SGFS/N439

SGFS Subgroup Meeting
Issues List
November 1st, 1991

Issue 1: Terminology: choice between AEP and OSE.

OSE is used (even cited) by N402 and hence supported by resolution 11 from JTC1.
Resolution 18 uses the term AEP.
Take both terms, and see how far we can come.
Should the definition be in TR10000-1 or in the OSE part of TR10000?
Assumption: 'or' in OSE definition (TSG-1, pg 57) should be read as an 'inclusive or' or as 'and/or'. Clearly not the intention that the 'or' implies multiple types of Open System Environments.
The following definition for AEP (based on the TSG-1 definition) is used: "The specification of a complete and coherent subset of the Open System Environment, together with the identification of the applicable classes, subsets, options and parameters of the referenced standards, necessary to support a class of applications".
Rationale for changes:

- the definition of OSE is suffucient comprehensive to provide a unique, global concept, of which there can only be one - hence "the", not "an";
- expand "options and parameters" to read "identification of the applicable classes, subsets, options and parameters of the referenced standards" - this picks up the generic definition of "Profile" in TR10000 and shows that AEP is one member of the class of "Profile";
- delete final part "for interoperability ..." since these terms are already included in the OSE definition already referenced.

Action: Implement proposed text in TR10000-1.3.
Status: Closed

Issue 2: New Functionality.

Should a profile be allowed the define new functionality (which is for example special for a specific application area) in a profile? Can functionality be specified in the profile without that functionality being specified (and needed) elsewhere?
Annex A (TSG-1) is relevant, as well as TB-7 item 3 and 4.
SC24 seems to have profiles which define such additional functionality.
It is noted that a profile may require functionality that is not specified in a base standard, does not 'naturally' fits in an existing base standard, but that will never warrant a complete base standards on its own (the 'grey area' problem).
Current uderstanding: new functionality should be added to the base standards.
Action 1: send an informal liaison statement to SC24.
Action 2: Review the examples on "Grey Areas of New Functionality" in Annex A.

SGFS/N439

Status: Open

Issue 3: Gaps.

What is a gap? How should gaps be identified? Can an profile that identifies gaps (has gaps) become ratified? See also TR10000-1, section 6.1 which (seems to) define how this should be done.
TB-7 item 3 and 4 apply to gaps (Gaps are functional requirements in the domain of the standardized profile that are not met by approved standards).
Gaps can be (should be?) classified: either the bit that is missing makes to whole profile useless or only a small piece of functionality is missing but the 'gross' functionality of the profile is still useful. In the first case probably NWI proposals should be created, in the other case something less drastic is needed??
Suggestion: let the submitter of the profile assess the gap. When gaps are identified, they should not be filled by specifications in the profile.
Gaps could be identified in an informal annex.
Profiles can be defined in 2 parts: first part which defines the functional requirements addressed by the profile, the second part defines the mapping of the functionality to the base standards; the ratification of the 2 parts should be handled in a consistent way, but could be separate.
General agreement: the ISPs should only contain the accepted parts, the rest should go into informative annexes.

==

Discussion of the issue of non-existing Base Standards ('gaps') needed in ISP definitions.

First a collection of individual positions taken and comments given during the discussion are presented (these are not consensus positions):

*SPs should not include vendor or consortia specifications, or Public Specifications. Also entries like 'user selected' or 'vendor defined' should be avoided.
Gaps in the inventory of Base Standards have different sizes and weight:

- central to the ISPs intended functionality
- minor to the ISPs intended functionality
- gaps which have associated standardization work to fill them
- gaps without work on them
- very small gaps in Base Standards for provision of 'glue'.

To preserve the tightness of ISPs and their verification, ISPs need to be precise and fully defined, without references to non-existing Base Standards.
There is value in the identification of gaps in context with a specific Profile, because this can lead to standardization efforts to cover the gaps. Also the Profiles purpose becomes more explicit if gaps are described.
OSI ISPs have gaps today, they are just left out of the document for later work.
*OSE ISPs with gaps would be kept as PDISPs or DISPs until completed.

OSE ISPs will never be complete due to technology progress and changes in user requirements. Therefore SGFS may have to stay out of ISPs for OSE.
There is time pressure. OSE Profiles will have a shorter life time (2-4 years) until reworked.
ISPs could have a set of complete definitions in the normative part. Also there could be an informative part with context and known gaps.
Even if incomplete Profiles are defined, their stable part can be recorded as a core Profile. For the complete specification, a taxonomy entry could be made early on, but without any specific specification yet.

Second: A proposed wording for the bullet under discussion follows:

3.1.2 Profile: A set of one or more base standards .....

6.1 During development of Profiles, functionality may be required which is not yet covered by base standards or Profiles. In this case a Profile can refer to missing standards in an informative part, but its normative part shall refer only to existing base standards and ISPs. If in the future a more extensive Profile is envisaged, a placeholder can be entered in the Taxonomy.

==

Action 1: Result should go into TR10000.
Action 2: Richard Lloyd to review text for section 6.1
Status: Open

Issue 4: Subsetting and Options.

Should it be allowed that profiles define subsets from base standards that are not defined by the group that defined the original base standard?
Subsetting seems to be more a problem of the base standards rather than of the profiles. Requirements for subsetting a base standard can result from the development of a profile; the actual subsetting should be done by the 'owner' of the base standard.
TB7, items 5 and 7 apply to this issue. Also Annex A of TR10000 should be considered.
Subsetting of profiles as opposed to subsetting of base standard should be investigated. This issue also applies to options.
Implementor defined options issue.
A document should be considered with guidelines for standards writers indicating what consequences there are when their base standards are included in profiles.

Action: Review the examples on "Conflicting Options" in Annex B.
Status: Open

Issue 5: Taxonomy.

It is observed that a taxonomy for OSE is less stable than an OSI taxonomy.

Work is underway on the development of an OSE taxonomy.
EWOS has proposed a taxonomy (ETG 12, SGFS N337) and is awaiting some response. TSG-1 report has also some references to taxonomy (Annex C TSG-1 report).
It is noted that there is no editor for TR10000-3.

Action 1: SGFS likes to be kept informed about developments by its members in the area of the taxonomy for OSE.
Status: Open

Issue 6: Base Standards.

Para 4 (section 6.1): need more words to extend the concept of Base Standard (the definition is recursive). The emphasis will be on profiles referencing other profiles rather than base standards.
One of the problems seems to be that the definition of base standards (3.1.5) includes TRs.

Action 1: Review clauses 3.1.5, 6.1 and A.4.3 for acceptability of this solution;
Action 2: Incorporate in TR10000-1.3.
Status: Open

Issue 7: Conformance testing.

Are ISO 9646 and/or IEEE 1003.3 applicable to OSE?
It is noted that base standards, other than those written by SC6 and SC21 need to have conformity clauses suitable for profiling. In general, SCs need to be aware of the fact their base standards may be referenced by a profile.

Action: SGFS to forward a liaison statement to the appropriate group on this issue.
Status: Open

Issue 8: Trivial Conformance.

In the process of converting a base standard from a large monolitic one to one suitable for profiling, you may encounter situations were trivial conformance is possible but was not desired originally.

Action: Review the examples on "Partitioning of Base Standards" in Annex C.
Status: Open

Issue 9: Indirect Reference of Profiles.

When referencing profiles from other profiles problems may occur (when profile-1 references profile-2 and profile-3, and profiles 2 and 3 both reference the same base standards with conflicting options ....).

Action: Review the examples on "Explicit Undefined Functionality" in Annex D.
Status: Open

Issue 10: The Usage of the Word 'Framework'.

Originally the word framework was used in the general sense. Currently the word has a more specific meaning.
Proposal 1; replace 'framework' by 'general principles'.
Proposal 2: explain exactly the meaning of the word.
Framework is used in 3 different places in the document with 2 different meanings.
Proposal 3: retain framework in the title (it is referenced elsewhere), remove the word from clause 7.
The NWI (Ref. JTC1 N1534) to create a model and framework for AEP may be relevant, as well as US contribution TB-6.

Action: Incorporate in TR10000-1.3, and review text.
Status: Closed.

Issue 11: JTC1 Resolution 18 - B) : Interaction with User Groups.

How should this be done? Is it needed now? Would it confuse the process? Are users not yet represented in SGFS? Is this a task for the National Member bodies? Or the S-liaisons?
Either in a explanatory report or in the profile it should be documented how the user requirements are taken into account, and how this process was implemented.

Action: SGFS is asked to member bodies and other organizations to have a process to represent user requirements and to translate them into technical requirements.
Status: Closed.

Issue 12: The Structure of TR10000.

The current proposal (in line with N402) is to have a part 1 with the generalized concepts of profiling, and subsequent parts that are application area specific:

Part 1:General + OSE scope
Part 2:OSI - OSI specifics + OSI taxonomy
Part 3:AEP - AEP specifics + AEP taxonomy (or taxonomies)

Part x:CIM - CIM specifics + CIM taxonomy
Part y:Banking - ....

Action: Review
Status: Closed

Issue 13: Defintion of OSI Profile.

It is recognized that more types of profiles can be identified than the only OSI type of profile in the current version of TR10000-1. It may be useful (and natural) to adapt the definition of OSI profile to reflect this new situation.

Action: New text to be introduced in TR10000-1.3.
Status: Open

Examples on "Grey Areas of New Functionality" (Donn Terry).
Preliminary (incomplete) version.

The current TR10000 explicitly disallows introducing "new functionality" in a profile.

There are two problems with this concept, and there needs to be a means to address each of them:

1. There is functionality that is not naturally within scope of any one of the referenced base standards (or even one which might hypothetically exist) because it deals with concepts that exist only when the universe of discourse includes several standards.

2. There is functionality that is specification of such things as new symbolic constants or other parameters which apply specifically when the profile is in use, but not when the same base standards happen be on a common system.

Some examples of the first class are:

Requirements for interoperability of files between languages. A profile which calls out more than one language must address the issue of how the languages might interoperate (or at least say that they do not). However, it is not necessary to specify very much about how the interoperation is accomplished. Rather it is simply necessary to specify the OBSERVABLE behaviour __from the point of view of each language__. Going any further would specify the implementation, which should be irrelevant.

A statement of this kind is not even meaningful in the language standards of today because in general they ignore the existence of other languages (for very good reasons). Extending the scope of the language standard to talk about other languages leads to the issue of "which other languages?" (of the 20 or so standardized in SC22, plus many more standardized in specific application domains, some of which SC22 may not even be aware of). It also could easily lead to having pair-wise interfaces between each pair, which is both an engineering nightmare and unnecessary.

A statement requiring such interoperability might be considered "new functionality".

Window systems have the characteristic that at times an application needs to be notified of a change of state of a window under its control that is not due to any action taken by the application. (Specifically, when the window is uncovered by the window manager.)

Many operating systems do not provide a mechanism for asynchronously notifying the application of such an event, but when present that mechanism is preferred.

The standard for the window system cannot assume that such a mechanism is available, either in its language-independent form or in the language binding. It is only when the window system is described together with an OS that provides such a mechanism that it is meaningful to talk about such a mechanism.

Is it appropriate to describe the relationship between the OS and the windows system in the profile (in such a way that semantics are provided), when the profile is the first document where the universe of discourse includes both functionalities.

(This is similar to the "files" problem above, but involves no data interchange!)

[This does bring to mind the possibility that there may need to be a "OS binding", in the same sense as a language binding, when a set of functionality is attached to a specific OS. If this is the case there will have to be significant work figuring out exactly what that means.]

Some examples of the second class are:

Identification: the negotiation for what features are present and the like occurs at compile time for application portability, and the application needs means of inquiring which options and which profiles are present. Mechanisms for such inquiry are provided in the standard, but there needs to be a means to add the appropriate constants necessary to request and interpret such information.

Is this "new functionality"?

ISO 9945-1 provides a "signal" mechanism, where a process can be informed of an outside event. One of the possible signals deals with changes in state of other related processes. It is possible to control exactly which processes can create signals to the "master" process from within the master process by passing a flag to the signal handling mechanism.

This flag is only meaningful for the one signal type associated with other processes, but it is possible (syntactically) to provide it with any signal.

Currently the standard is silent as to what happens if the flag is passed to any other but that one signal (presuming that a correct program would never do that).

Is it new functionality to specify that providing that flag for other signals is ignored?

Is it new functionality to specify that it generates an error?

Is giving the flag a meaning for some other signal new functionality?

Is giving a new flag constant (and a meaning) new functionality?

[Probably the latter are, but exactly where is the line?]

Examples on "Conflicting Options" (Donn Terry).
Preliminary (incomplete) version.

When more than one profile specifies an option in a standard, it is possible that they may do so in such a way as to make irreconcilable requirements on (potentially) the same system.

Some mechanism is needed to address this possibility.

Some examples:

POSIX provides a means to read a directory in a file system. There are (at least) two distinct ways that could be implemented, and the standard explicitly acknowledges both of them, giving explicit guidance as to the consequences (or lack there of) of certain choices by the implementor. This was specifically intended to give freedom to the implementor in his mechanism of implementation, and it is expected that any well-written application would be immune to the difference.

It is conceivable that a profile may wish to specify exactly one of these choices, either to enhance importability of applications that (improperly) assume that a specific choice was made, or for some sound technical reason.

For most purposes the decision of the implementor as to which of the mechanisms he uses is permanently single valued (due to the nature of the technology), and it is impractical to require both in a profile.

Two profiles which conflict on this issue will not be able to coexist on the same system.

POSIX specifies that either of two error behaviours are acceptable when a read or write operation is interrupted due to an external event.

In one instance it returns an indication of an error, and all the data is lost.

In another alternative, it returns as much data as was read (or the count of data actually written).

It is conceivable that a profile with a goal of compatibility with existing systems might choose either value (depending on the systems of interest), or that one or the other values may be required based on engineering issues. (Error might be appropriate when the exact amount of data transferred cannot be determined, which does occur.)

A profile which requires one of these behaviours cannot exist comfortably with one that requires the other. In this case the base standard could be extended to require inquiry and/or specification of the behaviour, but to do so would require changes to all applications which conformed to either profile (when the base standard was extended). (Specifying conflicting default values for the option would have the same effect.)

There are many other situations where a choice must be made by the vendor which may limit the set of profiles that the implementation can conform to, and which may in fact be inappropriate for specification by profile writers.

(Note: there are clearly examples of things that should not be specified in profiles, things that should (which the base standard should make into clear options), and ones for which the engineering decisions are not initially clearcut.)

Examples on "Partitioning of Base Standards" (Donn Terry).
Preliminary (incomplete) version.

This is an example of a situation encountered when a standard is divided into options to support profiling.

The current POSIX standard (9945-1) has a very small number of identified options, with the bulk of the document mandatory.

The "realtime" groups are requesting the ability to refer to some subset of this large functionality.

To do this, these groups wish to partition (in a mathematical sense of "partition") the functionality into a number of fairly small sets, so they can be selected for profiling individually. The cut lines in some cases go through a single interface (e.g. read/write without pipes).

However, any one of these partitions, taken alone, is of negligible interest to the user of 9945-1. In general, it takes a significant fraction of the total capability of 9945-1 to assure application portability within a single application domain. However, depending on the application domains in question, the portions required can be (nearly) disjoint.

This then leads to a question: is there a way of having 9945-1 (or any standard) "profile" itself so that it has the following characteristics:

1. The subset necessary to claim conformance to the standard is a significant fraction of the total (and represents the usual way in which the standard, alone, is used).

2. Profiles may use other subsets (as defined by options), and use rather small fractions of it, but not imply conformance to the base standard by that usage alone.

3. Conformance to some (uninteresting) lowest common denominator does not imply conformance to the base standard.

It is also desirable to avoid divergence of the specification, and to avoid creating large numbers of subsets. Thus the concept of copying the standard and customizing it for a particular profile seems very unattractive.

An example:

Real time imbedded systems have a bona-fide need for systems which are

> significantly less than 9945-1 for their specialized needs. Insisting that they provide unused functionality is impractical. (They have significant memory constraints; many successful commercial RT operating systems are <4K bytes.)
>
> There is a need for a minimal profile which includes a subset of the POSIX read/write and some process management interfaces.
>
> There is also a need for a minimal profile which includes the same subset of read/write, and NO process management, but including the "threads" capability. (Threads represent multitasking in a single address space, processes imply different address spaces and other separated resources).
>
> This leaves a situation where the minimal intersection is read/write, which by itself is so minimal as to be useless. No practical profile would ever be so minimal as to use just the minimal core, but the intersection is minimal.

The goal is to indicate that the minimal possible profile is not just the lowest common denominator which arises from the partitioning.

In addition, 9945-1 has come to represent a significant level of portability. By introducing subsets that can weaken the user expectation of what 9945-1 provides to them, the credibility of that standard is weakened.

Other large monolithic base standards would appear to have a similar problem, and this could affect the credibility of the whole standards process.

There is an implied guarantee of application portability created by an API standard. This guarantee is what gives the standard its value. From the perspective of a general user (not one using one of the specialized profiles), the minimal level is approximately equal to the current standard. It is desirable to keep this implied guarantee while still allowing "subsets" for other well-considered profiles.

A solution to this problem is desired as part of the extensions of TR 10000 to OSE.

Currently, clause 6.3.1 (c) would appear to prohibit any solution along these lines.

One solution (assuming it were permitted by TR 10000) might be a conformance clause might be along the lines of the following:

> An implementation claiming conformance to this standard shall provide all the following options:
>
> > LIST

An approved profile may refer to a smaller number of these options.

Note: an implementation which provides which provides less than the minimum defined above may be considered as conforming with such a profile, but shall not be considered as conforming with this standard.

Examples on "Explicit Undefined Functionality in Base Standards" (Donn Terry). Preliminary (incomplete) version.

Many SC22 (and other) standards leave certain areas that naturally fall within their scope of interest in some way "undefined", because it is not believed to matter to application portability. These areas are either implicitly identified by silence, or are explicitly identified by the use of such words as "if", "may", "unspecified" or "undefined", each of which carries a precise and subtly different meaning.

It is possible that due either to historical precedent or due to lack of perfect foresight on the part of base standards writers that some of these instances may be a place where a profile writer wishes to specify a behaviour.

Some of these behaviours fall into the "new functionality" arena, and thus should be considered in that topic. However, some do not deal with new functionality at all, but nevertheless do not either deal with options that were intended by the base standards developers.

Some examples:

POSIX provides a means to read a directory in a file system. There are (at least) two distinct ways that could be implemented, and the standard explicitly acknowledges both of them, giving explicit guidance as to the consequences (or lack there of) of certain choices by the implementor. This was specifically intended to give freedom to the implementor in his mechanism of implementation, and it is expected that any well-written application would be immune to the difference.

It is conceivable that a profile may wish to specify exactly one of these choices, either to enhance importability of applications that (improperly) assume that a specific choice was made, or for some sound technical reason.

For most purposes the decision of the implementor as to which of the mechanisms he uses is permanently single valued (due to the nature of the technology), and it is impractical to require both in a profile.

Is it appropriate for a profile to require one or the other behaviours? An ISP?

POSIX provides the ability to lock regions of a file, either for reading or read/write. It is possible also to interrogate for the first lock that is blocking an attempt to create a new lock.

It is reasonable to ask "is the lock read?" or "is the lock read/write?", and this is provided. However, in examining the matrix of possible inputs (as specified by POSIX), it is legal to attempt this operation with a request for a type of "unlocked".

This then asks the question "what does this mean?"

If a profile writer were to find a use for this, could a specific meaning be specified?

The answer to this may vary with the selected meaning. Which of the following are new functionality?

+ The operation is explicitly defined as being an error.

+ If the operation is specified as being equivalent to an existing operation, is that new functionality? (For this example, it might be specified as exactly t he same as asking for a read lock.)

+ If the operation is something new, but is fully describable in terms of the existing functionality. (For this example, it might be specified as returning the first byte that is not locked (that is, the end of the first locked region, if I t is at the beginning of the region of interest).)

+ If the operation is "just a little" beyond this. (For this example, it might return the number of unlocked bytes instead of the location.)

(To my view, the third and fourth should be disallowed, but the first two are not clear to me as to what should be done.