



➔ **For a more accurate definition of semiformal language in the CC**
11th ICCG, Antalya, 21-23 September 2010

Discussion for a common interpretation of what semiformal language means for CC

- ▶ Context
- ▶ Objectives
- ▶ Proposal
- ▶ Conclusion



- ▶ Author: **Franck ELLERO**
 - Embedded software developer for 10 years
 - Common Criteria consultant since 2001
 - CC Evaluator for 4 years



Thales ITSEF:

- ➔ HW & embedded SW ITSEF
- ➔ Under ANSSI agreement

▶ **Speaker: Jean-Yves BERNARD**

- Evaluator for 9 years
- Thales ITSEF technical manager
- Lead auditor ISO/IEC 27001:2005 (certified by LSTI)
- Risk manager ISO/IEC 27005:2008 (certified by LSTI)

- ▶ Semiformal language :
 - ▶ Compromise
 - between informal language (expressed in natural language)
 - and formal language (based on well-established mathematical concepts)
 - ▶ CC definition = restricted syntax language with defined semantics
 - ▶ Semiformal presentation = standardized format with well-defined syntax that reduces ambiguity
 - Structured description, written with a standard presentation template

→ CC definition is general and subjective

- ▶ What is a structured description?
 - ▶ Shall it be a simple structured presentation?
 - e.g. a specification description using key words: “Purpose, Input, Action, Output”, and including a glossary of these terms
 - ▶ Shall it be a stricter language where all actions in the specification have to be described using a well-defined syntax, with either restricted language or diagrams?
 - e.g. a specification described using UML

=> Which level of structured description is acceptable to be considered as semiformal?

- ▶ Objective 1:
 - ▶ Evaluations including descriptions in semiformal language shall be comparable, i.e. evaluated with a similar level of rigor
 - Need of a common interpretation from CC community for the notion of “semiformal description” among all actors of a CC evaluation
- ▶ Objective 2:
 - ▶ Requiring semiformal descriptions (e.g. EAL5 vs EAL4) is supposed to provide a meaningful increase in assurance
 - Higher assurance level evaluations requires higher degree of formalism in order to gain better confidence in security functionality correctness of evaluated products

- ▶ Common interpretation of term “semiformal”
 - ▶ Level of rigor required in terms of semiformalism shall not be too minimalist otherwise the benefit provided by a semiformal description against informal one will almost nil
 - ▶ Shall not only focus on the form (e.g. standard presentation template, diagrams)
 - Definition shall prevent from “cosmetic” representation
 - It is a fact that a description can be in a structured form (e.g. UML-like form), and yet have its content just as poor as any other informal description
 - ▶ Shall focus on the substance
 - To ensure that the TSF, and hence the SFRs, is more accurately specified and instantiated from one abstract level of semiformal description to a lower one

- ▶ Gaining better confidence in the correctness of the TSF is to obtain better assurance that:
 - ▶ The SFRs are correctly instantiated through all the various levels of abstraction, reducing ambiguity that may occur in informal presentations
 - More accurate specification and rigorous refinement of semiformal description from one abstract level to the lower one, so that mapping between abstract levels becomes obvious
 - ▶ Implementation of the SFRs is correctly tested
 - Unambiguous semiformal description leads to more accurate functional testing, so that evidence of test coverage becomes obvious

- ▶ Semiformal language shall provide obvious evidence that instantiation of the SFRs is correct and test coverage is accurate
 - ▶ Semiformal specification in the FSP should be such that it enables to set a direct, accurate and unambiguous mapping between
 - The SFRs and the functional elements, on the one hand
 - The SFRs and the functional test coverage, on the other hand

- ▶ For instance:
 - ▶ FCS_CKM.4.1: “The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [...] that meets the following: [...]”
 - In the FSP, key destruction may be spread into many actions invoked by TSFIs (and for some, depending on the design)
 - In informal description, this could lead not accurate instantiation or else incomplete
 - Another consequence is also less assurance in functional test coverage completeness

▶ Rule 1:

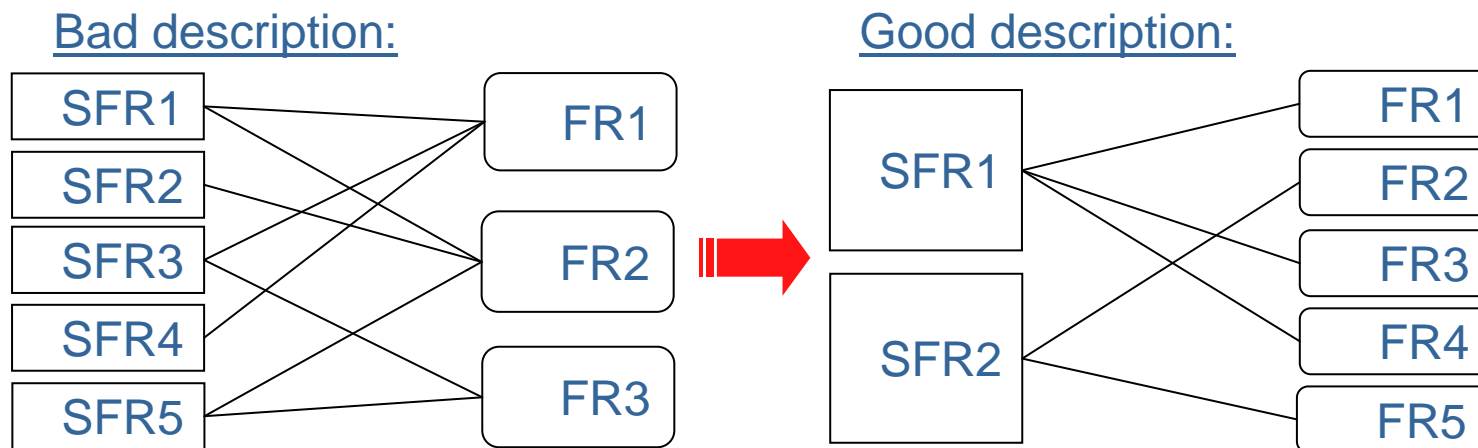
- ▶ Description uniquely identifies each **specification element** (i.e. unitary piece of description) related to the TSF (e.g. actions invoked by the TSFI)
 - This is important for traceability both to the design specification and to the functional testing
 - Very often, functional requirements are identified but those related to security are not specifically identified. It results that the mapping between the SFRs and the functional specification is not so obvious, making traceability of the SFRs into the design specification quite difficult

▶ Rule 2:

- ▶ The specification shall be written such that each “specification element” can be functionally tested
 - This rule should be required even with informal description
 - Semiformal language shall provide means to ensure that this rule is enforced

▶ Rule 3 (1/2):

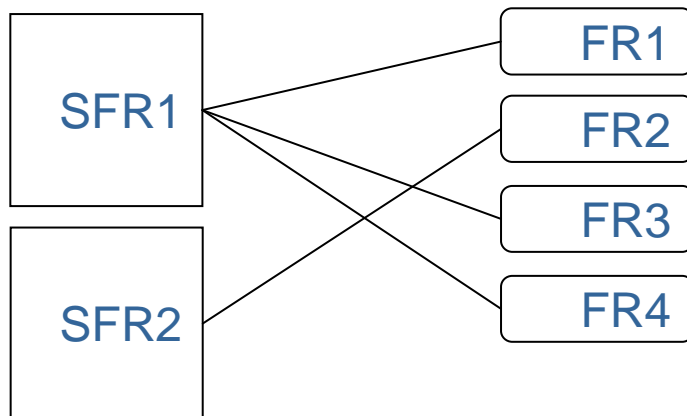
- ▶ The specification elements shall be commensurate such that each of them does not refer to several SFRs (or link is obvious)
 - When security requirements are not specifically included into the specification, each SFR is spread into the functional specification. The result is that complete and accurate mapping of the SFRs cannot be determined so easily



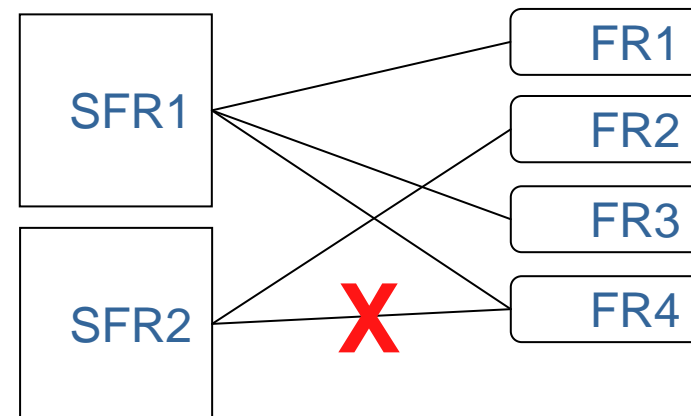
▶ Rule 3 (2/2):

- ▶ The specification elements shall be commensurate such that each of them does not refer to several SFRs (or link is obvious)

Example 1:



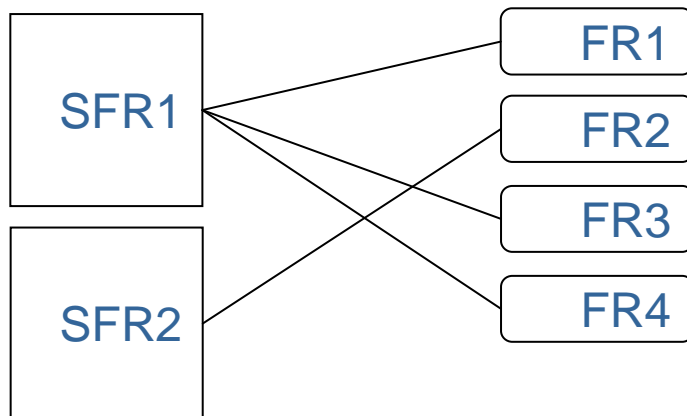
Example 2:



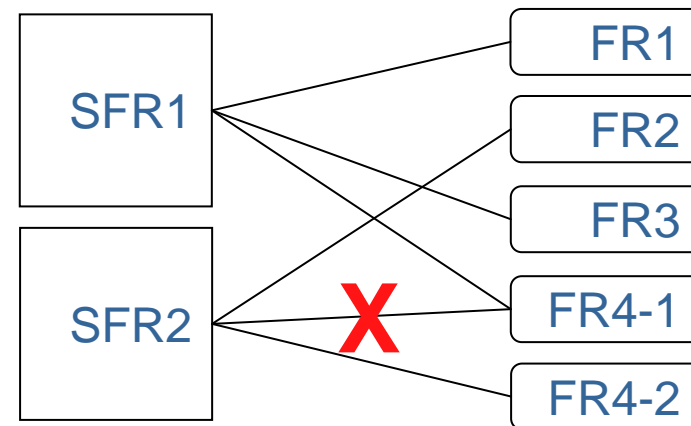
▶ Rule 3 (2/2):

- ▶ The specification elements shall be commensurate such that each of them does not refer to several SFRs (or link is obvious)

Example 1:



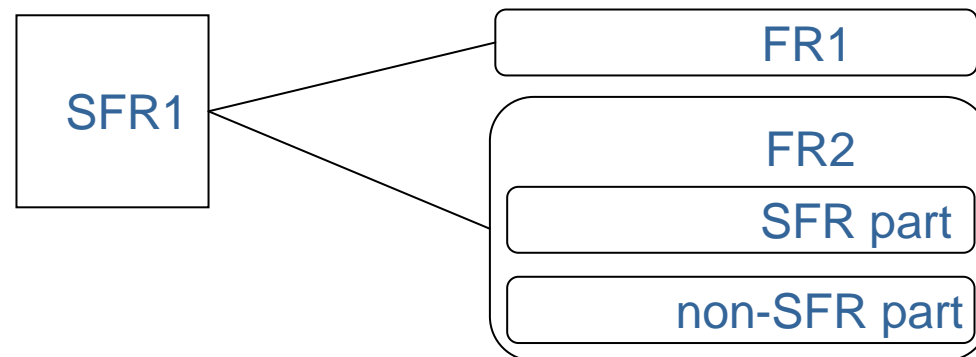
Example 2:



▶ Rule 4:

- ▶ The specification elements shall be commensurate such that each of them does not refer both to SFR and non-SFR parts either
 - Tests cannot be exhaustive. When specification elements include both SFRs and non-SFRs parts, with a poor mapping for test coverage, even if all TSFIs are tested, it is difficult to check whether there is a test for each SFR part or not

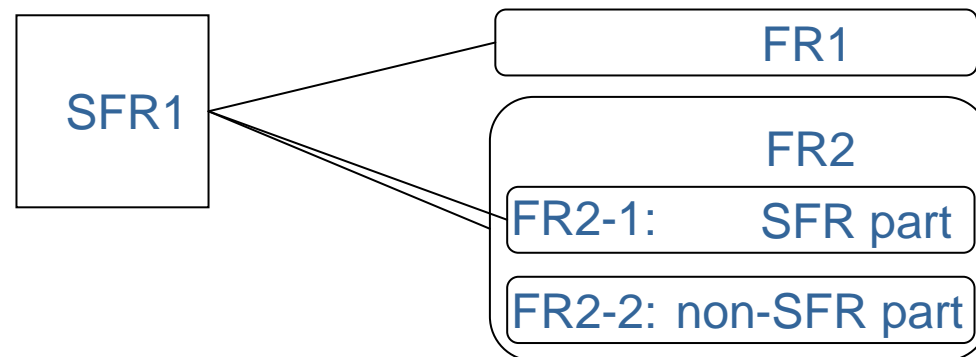
Example:



▶ Rule 4:

- ▶ The specification elements shall be commensurate such that each of them does not refer both to SFR and non-SFR parts either
 - Tests cannot be exhaustive. When specification elements include both SFRs and non-SFRs parts, with a poor mapping for test coverage, even if all TSFIs are tested, it is difficult to check whether there is a test for each SFR part or not

Example:



▶ Rule 5:

- ▶ Semiformal specification shall provide means to be able to easily check internal consistency of these specification elements (and hence the TSFIs)
 - The semiformal language shall include specific syntax and rules to ensure internal consistency of the TSFIs.
 - This could be otherwise met with a development tool (e.g. IDE) performing cross-check of the TSFIs

▶ Rule 6:

- ▶ Semiformal design specification in the TDS shall provide a means to easily and unambiguously show the connections between the design elements (i.e. subsystems or modules)
 - e.g. hyperlinks

▶ Rule 7:

- ▶ Semiformal design specification in the TDS shall enable to set a direct, accurate and unambiguous mapping between the identified functional elements in the FSP and the design elements in the TDS
 - It shall be easy to show the complete instantiation of the SFRs down to the design elements

▶ Rule 8:

- ▶ At lowest level of description (e.g. pseudo-code), semiformal design specification in the TDS shall enable to minimize ambiguousness and risk of misinterpretation
 - Example for multi-project source codes:
 - In case of source code subject to numerous and nested conditional compilations, readability may become tough and without any help could lead to misinterpretation
 - Example for object-oriented languages:
 - Polymorphism (i.e. different codes for a single method name) may lead to misinterpretation when reading comments in source code without a smart editor (e.g. IDE tools)

- ▶ CC market trend goes to ever higher level evaluations
 - ▶ High level CC evaluations (EAL5 to 7) require for more and more formalism in order to gain better confidence in security functionality correctness
 - Description in semiformal language shall be such that evidence of correct instantiation of the SFRs and of test coverage becomes obvious, i.e. easy to check
 - Double advantage: evaluator's checking is more accurate and more efficient (easier check => quicker check)
 - ▶ Consensus on a clear definition of term "semiformal" should be reached by CC community
 - Necessary to keep comparability for high level certifications (as a minimum under each national schemes)



Any questions?

Thank you for your attention...