Background on SDL-2010

Rick Reed

TSE, The Laurels, Victoria Road, Windermere, LA23 2DL,United Kingdom.
rickreed@tseng.co.uk

Abstract. *When I proposed a session on SDL-2010 in the SAM-2010 organising committee a few weeks ago, I had not expected to be asked to prepare an accompanying "paper ". What I have done is to update the "SDL-2010 route map ". I am currently updating documents for the SDL-2010 standard and plan to make at least a snapshot of these available for download at SAM 2010.*

This document gives some background on the development of a revised version of the ITU-T Specification and Description Language standard, which is scheduled to be consented for approval by ITU-T at the end of the year 2010. At the time of writing the current standardized (or in ITU-T terminology Recommended) version is called SDL-2000. In this document and ongoing work the revised version is called SDL-2010.

The background provided here is to supplement the presentation to be given in the "Tutorial on SDL-2010" session during SAM 2010. The intention of the session is to present SDL-2010 at its current status as the basis of a general discussion on future development of Specification and Description Language including (but not limited to) whether SDL-2010 should be approved as proposed.

1 Background, history and status of SDL-2000

SDL-2000 was completed in 1999 with revised versions of the ITU-T Recommendations Z.100, Z.105, Z.107 and Z.109. The main document was Z.100 with Z.105 and Z.107 covering use of ASN.1 with the SDL-2000, and Z.109 covering use with the Unified Modelling Language. A revised the Common Interchange Format for SDL-2000 in ITU-T Recommendation Z.106 supplemented these in the year 2000. Since then there have been some minor updates to these Recommendations. The text was re-organised in 2002 so that Z.100 describes the graphical language and the parts of the textual (SDL/PR phase representation) that are alternatives to graphical presentation were moved to the interchange format in Z.106. At the same time a number of corrections and a few minor changes were made. In 2003 a Corrigendum was issued to incorporate to new Annexes B and C that concern backwards compatibility and conformance to the standard. But these changes have been minor or re-organization only or to correct flaws in the 1999 version, so that essentially SDL-2000 has not changed and has remained stable.

When SDL-2000 was being developed, right up until the ITU-T meeting at which it was approved there were two sizeable software organizations that were promising to produce tools to support SDL-2000 in 2000 or 2001: Telelogic and Verilog. A merger of these two organizations in Telelogic had been announced before the end of 1999, so that some competition was removed in the tool market. Although these commercial tools already supported some of the features of SDL-2000 by 2000, it is now unlikely that there will ever be a tool that approaches full support of SDL-2000 in its final form of Z.100 (11/2007). Even the tool that best supported SDL-92, Cinderella, reached a position by 2007 when it would probably never offer full SDL-2000 support, because it has a smaller (at least in value terms) share of the ITU Specification and Description Language tool market and had to offer compatibility with Telelogic as the market leader at that time. Cinderella collaborated with Humboldt University that previously had not entered into the commercial tool market. The Humboldt SDL-tool implemented many of the features of SDL-2000 on a trial basis to test the feasibility of various ideas - indeed some features such as nested packages were implemented specifically to support feature requests promoted by Humboldt for OMG related work. In 2003 SOLINET announced the SAFIRE tool set, claiming that it is based on Z.100. In late 2004 PragmaDev, which previously supported a dialect called SDL-RT announced support also of Z.100. All four organisations (Cinderella, PragmaDev, SOLINET/SAFIRE, Telelogic) had commercial tools available in May 2006, though SOLINET/SAFIRE had ceased to be involved in ITU or SDL-Forum activities and the future of the language. By November 2008 all Telelogic products and services had become part of the IBM Rational Software portfolio, and the main tool vendors were IBM, PragmaDev and Cinderella (probably in order of market share at that time). At the time of writing all three of these vendors are still offer Specification and Description Language tools.

Since 1999 the general market perception has developed. In 1999, part of the rationale for developing Z.109 as a UML profile for the ITU Specification and Description Language was because UML was perceived as a major competitor to the ITU language. Within the telecommunications industry some organisations were divided internally between those that favoured the ITU Specification and Description Language and fans of UML. A decade later the perspective is quite different, because the issue is not seen as whether to use UML or SDL-2000 (and other ITU languages), but how to use these together. In retrospect it would be easy to say this was always the way it was seen - but to be truthful this was not the case especially in 1997 and 1998 when SDL-2000 was being formulated. However, it is now clear that the state machine specification part of UML 2.1.2 is not really a complete language in itself, because of the semantic and syntactic variations that are allowed, and that to make its use practical an (implicit or explicit) profile for UML has to be used. The revised Z.109 profile of 2007 is geared to the needs of the telecommunications industry by mapping UML 2.1.2 onto the more precise (and therefore more practical) Z.100 semantics and (where UML 2.1.2 gives notation options or no specific notation or no notation) binding to the Z.100 syntax.

It is not by accident that the situation has been reached today where UML and ITU System Design Languages are seen as complementary rather than competing. Between 1999 and 2004 there was a significant involvement of ITU System Design Language experts in the ongoing development of UML, in particular for UML2.0. The ITU languages have the good features of being well-defined and having action semantics that ensure specific behaviours. UML is good at object modelling and has proven to be a success at providing a framework for using different languages together - a feature that the ITU languages (for historical reasons) lack. Rather defining new precise action languages for UML, or adding a framework scheme and object modelling to the ITU System Design Languages, the sensible way forward from a telecommunications system engineering point of view is to combine these features of both approaches.

It was therefore not a surprise to see the industry use tools that combine UML with the SDL-2000 semantic engine. This was the perspective of several major telecommunications manufacturers, and therefore the general direction of industry.

However, the situation with SDL-2000 after nearly a decade is unsatisfactory for all parties. Despite the 1996 1999 intention to ensure the language standard and tool support should be closely aligned (of course, ideally the same), this was not reality in 2006 through to the start of 2009. The language available to users is effectively SDL-92 with the 1996 addendum plus some of the features of SDL-2000 (depending on which tool is used) and often using legacy syntax for data. It was to ensure users are still able to produce SDL-models that are valid according to the standard that Annex B was added to Z.100 for SDL-2000, which allows the legacy syntax supported by tools.

Not only was SDL-2000 not fully supported, but also as UML and other languages such as the ITU-T User Requirements Language, become more commonly used there have been changing expectations of the facilities offered by the Specification and Description Language. Some, such as UML-like syntax, do not seem to be required. Others, such as time-supervised states, that are not included in SDL-2000 seem to be desirable.

2 The data issue

A data model that provides data with both sets of values (as in ASN.1) and operations is essential for any language that is to provide executable models or implementations. SDL-2000 made a major change to the way that data was defined: the algebraic axiom approach was removed from the user language and leaving just a constructive data approach (as in most programming languages). At the same time object data types were added. Leaving aside whether reference (object) data is actually needed and the “modernized” syntax, it was reconsidered if SDL-2000 data is the best approach for SDL-2010.

An SDL-2000 user is faced with the option of using either ASN.1 or SDL-to define data types. If ASN.1 is used SDL-2000 provides a built-in set of operators. Similarly the built-in SDL-2000 data types provide a set of defined operators.

The only real advantage of the SDL-2000 data types over ASN.1 is an arguably nicer syntax. The language could be made simpler by removing the SDL-2000 data types, but this would not be acceptable for legacy reasons and Z.109 (06/07) essentially incorporates the SDL-2000 data types.

Tools that produce target code for SDL-2000 are usually proprietary products of larger companies. Commercial tools usually implement ASN.1[1] and SDL-2000 data in one of two ways: providing translation to another programming language (usually C or C++), or producing code for a virtual machine and providing an emulator for that machine (written in some other language like Java or C). The advantage of either of these approaches is that they are target machine independent. However, there remains the issue of interfacing code from SDL-2000 with other code, especially device and message handlers and possibly the RTOS.

An alternative is to open up the language to external data types. In fact this was envisaged in SDL-92, with the **external** data syntax, but (as seen from the MSC and UML experience) it is difficult to define a language that can use the declarations and expression syntax in a plug-compatible way. Moreover, Z.121 now provides an MSC to SDL-2000 data binding. Also from the user viewpoint the meaning of an expression in SDL-2000 would depend on the actual data language used, which may not be clear from context. As with MSC, there are requirements on any data language used, so that data is compatible with essential features such as timers. Despite these issues, from a user point of view using a data expression notation from another language can be a practical approach (as evidence see SDL-RT). The plan therefore for SDL-2010 is to first ensure SDL-2000 data is supported, and then define a way of providing a binding to other language syntaxes such as Java, C (or C++) or the data language of SDL-RT.

3 Diagram structure

It has been agreed for some time that although the Specification and Description Language has both a graphical (SDL/GR) and textual (SDL/PR) presentation form coupled by a common abstract grammar, the primary presentation form is graphical, and the textual form is used mainly as an interchange format. Specifications in the language therefore usually consist of a number of diagrams. While in SDL-2000 it is permitted diagrams for inner components to be drawn nested inside the diagrams for the enclosing component, in practice this is not done for two reasons: in general the resulting diagram would be too large, and full tool support is not provided. Even though some tools support the printing of such physically nested diagrams to some extent, diagrams are usually generated separately, and for any reasonable size system nested diagrams become too extensive to handle, read or comprehend. Instead inner diagrams are referenced from enclosing diagrams, so that each diagram is of a reasonable size and this is the form supported by tools.

[1] Whether ASN.1 compilation is done in a separate tool or not is a tool issue, not a language issue.

On the other hand, the semantics of the language is defined in terms of a single hierarchical model in the abstract syntax, in which the referenced diagram replaces each reference (after eliminating any duplicates). In SDL-2000 the change from references to the hierarchy this is done by transformations to a nested concrete syntax (not supported by tools) and then this concrete syntax is mapped to the abstract syntax. In SDL-2010 the change from references to the hierarchy is done when by mapping referenced diagrams directly to the abstract grammar. In the concrete syntax the nested graphical form (which is not generally tool supported for diagrams in any case) is no longer part of the language. This is a worthwhile simplification. To some extent, moving SDL/PR to Z.106 in 2002 enabled this change.

4 Shorthand transformation models

Since SDL-88 and in SDL-92 and SDL-2000, transformation models have been used to define a number of language features, where a given concrete syntax is transformed into another concrete syntax. These features are often called "shorthand" productions. While these features are often so useful and practical that they are essential, they are not essential in a theoretical sense as the transformed concrete syntax can (usually) be used instead of the shorthand version. In fact the Abstract Syntax and language Semantics are (or at least should be) defined only for concrete syntax that cannot be transformed. It is this canonical syntax that is mapped to Abstract Syntax. The Semantics and (as far as possible) constraints are expressed in terms of the Abstract Syntax.

An objective in SDL-2010 is to keep the core of the language as small as possible (and therefore easier to understand), and as far as possible separate the description of the transformation models from the core parts of the language. This is reflected in a reorganization of the SDL-2010 standard compared with SDL-2000 (see below).

5 Features without formal semantics

Some features (such as comments, paging, create lines, multiple type references) do not add to the semantics of an application model, but are provided to allow annotation to be presented for the benefit of engineers. While these features should be checked for consistency, tools otherwise ignore them. In SDL-2010 these features are separated from the extended finite state machine parts of the language.

In particular, the Association feature was added in SDL-2000, to allow UML-like associations to be shown between types (or "classes" in UML terminology). This had no semantic meaning in SDL-2000, and is arguably now better covered by using UML tools and applying the UML profile in Z.109.

6 Feature deletion, retention and extension

So where does this leave SDL-2000.

As for previous versions (SDL-88, SDL-92) the language definition has had several years of stability, and it was appropriate to consider what change should be made for the new version. This version was initially scheduled for consent in 2008 in line with the end of the ITU study period 2005 2008 (and hence initially named SDL-2008). This was not achieved, but it was decided at the September 2009 ITU-T meeting to change the name to SDL-2010, for the expected year for consent.

As before, one objective is to simplify the language and to have a clearly defined basic SDL: the SDL-Task Force (a small consortium outside ITU-T) was ostensibly set up with this objective, but it seems that this organisation had (October 2005) effectively ceased to exist, and the target of this group was not an SDL-2000 subset. In addition extension proposals for the language have come from many sources such as the SDL-RT, the SDL-Task Force, and industry users. There are ideas considered previously but not incorporated into SDL-2000 and a few ideas to support UML profiling not in SDL-2000. There are some changes in SDL-2000 compared to the previous version that have **not** been widely implemented such as object data, the UML class symbol for types and UML like associations. Some features, such as exception handing have been implemented in just one tool.

Ideally there should have been a critical look at features to be deleted to assess if the potential benefit is worth the complication of having the feature in the language, and if it is likely to be widely implemented and usefully deployed if it is retained. In this case, to justify its existence a feature has to be useful for a reasonable body telecommunications system engineering applications, either for modelling or programming: being theoretically interesting or elegant is not sufficient reason for retention or addition of a feature. Although for feature deletion in the 1996-2000 period, a two steps were adopted (first open discussion on candidate features for deletion; second a list widely circulation for agreement) for SDL-2010 it was proposed the same process need not, because SDL-2000, because SDL-2010 is a richer language the SDL-92 and there is no point in retaining features that have not been widely supported or used. On the other hand, features that are widely used should not be deleted, even if a better alternative exists or is proposed, because it has been found this kind of change leads to significant legacy problems.

A more pragmatic approach has been being taken: some features are being deleted and some potentially useful ones (based on the participating expert contributions tempered by user and tool vendor feedback) are being added. The agreement step is also less necessary, because way Recommendations are approved at ITU-T has changed since 1999, so that significant comments can now be made and handled in a more open way during the approval stage. An important criterion for feature retention or addition is compatibility with UML. There are two reasons for this: UML is a coherent framework for binding ITU-T languages together so SDL-2010 needs to be consistent with the UML model, and provide the needed precise action semantics to UML. The creation of a UML profile for the telecommunications action language (in Z.109 (06/07) for SDL-2000) was obviously a key determinant for this compatibility, and generated a few necessary or highly desirable changes to Z.100.

The current draft has been prepared on the assumption that **exception handling[2] is deleted (while keeping the timers on remote procedures), object data is deleted (or at least simplified), and esoteric features (such as name class) are removed. In addition the Association feature and the use of UML class symbols for types are removed. Features added to support the UML profile are in Annex A.**

7 Re-organisation of the documents for the language standard

SDL-2010 is re-organised so that core features are defined within the Z.101 part of the language definition, with the remaining (retained) more complex language features described in subsequent parts (Z.102, Z.103, Z.104, Z.105 and Z106). In the new organisation Z.100 is re-utilized to provide an overview of the set of Recommendations.

Anyone who has been tracking SDL-for a number of years will be aware that this structure for the language definition is not new: the 1988 version of SDL-defined "Basic SDL" and then a number of additional features that extended "Basic SDL". This structure does not invalidate tools and applications that use the "full" language, while providing an identifiable subset. Initially for a revision of SDL-2010 there was interest and support for identifying a subset of SDL. Some of the proposed benefits of having a clearly defined subset were:

– It makes it easier to teach and learn the basics of the language;
– It makes it easier to produce and maintain tools that can handle such a subset;
– If all tools that claim to support the standard have to support this subset, it gives a level of guaranteed portability.
– Such a subset would characterise essential "SDLness".

Getting agreement on what should and should not be in such a subset is not an easy task. There will be many different opinions backed up by different experiences and value judgements. Work already existed such as studies at ETSI, which could lead a consensus result, so it was argued the potential benefits (some of which are outlined above) would justify the effort. Eventually it was agreed there were insufficient participants really interested in formally defining a subset, so that explicit defining such a subset was not a specific objective SDL-2010.

The objective of the reorganisation has been separation of concerns. The essential behaviour of a system defined using SDL-2010 depends on the extended

[2] Exceptions are still raised by certain constructs (such as indexing OutOfRange), but cause the further behaviour of the system to be undefined because they are not handled.

finite state machine model of Rec. Z.101 coupled with the behaviour of expressions of Rec. Z.104. The other Recommendations Z.102, Z.103, Z.105 and Z.106 provide language features that (respectively): make the language more comprehensive, make the language easier and more practical to use, provide the full data model and action language, enable ASN.1 to be used, and define the interchange format.

7.1 Basic SDL-2010 - Z.101

Recommendation Z.101 contains the part of the Specification and Description Language Recommendations for SDL-2010, that covers core features such as agent (block, process) type diagrams containing agent instance structures with channels, diagrams for extended finite state machines and the associated semantics for these basic features.

7.2 Comprehensive SDL-2010 - Z.102

Recommendation Z.102 contains a part of the Specification and Description Language Recommendations for SDL-2010 that extends the semantics and syntax of the Basic language in Rec. Z.101 to cover the full abstract grammar and the corresponding canonical concrete notation. This includes features such as continuous signals, enabling conditions, type inheritance, and composite states.

7.3 Shorthand notation and annotation in SDL-2010 - Z.103

Recommendation Z.103 contains the part of the Specification and Description Language Recommendations for SDL-2010 that adds shorthand notations (such as asterisk state) that make the language easier to use and more concise, and various annotations that make models easier to understand (such as comments or create lines), but do not add to the formal semantics of the models. Models transform shorthand notations from the concrete syntax of Rec. Z.103 into concrete syntax of Rec. Z.102 or Rec. Z.101.

7.4 Data and action language in SDL-2010 - Z.104

Recommendation Z.104 contains the part of the Specification and Description Language Recommendations for SDL-2010 that adds the data and action language used to define data types and expressions. In SDL-2010 it is allowed to use different concrete data notations, such as the SDL-2000 data notation or C with bindings to the abstract grammar and the Predefined data package. The underlying data model is fundamental to behaviour and provides sorts of data such as Boolean and Integer that are used in other language features. For that reason this underlying model and an overview of predefined data sorts and constructs is given in Z.100 annex D.

7.5 SDL-2010 combined with ASN.1 modules - Z.105

Recommendation Z.105 provides a mapping for ASN.1 modules to features defined in rest of the Specification and Description Language Recommendations for SDL-2010, so that the ASN.1 modules define data items that can be used with the rest of SDL-2010.

7.6 Common Interchange Format for SDL-2010 - Z.106

Recommendation Z.106 provides alternative textual syntax for the graphical syntax items defined in Z.101 to Z.105 that can be used as a Common Interchange Format (CIF) between SDL-2010 tools. The basic level of CIF provides only a textual equivalent of graphical items. The full CIF is intended for the interchange of graphical SDL-2010 specifications (SDL-GR) so that the drawings are recognisably the same.

8 Formal definition (Annex F to Z.100)

The maintenance of the formal definition needs to be considered, and the most likely that the formal definition work will rely on a metamodelling. The current plan is that no formal definition is provided for SDL-2010, due to a lack of resources to modify the existing model or generate a new one. Instead the published Z.100 Annex F for SDL-2000 is referenced. It is therefore note this is out of date, but in combination with the obsolete 2007 version of Z.100 (for SDL-2000) provides a more formal definition for SDL-2000 than currently available for SDL-2010. Most of SDL-2010 is intended to be unchanged from SDL-2000, therefore Annex F to Z.100 with the obsolete 2007 version of Z.100 provides more detail than Z.100 to Z.106 for SDL-2010. If there is an inconsistency between Annex F to Z.100 for SDL-2000 and other parts of Z.100 to Z.106 for SDL-2010, it is either because there is an error in Z.100 to Z.106 or because there is a specific change to SDL-2010 compared with SDL-2000. If a change from SDL-2000 is not documented in Z.100 to Z.106, further study is needed to determine if the inconsistency is an error or intended.

If work is done to replace the formal definition, alternatives are to update the existing model or completely replace it with a new one, in which case a different approach (such as metamodelling) might be considered.

9 Meta-language issues

For further work and maintenance on SDL-2010, some extensions to the language meta-grammar may be beneficial. In this respect the Z.111 standard is relevant, and it needs to be considered whether to update Z.111 to further support any approach taken for SDL-2010 for other ITU languages.

Annex A: Language Changes to support Z.109

A.1 Changes made

The following are a collection of changes in SDL-2010 to better support UML SDL.

A.1.1 Synonym

Synonym is changed to be a “read only variable” and be added to abstract syntax. UML attributes that are read-only can then be mapped to the abstract syntax for synonym.

A.1.2 Lower bound of agent instance sets

It is allowed to specify a Lower-bound on agent instance sets, which by default is zero (as at present). An attempt to interpret a stop in an agent in an instance set that is already at the Lower-bound causes the predefined exception OutOfRange to be raised. If the exception is not handled (the only possibility if exception handlers are deleted), the future behaviour of the system is undefined. This allows the constraint that the lower bound on instance sets in UML SDL-to be simply mapped.

A.1.3 Signal identifier sets

In-signal-identifier-set and Out-signal-identifier-set are extended to contain interface identifiers (see 10.4 Signal List), so that an interface that includes other interfaces can more easily be mapped from UML.

A.1.4 Signals for remote procedures and remote variables on gates, not channels

The remote procedure and remote variable models used to put the implicit signals on channels, but these are placed on gates, and there are implicit gates for the reverse direction instead of an implicit channel. The forward channel is explicit or implicit and has the signals derived from the gates. The reverse channel is an implicit channel created between these implicit gates. The UML SDL-mapping to SDL-is then easier, because the implicit signals on ports map to signals on gates.

A.1.5 Input via

A Trigger with a non-empty port corresponds to an input via a new feature in SDL-2010. The UML SDL-restriction that the port of a Trigger shall be empty can be removed, and a definition how this port maps to the SDL-2010 abstract syntax added.

A.1.6 Time supervised states

SDL-2010 is extended to cover timesupervised states to support TimeEvents. This is a shorthand on state such that every transition leading to the state from another state (or a start) sets an implicit timer. The timer is not set when the state leads to itself (for example in an implicit transition).

A.1.7 Unicode names

SDL-2000 only handles T.50 (International Reference Alphabet) characters except in annotations. UML allows Unicode. This is not really an issue for UML SDL-because any Unicode name can be systematically mapped to T.50 name using a mapping (such as the IETF Punycode RFC 3492 - an algorithm that uniquely and reversibly transforms Unicode strings into the limited character set supported by the Domain Name System). But to refer to such a Unicode name in some SDL-2000 part (to be used with the UML SDL-part) requires the limited character form to be used in the SDL-2000, which implies (1) the user must know the mapping; (2) SDL-2000 allows all the characters in names used by the mapping.

SDL-2010 is extended to allow Unicode names. As far as the SDL-abstract grammar is concerned the issue is only to have a unique Token from the mapping of a name (whether Unicode or not). The concrete syntax is extended to Unicode. The first 127 characters are taken to correspond to T.50 (as before). Two names have the same Token if and only if they have the same UTF-8 encoding.

A.2 Changes considered

A.2.1 Internal transition

Should SDL-2010 be extended to cover internal transition? This occurs without exiting or entering the source state, therefore does not cause a state change, and means that the entry or exit condition of the source state will not be invoked. An internal transition can be taken even if the state machine is in one or more regions nested within this state, and the restriction on UML SDL-can be removed.

A.2.2 Abstract grammar for loops

The abstract grammar for loops is an embedded part of Compound-node in (Z.100 11.14.1), but the relationship between the loop concrete syntax in Z.100 11.14.6 and the abstract syntax in Z.100 11.14.1 is complex. If specific abstract syntax for loops in Z.100 is introduced the mapping from UML may be simpler. This issue applies for both LoopNode stereotypes.