INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2009-2012

STUDY GROUP 17

TD 0469*

English only

Original: English

Question(s): 13/17 Geneva, 16-25 September 2009

TEMPORARY DOCUMENT

Source: Q.13/17 Rapporteur

Title: SDL-2010 Route Map (TAG08)

TAG08r05 – Minor modifications to main text. SDL-2008 renamed to SDL-2010.

This document is the current plan for SDL-2010 based on TD0078 of February 2009, itself an update of TD3498 of April 2008, an update of TD 3424 Sep 2007 (Z.100 Route Map), in turn was based on earlier versions. Note that the Route Map was not updated for the September 2008 meeting. To consolidate information on work to be done, TD3498 included a revised version of TD3429R1 Sep 2007 (Z.100 Changes to support Z.109) as ANNEX 1, and an unrevised version of TD3423 Sep 2007 (Z.100 Action items) as ANNEX 2. These last two items were updated for February 2009. A Specification and Description Language experts meeting was held immediately after the September 2008 SG17 meeting in Geneva on 20-22 September 2008. During this meeting a number of the action items were reviewed and Annex 2 was updated. The rapporteur subsequently continued to work on the action item list and made further changes.

1. Background, history and status of SDL-2000

SDL-2000 was completed in 1999. Since then there have been some minor updates to the Recommendation. The text was re-organised in 2002 so that Z.100 describes the graphical language and the textual common interchange format appears in Z.106 and 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 SG10 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 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, will probably not offer full SDL-2000 support, because it has a smaller (at least in value terms) share of the ITU

Contact: Rick Reed, TSE
The Laurels Victoria Road, Windermere
Cumbria LA23 2DL United Kingdom
Tel: +44 15394 88462
Mob: +44 7970509650
Email: rickreed@tseng.co.uk

Specification and Description Language tool market and has to offer compatibility with Telelogic as the market leader. Cinderella has collaborated with Humboldt Unviversity 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).

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 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 2003 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 in use in the industry tools that combine UML with the SDL-2000 semantic engine. This is 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. To ensure that

users can still produce SDL models that are valid according to the standard, Annex B was added to Z.100 for SDL-2000, which allows the legacy syntax supported by tools.

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 (to be reviewed). 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. 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 is probably appropriate to consider what change should be made for a new version. This version was scheduled for consent in 2008 in line with the end of the ITU study period 2005-2008; hence the name was SDL-2008. This was not achieved, but

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

it was decided at the September 2009 meeting to change the name to SDL-2010, 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 also many ideas considered previously but not incorporated into SDL-2000 and a few ideas from UML 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 and composite states. Some features, such as exception handing have been implemented in just one tool. The current work is proceeding on the assumption that exception handling[2] is deleted (while keeping the timers on remote procedures), object data is deleted or simplified, and esoteric features (such as name class) are removed.

Ideally there should be a critical look at features 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.

The Specification and Description Language group already went through the exercise of considering features for deletion in the 1996-2000 study-period. A two step process was adopted. In the first step there was an open discussion on which features were candidates for deletion and a list was agreed. This list was then circulated widely for agreement. At that time features were retained for which there was not significant experience, because they had only recently been implemented by tools. It is suggested that the same criterion need not apply in the SDL-2010 study, because SDL-2000 is in any case a richer language and retention should mainly be based on a feature being useful. On the other hand, features that are widely used should not be deleted, even if a better alternative exists or is proposed, because this kind of change leads to significant legacy problems.

A more pragmatic approach is 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. This time more attention will also be given to avoiding legacy issues.

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 the Z.100 language needs to be consistent with the UML model, and the Z.100 language provides the needed precise action semantics to UML. The creation of a UML profile for the telecommunications action language (that is, SDL-2000) as Z.109 (06/07) is obviously a key determinant for this compatibility, and may lead to a few necessary or highly desirable changes to Z.100.

4. Subset definition

There was previously 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 SDL;

[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.

- It makes it easier to produce and maintain tools that can handle such a subset;
- If all tools that claim to support SDL have to support this subset, it gives a level a 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 exists 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.

It now seems that few participants are really interested in formally defining a subset, so explicit defining such a subset is no longer an objective.

However, SDL-2010 is re-organised so that core features are defined within Z.101 part 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). Z.100 provides and overview. 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. This structure does not invalidate tools and applications that use the "full" language, while still providing some of the benefits of a subset.

Specifications in the language usually consist of a number of diagrams, with inner diagrams referenced from enclosing diagrams. On the other hand, the semantics 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). 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. In SDL-2010 the change from references to the hierarchy is done when mapping to the abstract grammar, compared with SDL-2000 where this was done by transformations. This means that in the concrete syntax the nested form (which is not generally tool supported for diagrams in any case) is no longer part of the language. It is a worthwhile simplification. To some extent, moving SDL/PR to Z.106 enabled this.

5. Shorthand transformation models

Transformation models 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 useful and practical, they are not essential (in a theoretical sense) as the derived 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 therefore proposed that (to keep the core of the language as small as possible and therefore easier to understand) transformation models are generally described separately from the core parts of the language.

6. Features without formal semantics

Some features (such as comments, paging, create lines, associations, 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.

7. Meta-language issues

To support the above proposals Z.100 is restructured. Some extensions to the language meta-syntax may be required. The structure and language meta-model should be considered. In this respect the Z.111 standard is relevant. Some parts of Z.100 are replaced by references to Z.111. It may be necessary to update Z.111 to further support the approach taken for SDL-2010.

The maintenance of the formal definition needs to be considered, and the most likely that the formal definition work will rely on a metamodelling. Another possibility is that no formal model is provided for SDL-2010, due to a lack of resources to modify the existing model or generate a new one.

8. Summary of agreed strategy

1. Data is being re-examined (taking SDL-2000, UML, SDL-RT and MSC experience into account). The use of legacy syntax and the use of embedded C (C++ or other programming languages) is considered.
2. The language feature set is re-considered taking into account: experience, UML compatibility, usage, legacy and language size (bigger is not better). The resulting language should ideally be no larger (and preferably smaller) than SDL-2000.

ANNEX 1

Language Changes to support Z.109

The following are a collection of changes that to be made to SDL-2010 to better support UML SDL.

1. Synonym

An SDL synonym should be 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.

2. Lower bound of agent instance sets

There should an extension to include a *Lower-bound* on agent instance sets, which by default would be 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 and the agent would continue to exist. If the exception is not handled, the future behaviour of the system would be undefined. This allows the constraint that the lower bound on instance sets in UML SDL have to have the expected meaning.

3. Signal identifier sets

*In-signal-identifier-**set*** and *Out-signal-identifier-**set*** should be extended to contain interface identifiers (see 10.4 Signal List). Then an interface that includes other interfaces can more easily be mapped to SDL.

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

The remote procedure and remote variable models currently put the implicit signals on channels, but these should really be placed on gates and there should be implicit gates for the reverse direction instead of an implicit channel. 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.

5. Input via

A Trigger with a non-empty port corresponds to an **input via** – an extension to Z.100. It is agreed to update SDL-2010 with this feature, but the detailed change to has not yet (Apr 2008 and Sep 2009) been prepared. Currently there is a restriction that the port of a Trigger shall be empty. The SDL-2010 change allows this constraint to be removed in Z.109, where a definition how this port maps to the (revised) SDL-2010 abstract syntax should be added.

6. 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 this in Z.109 can be removed.

7. Time supervised states

Should Z.100 be extended to cover timesupervised states to support TimeEvents?

8. 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.

9. Unicode names

SDL only handles T.50 characters except in annotations. This is probably not an issue 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). To refer to such a Unicode name in some SDL-2010 part (to be used with the Z.109 UML SDL part) would require the limited character form to be used in the SDL-2010, which implies (1) the user must know the mapping; (2) SDL-2010 allows all the characters in names used by the mapping. On the other hand SDL could be 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). It is suggested to extend SDL to Unicode – the details need to be worked out.

ANNEX 2

SDL-2010 Action Items

References:

TDw6xx. nn[.mm] is TDw6xx, section nn[.mm] of an Experts Meeting. w is a letter, xx are digits.
Before Apr 2000 letter w (usually) 1st letter of the meeting location (e.g. T= Toulouse)
For example TDT612.3.4 is the Toulouse meeting temporary document 12 paragraph 3.4
TDwxx.nn[.mm] is TDwxx, section nn[.mm] of an Experts Meeting. w is a letter, xx are digits.
The letter w was allocated for each meeting starting with the A = Oslo April 2000
For example TDC12.4 would be for meeting 'C' temporary document 12 paragraph 4
TDxx.nn[.mm] refers to a TD from the Geneva 2000 SG10 meeting.
T01-3xxx.nn[.mm] refers to a TD of SG17 2001-2004.
T05-3xxx.nn[.mm] refers to a TD of SG17 2005-2008
T09-3xxx.nn[.mm] refers to a TD of SG17 2009-2012
An expert's name, followed by a date, refers to an email sent by that expert regarding this issue.
Emails later than 26 Nov 2000 can be found at <http://sdl-forum.org/Archives/meeting/>.
A reference to Delayed Contribution (D10.xx nn[.mm]) is for SG10 2001-2004.
A reference to Delayed Contribution (D17-01.xx nn[.mm]) is for SG17 2001-2004.
A reference to Delayed Contribution (D17-05.xx nn[.mm]) is for SG17 2005-2008.

Tables:

The tables below summarise open items that have been agreed upon during Experts meetings. Each item in the table is given a reference number (such as Open 1) for reference in other documents. Any such reference needs to refer to the issue of this document, as the numbering is automatic. The second column gives the Expert(s) who originally accepted responsibility for ensuring completion of that item (see initials listed under "**who**" below). However, many of these names are only partially relevant because some of the individuals have not participated in the work for some time. This document is derived from TD3313. TD3313 was derived from TD3187. TD 3187 was derived from TD3108R1. TD3108 was derived from TD3053 of the SG17 meeting in Moscow April 2005, which was derived from TD3251 of the meeting in Geneva, July 2004.

Type Items are classified as Deficiency (D), Clarification (C), or (F) Future Extensions.

Priority Items are triaged into high (1), medium (2), or low (3) priority.

Who

AE -Anders Ek	All – Everyone	AP - Andreas Prinz
BMP - Birger Møller-Pedersen	EH - Eckhardt Holz	Hum - Humboldt Univ
MvL - Martin von Lövis	RR - Rick Reed	TW - Thomas Weigert

Table 1 Open to do items. 11

Table 2 Open to do items related to data. 13

Table 3 Solution determined but not included in Z.100. 17

Table 4 Open Questions or Actions. 18

A √w indicates that the issue was agreed at the w meeting where w= P for Ottawa June 2004 experts' meeting + Geneva July 2004, Q = Windermere Jan 2005 + Moscow April 2005, R = Grimstad Jun 2005, S = Windermere, Sep 2005, T = Geneva Oct 2005, U = Geneva Feb 2006, V= Jeju Apr 2006, W = Kaiserslautern May-Jun 2006, X = Paris Sep 2006, Y = ETSI Oct 2006, Z = Geneva Dec 2006, AB = Geneva Apr 2007, AC = Geneva Sep 2007, AD = Geneva Apr 2008, AE Geneva Sep 2008, AF = Geneva Feb 2009.

Table 1 Open to do items.

Ref.	Type	Prio	Who	Action description
Open 1. D10.19/ TDC32r02.2 TDC32r02.16 TDC32r02.17 TDC32r02.18 D10.19/ TDC32r02.10	D	1	BMP	*In TDB18 it is said that "A state represents either a basic state or a composite state application." However, this should be expressed in terms of the abstract syntax, where there is no concept of a basic state, nor composite state application.* *(Note also that the semantics of composite state application is claimed to be in Z.100 11.11, where it is not.)* I (BMP) take this action point together with the one on confusion between type and instance level. I (BMP) propose to do it the same way as it is done for agents. However, it is a rather large restructuring. As part of this I encountered that a composite state cannot be a specialisation of a state type, while this is possible for agent definitions. I (BMP) propose to change the terms starting with "composite state type" to just start with "state type". The reason is that a "state aggregation" is also a composite state. The terms here were introduced in a rush, and that shows, so we may even want to take another round on this. I (BMP) also propose to remove the "composite" from "composite state type definition", as there will be no state type definitions that do not define composite states. The right term would be something like "state contents" (in contrast to the properties specified in its occurrence/application in an enclosing state or state machine). This would also defend the use of the keyword **substructure**! I (BMP) propose the term "plain state body> for a state contents body that are just plain contained states (in contrast to state partitions). Geneva (Jul 2004) comment: The solutions proposed, while addressing an outstanding issue, were based on an outdated version of Z.100. Everybody was encouraged to read this section and come up with more specific ideas on how to improve it. This item also covers open items 5, 6, and 7 of T05-3053. Open item 14 of T05-3053 may also be resolved by this item (see D10.19/TDC32r02.10 for the issue). May need answers to the questions in the comments of TDB18 but possibly resolved by implementation of paragraph 2 of D.19/TDC32r2. **February 2009: to be done. Should be revisited with the objective of good alignment between UML2 and SDL.**
Open 2. D10.21/ TDC18r01	D	1	TW (+ BMP ?)	It was pointed out that after the introduction of composite states there is an opportunity for further harmonization between tasks with exit points and composite states, and also, for harmonization between procedure and composite states. Study of the convergence of composite states, tasks, and procedure was added to the open item list. It was discovered that there exists a problem with returns from composite states in that a return from a composite state only exits the composite state rather then a surrounding procedure. However, a value return (e.g., "**return** 3+x") would exit the procedure. This is, of course, highly undesirable. This raises the following questions: ▪ Are the concept of exiting a procedure and exiting a composite state different and need different keywords/symbols? ▪ If these are the same concept, should it be possible to qualify the return to make clear where one returns from?

Ref.	Type	Prio	Who	Action description
				This problem needs to be addressed in the following study period and is put onto the open item list. In light of this issue, the extension proposed in TD33 was retained for further study in order to ensure that a solution to the above problem will be consistent with the extension proposed and vice versa. However, the extension proposed was deemed useful and satisfying a user need. **February 2009: Still to be done. Should be revisited with the objective of good alignment between UML2 and SDL.**
Open 3. TDF18, TW, Thu 10/19/00 13:20	D	3	TW	In the AS0 we have defined the <compound action> to serve as a place for transforming the algorithmic subset of SDL into. There we can attach a standard exception handler. As far as the semantics of exception handling is concerned, all what is said is that "Handling an exception instance results in a transition. The state of the process or procedure is not changed." (11.16.1). The transition this speaks of is, of course, the transition attached to the exception handler. What happens after that transition is completed depends on the semantics of the Compound-node (which corresponds to the <compound action>). There is nothing that says we cannot, after the transition of the attached exception handler has been interpreted, continue the interpretation at the node after the Compound-node. The only issue is that transitions need to be terminated, so we have to hook them up appropriately.... And the text in Compound-node has to be written that explains the exception handling, and the On-exception has to be added to Compound-node, as this was forgotten. **September 2008: To be done – but not needed if exceptions dropped. Otherwise it was agreed this should be fixed as advised by TW.**
Open 4. D10.29/ TDC29	C	3	RR	Virtuality for a state diagram The contribution proposes to allow defining a state diagram to be virtual with the intention that it should be possible to redefine the state diagram. It was agreed to put the issues on the Todo list and potentially also discuss it further. **September 2008: To be done.**
Open 5.	F	2	RR	Should items added in SDL-2000 to support UML-like modelling (such as associations) be deleted, because the Z.109 UML profile would be used to express such models? **September 2008: It was agreed this should be done. SDL-2008 removal.**
Open 6 Was Question 22 in TD3498 of April 2008		1	RR was TW	Change the static constraints on the usages of <gate property area>s so that they are valid even if the referenced definitions are textual (currently they specifically make reference to diagrams). See the formulation in TDC22, at the bottom of section 14.3. **After Sep 2008: Subsequent to this comment being made, the textual syntax was removed to Z.106 and a textual definition is assumed to be mapped (or a least able to be mapped) to the graphical form so that change to the constraints were not needed. Looking at <gate property area> again, and considering that UML-like type references are to be removed in SDL-2008, it is now proposed to delete <gate property area> entirely. Question moved to Open section.**

Table 2 Open to do items related to data.

This list is an input to the review of data in SDL for SDL-2010. Because it is intended to revise the (unimplemented) object part of SDL-2008 and many of the items are concerned with object types, the issues may become irrelevant.

Ref.	Type	Prio	Who	Action description

Data 1
TD29.1.1
	C	Data	TW	Are these signatures distinct:
				f1(value Z)->value Z;
				f1(object Z)->value Z;
				f1(value Z)->object Z;
				f1(object Z)->object Z;
				September 2008: to be decided.

Data 2
TDD03r1.4
	C	3	TW	The following examples were discussed. Assume the following:
				`value type vt {struct a Integer;}`
				`value type subvt inherits vt {struct b Integer;}`
				`dcl v vt, ob object vt, subob object subvt;`
				`subob := (. 1, 2 .);`
				`ob := subob; /* Static error because it is not sort compatible */`
				`v := ob;`
				Is this example statically correct? If yes, what happens at runtime? It was agreed to investigate this issue further.
				`object type ot {struct a Integer;}`
				`object type subot inherits ot {struct b Integer;}`
				`dcl v value ot, ob ot, subob subot;`
				`subob := (. 1, 2 .);`
				`ob := subob;`
				`v := ob; /* Dynamic error ? */`
				It was discussed whether the above cases are different or the same. It was agreed that according to the current Z.100 they are not the same, but there was no agreement whether that was desirable or not.
				Another interesting example was discussed:
				`value type vt {struct a Integer;}`
				`value type subvt inherits vt {struct b Integer;}`
				`dcl v vt, ob object vt, subob subvt;`
				`subob := (. 1, 2 .);`
				`ob := subob; /* Static error? This hinges on the interpretation of suporsort in the rules for direct compatibility (clause a) */`
				`v := ob;`
				`September 2008: to be decided.`

Ref.	Type	Prio	Who	Action description

Data 3
 	
TDD03r1.7
	C	3	TW	Another issue is illustrated with the following example:
				
				`object type small {`
				`  choice a Integer;`
				`  method checkA() {`
				`  if (this.Present()=a) { output s1; }`
				`  if (this.aPresent()) { output s2; }`
				`  }`
				
				`object type large inherits small adding { choice b Boolean; }`
				
				`dcl s small, l large;`
				`s:= <<large>> a(20);`
				`s.checkA(); /* Will NOT send the signal s1. Will send the signal s2. */`
				`s := <<large>>b(True);`
				`s.checkA() /* Will not send s1 or s2 */`
				
				One issue is what is the value returned by *this.Present* in the *checkA* method for the last *s.checkA* call. One potential solution for this question that was proposed is to define a special literal in the enumeration type returned by Present to represent this.
				
				September 2008: to be decided.

Data 4 TDD03r1.8	C	3	TW	It was pointed out that the current definition of the Make operator includes all fields of the data type including private and protected fields. This violates the idea behind the distinction between private, protected, and public fields. It was agreed to study this topic further. Discussed solutions where: ▪ Define only a nullary public *Make* operator. ▪ Only define public *Make* operators by default. Protected and private *make* operators would have to be explicitly added by the user. ▪ Define different private, protected, public *Make* operators by default, if private, protected, or public fields, respectively, are present in the data type definition **September 2008: Requires further work. Harmonize with UML2?**
Data 5 Geneva 2001,TD48	C	3	TW	Can we allow redefined/finalized in <argument virtuality> also, i.e., just use <virtuality> there as well? **September 2008: to be decided.**
Data 6 TD10.14	D	3	MvL	Missing transformations for handle statement. The transformations for asterisk and longer lists are missing. **September 2008: to do. Issue disappears if exceptions removed.**

Ref.	Type	Prio	Who	Action description
Data 7 TDD03r1.3	D	3	TW	Z.100 12.15 It was agreed that the explanation of *clone* needs to be improved: i) The first sentence uses X in two cases where Y was meant; ii) an implicit variable for X needs to be created in the informal definition; and iii) it needs to be defined what the return value is. The intention was that clone should be similar to *copy* but for the construction of a new object, if applied to an object. For values the *clone* operation should return a copy of the same value as was given as actual the argument, which amounts to returning the same value as was given as the actual argument. **September 2008: to do.**

Data 8
TDD03r1.3
	D	3	TW	It was pointed out that permitting to leave out the trailing commas in operator applications may result in ambiguities when there is more than one type with similar signature. In particular, the *Make* operator will likely lead to such ambiguities. However, an example was given that supports allowing the trailing arguments (and trailing commas). Consider:
				`object type Short { struct a Integer; method twice -> this Short { return (. this.a * 2 .); } }`
				`object type Long inherits Short adding { struct b Boolean; }`
				Assuming that the *Make* operation is resolved after inheritance has been taken into account the application of *Make* in the body of *twice* applied to a *Long* object would call the *Make* operator of *Long* with a single argument.
				It was also pointed out that this example seems to motivate allowing qualifiers to include ***this*** (e.g., qualifying *Make* with <<***this*** *Long*>> in the above example.
				September 2008: to do.

Data 9 TDD03r1.7	D	3	TW RR?	TD31 from the Geneva meeting November 2000 was discussed. The contribution changes the model for choice to take inheritance into account. It was agreed that there is a problem both with the current definition in Z.100 and with the model proposed in the TD. The problems were mainly that in a specialized type there would be more than one *Present* field and also more than one field representing inherited choice fields. It was also pointed out that there is a problem with the proposal with the return type of the *PresentExtract* operation. **September 2008: Some agreed clarifications have been made, but the item remains an issue. A model that works is needed.**

Ref.	Type	Prio	Who	Action description
Data 10 TDD03r1.3	F	3	TW	A potential problem with the *equal* operation is that if we have two object types that only have Any as their common ancestor then result of *equal* is *False* albeit a static error might be more informative for the user. It was agreed that the preferred semantics would be to give a static error instead. The same would hold also for *equal* applied to *Pid* and an object sort. Application of *equal* in this situation would preferably be statically illegal which could be achieved by either *Pid* not being a subtype of *Any*, or by changing the sort compatibility rules to ensure that *Pid* sorts are never compatible to object sorts. It is not even clear from Z.100 if *Pid* inherits from *Any* or not. It was agreed that this should be investigated further. **September 2008: Still waiting further investigation.**
Data 11 TDD03r1.3	F	3	AE	It was discussed whether it should be allowed to assign to **this**. It is currently not allowed, but there may not be a problem if one were to allow such assignments. It was agreed to investigate this issue further. **September 2008: Still waiting for further investigation. Potentially useful.**
Data 12 TDD03r1.3	F	3	AE	Optional fields were discussed. It was agreed that there might be interest in a method that 'unsets' an optional field and makes it not present. **September 2008. To be done**
Data 13 TDD03r1.3	D	3	TW	It was pointed out that *num*, *succ*, etc., are defined as operators. This causes problems with subtypes of object literals, because the operator in the supersort (which may not know anything about the relevant properties of the subsort) is always called. **September 2008. Try to resolve in SDL-2010**
Data 14 (was Open 7 in T05-3108R1) TDD03r1.11	D	1	RR	Agreed to investigate how concatenation of Bitstrings behaves with respect to padding. It was unclear what the semantics of ASN.1 was for Bitstrings. It was agreed that for most users the intuitive semantics of the padding would be that the padding is done in the leftmost bits of the string. It was also agreed that intuitive semantics is that the rightmost bit is the least significant bit. This has an impact on the *num* operator. *num('10'B)* should with this semantics give 2, not 1. **September 2008: Still to be done. Needs consensus.**
Data 15 TD29.1.2	D	2	RR	For some reason it is not allowed to specify directly that <anchored sort> is always inherited as the **value** or as the **object** variant of sort of the data type. If this was intentional, what is the reason? It seems desirable the define that an inherited operator/method parameter always takes a **value**(**object**) and the result of an operator/method always has a **value**(**object**) result, regardless of whether the derived data type is **value** or **object**. As it is, if this feature is needed then a convoluted form using **syntype** has to be used. **September 2008: This needs to be resolved.**

Table 3 Solution determined but not included in Z.100.

Ref.	Type	Prio	Who	Action description	Res.
There are currently (September 2008) no cases of solutions that have been determined but not added to the Implementers' guide and Z.100(11/07). New issues to be resolved in SDL-2010.					

Table 4 Open Questions or Actions.

Ref.	Who	Question or Action description
Items in this table have probably been resolved or are obsolete or both, but the list has been retained so that a check can be made that this really is the case before the items are deleted.		
Question 1	RR	See email from RR, Tue 9/26/00 2:21 PM: References to types - followed by an identifier in PR or [qualifier] name in GR. There IS some inconsistency between the PR and GR here, and I [RR] think it was agreed it should be optional qualifier followed by name in both cases - the rationale being it is that the reference corresponds to defining instance of the name. For package references there is again an inconsistency. PR has [<qualifier>] name whereas GR has <identifier>. **September 2008 - inconsistency still exists in Z.100 and possibly Z.106 (not checked). <identifier> is still used in some cases. Needs to be corrected in SDL-2010.**
Question 2 TDB03.5.2		Implement changes in TDB03.5.2 This concerns complete valid input signal set and, the handling of implicitly consumed signals and asterisk input. It is clear that: " An <asterisk input list> is transformed to a list of <input area>s, one for each member of the complete valid input signal set of the enclosing <agent diagram>, except for <signal identifier>s of implicit input signals introduced by the concepts in 10.5, 10.6, 11.4, 11.5 and 11.6 and for <signal identifier>s contained in the other <input list>s and <save list>s of the <state area>." Is inadequate for the case of a <asterisk input list> in a state partition. In this case the valid input signal set of a partition should be used. For a procedure used in different partitions, the valid input signal set will be dependent on which partition it is called in. **September 2008: requires further study - but it is probably well-defined in Annex F.**
Question 3 TDB03.8.2		Virtuality for save is not kept in the abstract syntax. It needs to be checked that this is correct and clarified if virtuality has a runtime semantics. **September 2008: requires further study**
Question 4 TDA13.12	RR MvL	Enabling condition model. Is this wrapping of exception necessary in other places, such as continuous signal, or is it a left-over from before addition of direct semantics? **After Sep 2008: It is not obvious why a procedure is needed. It is suggested to remove this model. RR to check with MvL.**

Ref.	Who	Question or Action description
Question 5	TW	The Models for the statement list group of items do not work. The problem is that a statement list cannot contain actions, and therefore the transformations cannot be local. In a way, the complete statement list has to be transformed into a transition or something. This is different from the current handling of Models, where we have only a local transformation. AP The following solution is proposed: Just state how the items get mapped to the abstract syntax. This is possible in almost every case. This does imply that we would have to copy the static conditions from the action counterparts to the statement list counterparts. However, one problem remains: What to do with the exception statement? The problem here is to define the exception - where can this be done? We could for simplicity add handlers to the Compound-node, but this has to be checked by the exception experts.
Question 6 TDA13.3	TW	Referenced data, signal and interface definitions: Do we allow data definitions etc., to exist as stand alone textual definitions?
Question 7 TDB03.10	TW	<loop clause> improvements
Question 8 TDB20.44	TW	In rule <expression>: <value returning procedure call> vs. <operand>: this is really bad, because the precedences are harmed. AP suggestion: use the following productions <value returning procedure call> ::= <procedure call> | <remote procedure call> <simple value returning procedure call> ::= { <procedure identifier> | <remote procedure identifier> } [<actual parameters>] and insert an alternative like below <operation application> ::= <simple value returning procedure call> | <operator application> | <method application> TW: Note that we cannot move the **this** down to operator application et.al., neither can we move the variant of procedure call using actual context parameters. It is not clear whether the "harm" to the precedences can actually arise. Look for an example where the fact that one has to parse some variants of procedure call together with operator application will cause a problem.
Question 9 B20.x	TW	Consider fixes to "nice to haves."
Question 10 A12.4.10	TW, RR	Resolve the disagreement on how this issue should be handled. **The change was left in Z.100, albeit it is not yet agreed upon (see TDB03).** Comment TW: The grammar changes should be as follows. <package diagram> ::= <frame symbol> ***contains*** { <package heading> { {<package text area>}* {<diagram in package>}* <graphical package use area>* } ***set*** } [***is associated with*** <package use area>]

Ref.	Who	Question or Action description
		<diagram in package> ::= <package diagram> | <package reference area> | <type in agent area> | <data type reference area> | <signal reference area> | <procedure reference area> | <interface reference area> | <create line area> | <option area> <package reference area> ::= <package symbol> ***contains*** <package identifier> <graphical package use area> ::= <dependency symbol> ***is connected to*** { { <package reference area> } { <package diagram> | <package reference area> } }***set***