero;}
  \item \texttt{SeqNum := succ(SeqNum);}
  \item \texttt{MyEye := Brown;}
\end{itemize}
Syntype

◊ Restrict the set of values of a ground type

- NOTE! The syntype will not be a new type but a subtype, and can thus be assigned values of the ground type.

SYNTYPE Age = Natural
CONSTANTS 0:100
ENDSYNTYPE Age;

DCL
MyAge, YourAge Age;

MyAge:=1, YourAge:=100
Struct is a data type where the data structure is explicitly stated.

- Struct in SDL is similar to struct in C-Language.

```c
typedef enum
{
    type_802_11_a,
    type_802_11_b,
    type_802_11_g,
} Connect_Type;

typedef struct
{
    Connect_Type con_type;
    char id;
    int timing;
} Con_Req_Arg;
```

```c
newtype Connect_Type
literals type_802_11_a,
    type_802_11_b,
    type_802_11_g;
endnewtype Connect_Type;

newtype Con_Req_Arg struct
con_type Connect_Type;
    id Character;
    timing Integer;
endnewtype Con_Req_Arg;
```

---

Chi-I Hung
On timeout, a signal is sent to the process instance that set the timer.

- **SET:**
  - Start a timer with time value

- **RESET:**
  - The associated time value is lost
  - Corresponding signal in the input port is removed
Generator

The Generator is like the “template” in C++

NEWTYPE int_set
LITERALS empty_int_set;
OPERATORS
  add: int_set, Integer -> int_set;
  is_in: int_set, Integer -> Boolean;
AXIOMS
FOR ALL m, n IN Integer, s IN int_set
    ..........
ENDNEWTYPE:

NEWTYPE real_set
LITERALS empty_real_set;
OPERATORS
  add: real_set, Integer -> real_set;
  is_in: real_set, Integer -> Boolean;
AXIOMS
FOR ALL m, n IN Integer, s IN real_set
    ..........
ENDNEWTYPE:

GENERATOR set
  (TYPE element, LITERALS empty_set)
LITERALS empty_set;
OPERATORS
  add: set, element -> set;
  is_in: set, element -> Boolean;
AXIOMS
FOR ALL m, n IN element, s IN set
    ..............
ENDDGENERATOR;

NEWTYPE int_set set (Integer, empty_int_set)
ENDNEWTYPE;
NEWTYPE real_set set (real, empty_real_set)
ENDNEWTYPE;
Array

Array is the predefined generator.

-- GENERATOR Array(TYPE Index, TYPE Itemsort);

SYNTYPE IndexType = Natural
CONSTANTS 1:100
ENDSYNTYPE;

NEWTYPE Person STRUCT
  theName Charstring;
  theAge Age;
  theEyeColor EyeColorType;
ENDNEWTYPE Person;

NEWTYPE PArrayType
  Array (IndexType, Person)
ENDNEWTYPE PArrayType;

DCL PersonArray PArrayType;

PersonArray(2):= (. ‘John’, 26, Blue .)
System Integration (i)

• Light Integration

SDL system (process)

Process Instance

Communication Adaptor

Non-SDL Process

operating system
-- I/O, Timer, Process Management, IPC ...

Chi-I Hung
System Integration (ii)

• **Tight Integration**

SDL process 1
- Process Instance
- Communication Adaptor

SDL process 2
- Process Instance
- Communication Adaptor

SDL process 3
- Process Instance
- Communication Adaptor

Non-SDL Process

Operating system
-- I/O, Timer, Process Management, IPC...