|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Title\*:** | Why TPlan is a good basis for requirements and objectives specification | | |
|  |  | | |
| from **Source**\*: | Cadzow Communications | | |
| Contact: | Scott Cadzow | | |
|  |  | | |
| input for **Committee**\***:** | MTS | | |
|  |  | | |
| Contribution **For\*:** | Decision |  |  |
|  | Discussion | **X** |  |
|  | Information |  |  |
|  |  | | |
| Submission date**\***: | 2012-01-18 | | |
|  |  | | |
| Meeting & Allocation: | **MTS#55** - Session 1: Opening Formalities | | |
| Relevant WI(s), or deliverable(s): |  | | |
|  | | | |

**ABSTRACT:***A short outline of why TPlan is a good basis for requirements and objectives specification*

The starting assertion of why TPlan is underutilised is that ***a test purpose is an objective*** – it does not specify the test (that is done in the test cases) but sets the framework for a test. The relation of TP to TC is similar to that of objective to requirement in core standards. So from the starting assertion we can ask if we can use TPLan as a notation for the expression of any objective and not just test objectives.

ES 202 553 has a small set of structural requirements in its notation that if extended can maintain the structure of the TP as:

*Pre-conditions -- optional initial conditions*

*TP behaviour description -- comprising sequences of:*

*Stimuli and Responses*

The current restriction in TPlan is that the behaviour description is restricted to the statement "ensure that". Whilst this is grammatically correct in English and can apply to most objective statements. If we look to TR 187 011 which is a guide to the writing of security objectives and requirements we can examine how to transpose the guidance given there to an extension of TPlan. A well-formed statement of an objective should contain a number of key words that will indicate the class of functional capabilities likely to assist in getting to a detail requirement. The following example is a clear and concise objective.

EXAMPLE:   
An NGN should be able to ***restrict access*** to its services so that they are only available to ***validated known users***.

If we use the TPlan structure of:

**ensure that** {

**when** { *...* } *-- stimuli described from the viewpoint of the IUT*.

**then** { *...* } *-- IUT responses and other behaviour*

}

We could map TPlan to the example from TR 87 011 by re-thinking the statement in the overall context of precondition => stimulus => response, remembering that we are not attempting to define how the objective is fulfilled (that is the purpose of the next step of detailed requirements).

**ensure that {**

**when {service is accessed}**

**then {the accessing user is known and validated}**

**}**

The intent of updating TPlan is to extend its applicability to environments other than testing. In the first instance it is proposed to address the security domain using TR 187 011 as an example set. This will allow TPlan to be extended to the entire ISO/IEC 15408 functional capability set and allow users of TPlan to take their system specific key words and apply them to the library of keywords and structures of ISO/IEC 15408.

We may also look to extend TPlan to have alternatives to the "ensure that" statement to allow use of TPlan as a tool in making and validating security assertions but this is for further study as we look to extend TPlan away from a test purpose notation to an objective notation and supporting language.

In addition as we move from notation to language we can look to enforce the syntax and semantics of objectives statement, and ideally ensure that they are correctly translated to functional and detail requirements – in the same way we could reinforce the link between TP and TC.