Chapter

2

Introduction to the SDL Suite

This chapter contains a brief introduction to the SDL suite and to the functionality of its components.

After reading this chapter, you may want to familiarize yourself with the SDL suite: starting the tools, proceeding with a small example, or reading about what is new in this release in comparison to earlier versions. You are then recommended to study the following chapters:

- **For a beginner's tutorial on the SDL suite: chapter 3, *Tutorial: The Editors and the Analyzer*, in this volume.**
- **For news compared to previous versions: chapter 2, *Release Notes, in the Release Guide*.**

About the SDL Suite

Telelogic

The SDL suite is developed and marketed by Telelogic. Our company has been a firm supporter of the SDL, MSC and UML languages for a long time. We cooperate with ITU and OMG in the ongoing work of improving the languages and with ETSI in defining international standards in the field of communication protocols.

We initiate and participate in international research programs on how to use the languages in different application areas (such as the European Community programs RACE, ESPRIT and EUREKA, as well as the Swedish national IT program).

Our experience and know-how in these areas is put to practice when we develop software engineering tools that support the languages. Our family of tools is known as Telelogic Tau.

The SDL Suite

Tools for design and specification languages must be able to create, maintain, and verify a specification with respect to the language syntax and semantics. It is also fundamental that the tools can simulate, validate and generate application code to other high level languages.

To be able to perform a complete development cycle, the tool should support early analysis phases, and the move from object-oriented analysis to SDL design.

The tool should be able to export and import information from other SDL tools. Major documentation standards or de-facto standards should be supported.

The tool should provide an intuitive and consistent graphical user interface which reduces learning time and makes it easy to work with the tool. Besides the graphical user interface, a batch facility should allow to process a large amount of information without user interaction.

A powerful, context-sensitive Help facility should be provided, freeing you from time-consuming browsing through user documentation in search for the topic of interest.

The SDL suite can do all of this, and much more.

Overview of the SDL Suite

Architecture

The SDL suite is a member of the Telelogic Tau family, which also includes the TTCN suite and the UML suite.

Telelogic Tau consist of a number of separate tools that process information. The tools are integrated using a *selective broadcast integration mechanism*, making it possible to design a highly integrated system from separate tools. This approach also makes it possible to add new tools without creating any conflicts with the existing tools. In addition, the integration between two separate tools can be easily enhanced, and tools can communicate with each other over a network.

The interface that ties the tools in Telelogic Tau is called *The PostMaster*. The parts of the Postmaster interface that are of interest for the users, are documented so that you can read about how to integrate your own tools with Telelogic Tau.

Starting the SDL Suite tools

The SDL suite components are normally started by using the *sdt* start script in the bin directory of the Telelogic Tau installation. This script can take a number of options:

- `sdt -reuse`
  If there is a running Telelogic Tau session, then the Organizer from that session is displayed. If there is no running Telelogic Tau session, then a new session is started.
- `sdt <system file>`
  Starts the SDL suite components and loads the specified system file.
- `sdt <diagram file>`
  Starts the SDL suite components and loads the specified diagram file in an editor.
- sdt <archive file> [<unpack directory>]
  Starts the SDL suite components, unpacks the specified archive file, and loads any system file found in the archive file. The archive file is unpacked in the specified unpack directory. If no directory is

specified, then the Organizer will display a dialog, asking for a directory to unpack in.

- **On UNIX**, `sdt -fg`
  Normally, the *sdt* start script immediately gives the user a new command line prompt for other commands. When *sdt -fg* or is used, the start script will wait until the SDL suite is closed down before giving the user a new command line prompt.
- `sdt -noclients`
  This variant starts the Postmaster without starting any clients/applications such as the Organizer. Later, Postmaster clients can be started and attached to the postmaster. To attach to an existing postmaster, start a client with *-post*. Read more about this in "Run-Time Considerations" on page 515 in chapter 11, *The PostMaster*.
- `sdt -grdiff [ -notmoved ] v1.ssy v2.ssy [ [ -original o.ssy ] -mergeto r.ssy ]`
  Starts the SDL suite components and immediately invokes the *Compare Diagrams* operation (see "Compare Diagrams" on page 1961 in chapter 44, *Using the SDL Editor*, comparing the SDL diagram files v1.ssy and v2.ssy. The use of the -notmoved option corresponds to setting the option Ignore moved or resized objects to off. (Note that if you use the -n option on UNIX it will be processed by the X window system) Normally this means that moved and resized symbols will not be detected as being different. When the operation is finished, the SDL suite is closed down. If the *-mergeto* option is used, the Merge Diagrams operation is used instead. (See "Merge Diagrams" on page 1961 in chapter 44, *Using the SDL Editor* and the merge result diagram is saved in r.ssy. If the -original option is used as well, then the merge operation tries to auto-merge differences using the o.ssy file as a guide for deciding how to merge v1.ssy and v2.ssy. o.ssy should point out an original diagram file version that both v1.ssy and v2.ssy are derived from. When using -grdiff to start a merge operation from another tool (such as a configuration management tool), combine -grdiff with -fg to have the other tool wait with processing the merge result file until after the merge operation is finished.
- `sdt -sdtdiff v1.sdt v2.sdt [ -mergeto r.sdt ]`
  Starts the SDL suite components and immediately invokes the *Compare System* operation (see "Compare System" on page 73 in chapter 2, *The Organizer*, comparing the system files v1.sdt and v2.sdt.

When the operation is finished, the SDL suite is closed down. If the -*mergeto* option is used, the user will be prompted to save the merge result when the operation is finished, and r.sdt is used as a proposed file name.

Batch Facilities

The *Batch Facilities* are commands that you type from the OS prompt. These facilities take advantage of the Postmaster and pass messages to the tools, ordering the individual tools to process information as required. The batch facilities support the following operations:

- Printing (to file or to printer)
- Analyzing (typically syntactic and semantic check of an SDL system)
- Making (for instance building an application for target environment)
- Comparing SDL diagrams (with a textual report)

Licensing Mechanism

The software license server controls the licensing of the tools included in Telelogic Tau. This is performed through a floating license mechanism based on a third party software, FLEXlm™[1]. The current license numbers along with a key are stored on a text file, which is distributed at installation of the software. This provides a flexible way of upgrading licences and adding new license agreements, as well as allowing you to keep track of the actual usage of the tools you have purchased.

FLEXlm supports multiple tools (even from different tool manufacturers) sharing the same license server, so Telelogic should not cause any problems when installing it into your computer environment.

For increased flexibility in the use of licenses and to prevent started but unused tools from holding licenses, an optional timeout feature can be enabled. This will automatically release licenses when a user has been idle during a selectable interval.

1. FLEXlm stands for Flexible License Manager.

For more information on license mechanisms, see chapter 6, *A Primer on Licensing, in the Installation Guide*.

Common Telelogic Tau Tools

The SDL suite shares a set of tools common to Telelogic Tau:

- *The Organizer* features a graphical view of all diagrams and documents making up a system. This may include SDL hierarchies, Message Sequence Charts, Object Model diagrams, State Charts, High-level MSCs, TTCN documents and text documents. The view may be freely organized into chapters and modules according to your preference.

  Furthermore, the Organizer manages the other tools in Telelogic Tau, taking advantage of their respective functionality when needed and thus providing the feeling of a truly integrated tool set.

- *The Link Manager* and *The Entity Dictionary* maintains and visualizes *Implinks and Endpoints*. Implinks (short for implementation links) are used to trace the implementation and design decisions for concepts and objects between different phases in the development process. The link endpoints are texts and objects created in the editors described below. The Link Manager and the Entity Dictionary can be started from all of the editors.

- *The Preference Manager* allows you to set up or change the behavior of the Telelogic Tau tools by customizing the values of preference parameters. It is possible to specify whether a customized behavior should be project-wide or even company-wide, or if an individual user should be allowed to customize some behavior.

- When you are *Printing Documents and Diagrams*, there are various options that allow you to customize the printouts so that they fit in your documentation environment. Except for when you print TTCN documents, it is possible to generate, PostScript, encapsulated PostScript, FrameMaker™, Interleaf™ and web files. **In Windows**, you can also print to any printer you have set up in Microsoft Windows.

- *The on-line help* provides access to help on tools, windows, dialogs and commands. The on-line help is in HTML-format, featuring hypertext links and navigation support. The PostScript files that were used when printing this manual are also enclosed in the distribution. You are free to produce additional hard-printed copies of the manual pages that are of interest.

- *The PostMaster* implements the integration mechanism that ties the tools together. The public parts of the PostMaster interface are documented, allowing you to integrate your own tools with Telelogic Tau tools and the information they manage.

The SDL Suite Graphical Tools

The SDL suite comprises the following graphical tools:

- *The diagram editors* are used for creating, editing and printing Object Model, State Charts, Message Sequence Charts and High-level Message Sequence Charts diagrams.

  The OM Editor uses the full graphical notations of OMT/UML. It keeps track of all class and object definitions with the same name in a scope of OM diagrams, and supports the merging of these definitions for maintaining a combined view of a class. The SC and HMSC editors work in a similar way.

  The MSC Editor uses the graphical notation defined in the standard Z.120. Also, it can serve as a powerful graphical trace tool when you simulate and validate a system specified in SDL. MSCs can also be verified for consistency with an SDL system when you use the SDL Validator.

- *The SDL Editor* is used for creating, editing and printing specifications and descriptions using the graphical SDL notation defined in the standard Z.100. The SDL Editor also performs various syntax checks at editing time.

  – Advanced functions include a context-sensitive grammar help and signal dictionary. When you edit an SDL diagram, the signal dictionary is automatically updated to contain all SDL signals that you add to a system and provides immediate access to them.

  – The SDL Editor can also display *Overview Diagrams* of the SDL system, where the diagrams are displayed in a nested fashion.

- *The SDL Type Viewer* visualizes the impact of the inheritance and specialization mechanisms in your SDL-92 system. The Type Viewer produces a graphical tree that is of great assistance to understand and take full advantage of the SDL types[1] that you have defined in an SDL system.

- *The SDL Index Viewer* presents listings of definitions and cross-references in a clear and easy-to-understand graphical notation. The Index Viewer is provided with filtering and navigation functions, with a trace-back to the source SDL or MSC diagrams.
- *The SDL Coverage Viewer* is a test coverage and profiling tool that displays the results of a simulation or validation as a graphical transition or symbol tree. The tool can present an overview of the system, coverage or a detailed view on a part of the system.

Other SDL Suite Tools and Back-End Facilities

The following additional tools and facilities are available:

- *The Text Editor* is used for creating, editing and printing ASCII text documents. The text documents can be textual requirements, use cases and other textual documentation used in the development process. The Text Editor can also be used for writing ASN.1 or C code to be linked to the SDL system.
- *The ADT Library* (library of Abstract Data Types) features a number of general ADTs that provide the basic services that are often needed when you design an SDL system. The ADT library is distributed in source code so you can tailor the ADTs to fit your specific requirements, if needed.
- *The SDL Analyzer* performs several functions. It performs syntactic and semantic analysis of your SDL descriptions, generates error reports and warnings in appropriate cases, and has the ability to produce information about definitions and cross references in an SDL system. The Analyzer also converts SDL information from the Graphical Representation (SDL/GR) to the textual Phrase Representation (SDL/PR). The reverse conversion is also possible, allowing you for instance to import PR files from other tools supporting SDL.
- *The Cadvanced/Cbasic SDL to C Compiler* transforms your SDL system into a number of C source files that are compiled and linked with an SDL suite run-time library. The C code can be used for a

1. The SDL term *type* corresponds to the term *class*, used in many OO notations

number of purposes, depending on what libraries are available in your configuration (see below). The SDL to C Compiler is available in a *Cbasic* (for simulation and validation purposes), and a *Cadvanced* (for building any kind of application) version.

- *The SDL Simulator library* allows you to make an executable program, a *simulator*, which helps you to understand and debug the behavior of a system specification. The simulator can be controlled from a graphical user interface (SimUI).

  You can choose to focus on the external view of a system specification, where you are interested in the signal interface, or on the internal behavior of a system specification. The execution of a simulator can be traced in a graphical mode in the source SDL diagrams and can be logged graphically in terms of Message Sequence Charts. Target simulation is also supported.

- *The SDL Validator library* allows you to make a *validator*, an advanced "self-exploring" simulator that may be used for finding errors and inconsistencies in an SDL system and for verifying that a system is consistent with a Message Sequence Chart. The validator can be controlled from a graphical user interface (ValUI).

- *The Performance Library* allows you to create a performance model of your SDL system that you run on your host computer. The library is optimized with respect to performance, so that a large amount of statistical data can be produced during a reasonable execution time.

- The Cadvanced SDL to C Compiler can be used for *Building an Application* for both host and target environments. Predefined *Application libraries* are available for specific host environments. *The Master Library* is the SDL suite run-time library in source code format, which can be customized to fit different needs and operating systems.

  Integration with Operating Systems supports most of the commercially available real-time operating systems. You can also build applications where the runtime library schedules the system and sets the real-time pace.

- *The Cmicro SDL to C Compiler* is designed to meet the needs of small to mid-range microcomputer controlled applications. It translates an SDL system into optimized and compact C code with highly reduced memory requirements. The Cmicro SDL to C Compiler is part of the *Cmicro Package* which in addition consists of the Cmicro Library and the SDL Target Tester. The Cmicro Package can be ordered separately.
  - *The Cmicro Library* is the “virtual SDL machine” needed to build an executable from the generated Cmicro Code.
  - *The SDL Target Tester* allows testing and debugging of the generated SDL system while it is running on a target. A prerequisite is a communications link to a host system. The SDL Target Tester is an optional part of the Cmicro Package.
- *TTCN Test Suite Generation* is provided by two features: TTCN Link and Autolink. They provide a means to check the consistency between an SDL system managed by the SDL suite and a test specification, expressed in TTCN[1] managed by the TTCN suite. TTCN Link generates the declarations of the test specification automatically. In the TTCN suite, there is direct access to the SDL system specification and you can interactively build test cases. Autolink is a feature of the Validator, and can generate entire test suites from an SDL specification.

1. TTCN stands for Tree and Tabular Combined Notation. It is an ISO standard that is used to describe a test specification.

Information Management

In order to properly use Telelogic Tau, you need to understand the basics for how the information is organized.

SDL Diagrams

The SDL suite primarily handles SDL information in the graphical representation, SDL/GR. The major advantage that follows this approach is that you are free to apply any graphical style guide to your diagrams since the SDL suite lets you position symbols and shape lines the way you like.

Each SDL diagram consists of a number of diagram pages. An SDL diagram page may contain references to other SDL diagrams. This allows you to build a hierarchical structure which adheres to the SDL syntax rules. See Figure 16.

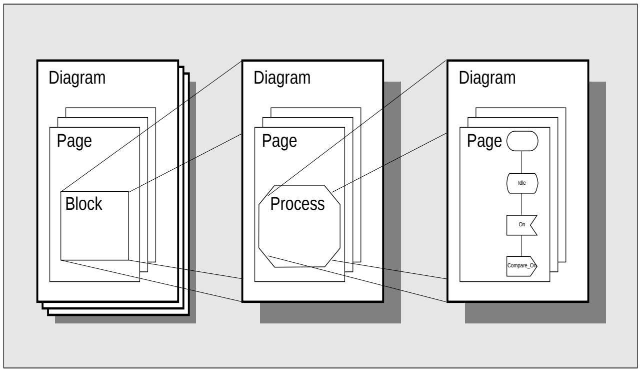

Figure 16: Organization of SDL information

Each diagram stored on its own individual file. An SDL structure is built up from a number of these SDL files. These files are logically tied together by the SDL suite components, in order to constitute a coherent SDL structure. This process is managed by the Organizer. The SDL suite can manage several separate SDL structures at the same time.

You may also include SDL/PR files into an SDL/GR structure. Transformation of SDL/GR to SDL/PR and vice versa is supported.

MSC Diagrams

Message Sequence Charts are mainly handled in the graphical representation, MSC/GR.

In contrast to SDL diagrams, MSCs are not paginated and cannot build a hierarchical structure. However, an MSC diagram can reference other MSCs (or HMSCs, see below), but without implying any structure between them.

MSC diagrams may be managed as entities of their own. The Organizer also supports including MSCs in an SDL structure using the concept of *associated documents*.

Telelogic Tau allows reading and writing of MSCs expressed in the textual form, MSC/PR. Both the *instance-oriented* and *event-oriented* forms are supported, according to the recommendation.

High-Level MSC Diagrams

In contrast to “plain” MSCs, High-level MSCs (HMSCs) are paginated, but they cannot build a hierarchical structure visible in the Organizer. However, an HMSC diagram can reference other HMSCs or MSCs, but without implying any structure between them.

Textual SDL and MSC Formats

Telelogic Tau has the ability to read and write SDL and MSC textual files, SDL/PR and MSC/PR. The primary purpose is to enable importing and exporting of SDL and MSC information, rather than to provide an alternative storage format, since the layout and exact appearance of a diagram is lost when stored in PR format and read back again.

Telelogic Tau also use the PR formats as temporary storage formats when processing information.

SDL/GR diagrams can be converted to and from CIF (Common Interchange Format) files, by using CIF converters supplied with the SDL suite. CIF is an extension to SDL/PR that also stores the graphical layout information. However, CIF files cannot be managed directly by the SDL suite.

Object Model Diagrams

Object Model (OM) diagrams may be managed as entities of their own, or grouped together using the concept of *modules* in the Organizer.

OM diagrams are paginated, but does not contain any structure information. However, a *scope* concept is used to allow the same object class to be defined in more than one OM page or diagram. All diagrams and pages within the scope are considered when the complete definition of a class is needed. OM diagram belong to the same scope if they are managed in the same *module* in the Organizer.

State Chart Diagrams

State Chart (SC) diagrams are paginated, but does not contain any structure information, nor any references to other diagrams. They are always managed as entities of their own.

Text Documents

Telelogic Tau handles text documents in the form of plain ASCII files. Telelogic Tau uses the file extension of text files to determine the type of text file. Telelogic Tau recognizes text files as C header, ASN.1 or as plain text files.

Text documents are managed as entities of their own, but C header and ASN.1 specifications can be linked to the rest of the system by using the concept of *dependency links*. In this way, these text documents can be analyzed and translated to SDL/PR format.

A text document can also be a *build script*, containing commands to control the analyze and code generation process in detail.

The System File

Once Telelogic Tau is up and running, you may work on individual documents, regarding them as individual objects of their own. However, this requires that you keep track of each individual file.

When the amount of documents increases, this process tends to become rather complicated, in particular when introducing inheritance and specialization between SDL diagrams, and dependency links between different document types.

To cope with this problem and as a means to ensure the consistency of a document structure, the *system file* is introduced. The system file is managed by the Organizer.

Document Structure

The system file holds the information about the SDL structure and all other documents included in a system. It also keeps track of the file bindings, i.e. what file a particular document is stored on, and the dependency links between the documents. When working on your documents, the Organizer keeps track of the changes you apply and updates the system file accordingly.

A graphical approach is used, in order to display the contents of the system file in the Organizer. Documents may be freely organized into *chapters* and *modules* (within chapters) to keep related documents together.

Connections between documents in different chapters, modules and SDL structures can be made in the form of *associations* and *dependency links*.

Options

In addition to the properties mentioned above, the system file may store information about what *options* you have set up for the document structure that is managed by the system file. Typically, *analysis* and *code generation* options are stored in the system file.

The Link File

The Link Manager keeps track of all link endpoints and implementation links in the system. This link database is stored on a separate link file, which is referenced from the system file.

Control Unit Files

Control unit files facilitate multiuser support when you work with an SDL system. They contain structure information for a subset of a document system and are suitable for configuration management (revision control). If control unit files exist, they are referenced from the system file.

Source Management

Since Telelogic Tau operates on files, you may use any revision handling system for checking out work copies of your SDL diagram files to your work directory (this task needs to be performed outside Telelogic Tau).

Telelogic Tau may also be configured to manage multiple versions of your source documents, by binding a document to any suitable file in your file system.

Telelogic Tau provides mechanisms for an easy rebinding of documents. These file binding mechanisms allow you to keep track of multiple versions of your source documents with a minimum of effort.

Target Management

Virtually all of the output information that is produced with Telelogic Tau consists of files, most of them use a text-based format (for instance SDL/PR files and C files).

You may specify default locations for files that are generated with Telelogic Tau. Also, you may specify the level of granularity, allowing you to generate multiple files or one file only.

Furthermore, the SDL suite features an *SDL-Make* mechanism that minimizes the turnaround time, by computing the passes the tool must run in response to a modification of a source diagram.

PCs and Workstations

User Interface

On UNIX workstations, Telelogic Tau is implemented as X Window applications, using the Motif widget set. On PCs, Telelogic Tau is designed as Microsoft Windows applications (Windows 95, Windows 98, Windows NT 4.0 or Windows 2000). At present, some features are not available on the PC platform.

Since the SDL suite is supported on different systems, there may be slight differences in the appearance of the tools between environments. However, the functionality is identical as long as the underlying system and the OS allow it.

All Telelogic Tau graphical applications follow the same style guide, described in chapter 1, *User Interface and Basic Operations, in the User's Manual*.

Supported UNIX Systems

Full compatibility between the SDL suite on PCs and the SDL suite on UNIX workstations ensures that future upgrading of your computers towards workstations is possible, and allows heterogeneous network solutions with, for instance, PCs connected to a UNIX based file server.

On workstation environments, the following architectures and operating systems are supported:

- Sun SPARCstation (Solaris)
- HP 9000/700 and 800 series (HP-UX)

For more information about the supported platforms, see chapter 1, *Platforms and Products, in the Installation Guide*.