Applus LGAI

Automated tools for supporting CC design evidence

September 2008

Applus

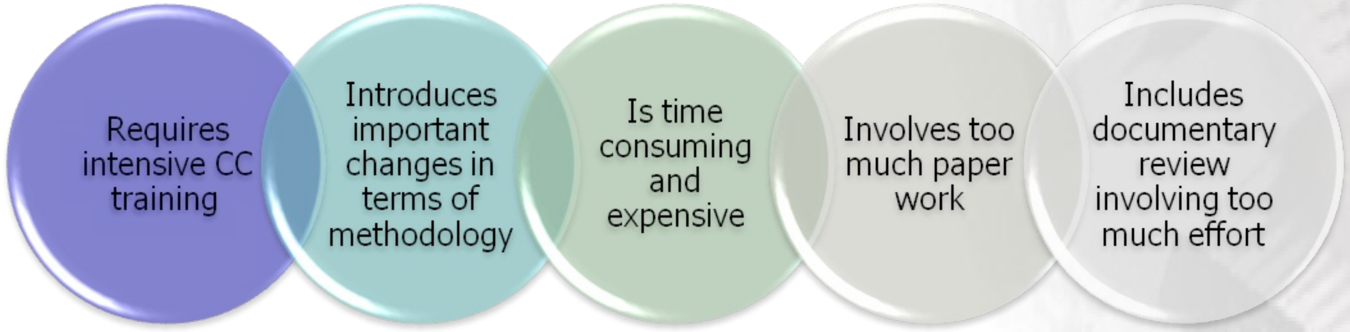
Outline

Ismael Kane
Common Criteria Senior Evaluator
ikane@appluscorp.com

01_ Introduction

The goal of Common Criteria is to get better products in terms of security.

Some developers are reluctant to use Common Criteria because the process:


Requires intensive CC training
Introduces important changes in terms of methodology
Is time consuming and expensive
Involves too much paper work
Includes documentary review involving too much effort

National schemes and CC working groups are working to solve these issues, creating new mandatory documents and procedures to improve methodology.

The purpose of this presentation is to describe procedures and techniques offered by the Lab, in order to reduce and automate some developer’s activities and consequently, to be more proactive during CC development stage.

Therefore, by offering solutions in advance, the Lab will be able to reduce some typical developer's mistakes and particularly increase the security of the final product.

02_ Automated tools adapted to developers

Based on claimed assurance package, there are different requirements to fulfill. Consequently, our solutions take them into account and adapt to:

Automated tools to create:

- extensible and customizable framework
- documentation without incorrect or inaccurate references
- a common framework for all documentary sources

Automated tools to verify:

- documents based on standard rules
- that the framework is absolutely extensible and customizable
- efficiency proof
- the identification of the developer's mistake pattern

02_1 Applicable to Security Target

Applus LGAI

Identification of typical problems in Security Target

Invalid reference of documents	• Bad reference to evidence (version, date,..)
Problems with SFR specification	• Bad or incorrect SFR representation
SFR Conflicts	• Specification of incompatible SFR: • Not inclusion of dependencies • Bad justification of non-included
Conflicts between overview, description	• Contradiction, incompleteness or non-accuracy between different parts of ST. Undefined Terms
Mappings and rationales	• Non coverage • Lack of traceability

Applus

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Invalid Reference**

Based on different technologies, the Lab offers plug-ins and scripts which can be included in any Edition tool, and which can avoid “mis-matched” references.

Example: Usin LaTeX package svninfo, every reference will be fully identified:

```
\documentclass{article}
\usepackage{svninfo}
\begin{document}
\svnInfo $Id$
\title{\svnInfoFile}
\author{\Author}
\date{\svnInfoDate}
\Version{\svnInfoRevision}
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Invalid Reference**

Example: Using Word variable to link SUBVERSION

```
Private Sub DocumentVariables()
        Dim doc As Document
        Dim objDoc, objFile, objFSO, objWord, strFile, strSVN
        Set oShell = CreateObject ("Scripting.shell")
        Set strFile = doc.Variables.get(WdFileName)
        Set strSVN = oshell.run "svn info "&strFile&
        doc.Variables.Add(CStr(VERSIONSVN), strSVN)
End Sub
```

Example: Integration of visualbasic with Visual source-safe

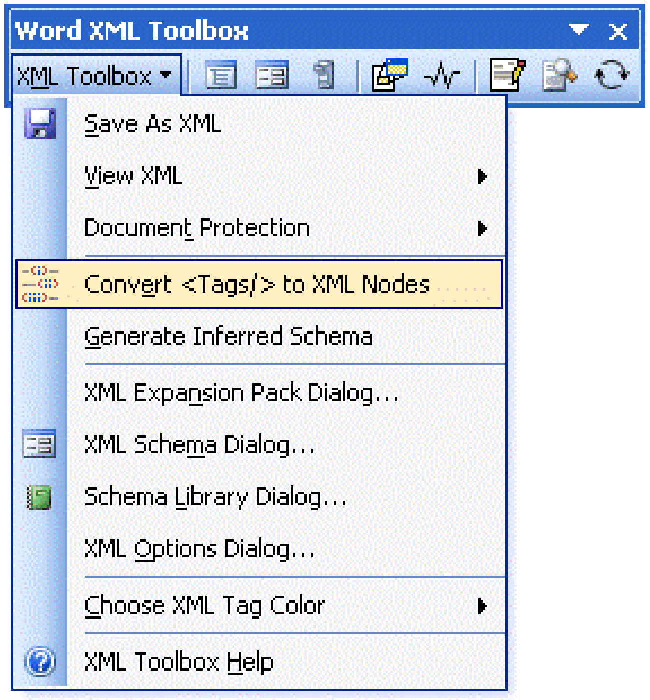
```
Dim version, etiqueta as string
Set oVSSDatabase = WScript.CreateObject("SourceSafe")
oVSSDatabase.Open ("C:\TOOLS\VStudio\Common\VSS\srcsafe.ini")
Set oVSSItem = oVSSDatabase.VSSItem("$/Version/fileTest")
Set oVSSItemVersion = oVSSItem.get_Version(0)
Set version =  oVSSItemVersion.VersionNumber
Set etiqueta =  oVSSItemVersion.Label
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: Uncorrected **representation of CC terms**

Integration of cc.xml files in any Edition tool:

Example: Xml toolbox in Microsoft Office



Example: Combining xml with Xinclude

```
<?xml version=”1.0”>
<requisites>
        <requisite id=”FCS_COP.1>
                <content>
                        <xi:include
href=”cc.xml” xpointer=”xpointer(//
f-component[@id='FCS_COP.1'])”>
                </content>
        </requisite>
</requisites>
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Conflict between SFR**

Conflicts will be basically avoided with generic rules (created by evaluators).

Rules based on real conflicts, there are some CC Part 2 requirements that are absolutely contradictory. Some examples are:

- FDP_UNL unlinkability with FIA_USB user-subject binding
- FDP_ROL rollback with FPT_RIP residual information protection
- FIA_UID user identification with FRP_ANO anonymity
- » FTA_TAH TOE Access history with FRP_ANO anonymity

Moreover, as everybody knows, there are some CC Part 2 requirements that have a mandatory dependency and, to avoid it, developers must provide appropriate justification.

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Conflicts between SFR**

Example: Detecting basic conflicts with XPath

```
<?xml version="1.0"?>
<requisites>
    <requisite id='FDP_ROL.1' title='Basic Rollback'/>
    <requisite id='FDP_RIP.1' title='Subset residual information protection'/>
</requisites>

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform
">
    <xsl:output method='text'/>
    <xsl:template match='/sfrTSS'>
    <!-- Potential Conflicts? →
           <xsl:if test='count(/sfrTSS/requisites/requisite[@id='FDP_ROL.1']) != 0
and
                  count(/sfrTSS/requisites/requisite[@id='FDP_RIP.1']) != 0 '>Error:
                  Potential Conflict between  FDP_ROL.1 and FDP_RIP.1
           </xsl:if>
    </xsl:template>
</xsl:stylesheet>
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Unknown subject**

Generating a tool which extracts any subject from ST overview, description, Security requirements and TSS, and which verifies subjects regarding terminology or reference.

Some features of extraction tools are as follows:

- Integration with ST reference chapter
- Integration with CC reference chapter
- Integration with Wiki or public repositories
- Reporting of unknown subject

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Cross-Section conflicts**

By using leveling tags, it is possible to index different parts of security target. That is why the Lab uses three relationship levels:

- Enforcing
- Supporting
- Non-interfering

This solution aims to provide benefits in work which tries to analyze consistence between different parts of ST.

Furthermore, it gives a formal representation of consistence justification.

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Cross-Section conflicts**

Example: Specification of DTD and xml's Instantation

```
<!--Definition of Statements in ST-->
<!DOCTYPE dependency [
<!ELEMENT dependency (id,statement,depen+)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT statement (#PCDATA)>
<!ELEMENT depen (type,related)>
<!ATTLIST type (enforcing|supporting|non-interfering) "enforcing" xml:lang CDATA 'en'
#REQUIRED>
<!ELEMENT related (#PCDATA)> ]>

 <?xml version="1.0"?>
<dependency>
        <id>13</id>
        <statement> TOE will protect confidentiality of secrets with symmetric encryption</
statement>
        <depen>
                <type>enforcing</type>
                <related>14</related>
        </depen>
        <depen>
                <type>supporting</type>
                <related>16</related>
        </depen>
</dependency>
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Mapping Coverage**

Verification shall be based on scripting, and it:

- Can detect incorrect traceability between SFR and TSS
- Can check if every SFR has at least one TSF
- Can check that rationale for missing dependency exists.

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Mapping Coverage**

Example: Incorrect traceability between SFR and TSS

```
<?xml version="1.0" ?>
<sfrTSS>
        <tssS>
                <tss id='AA' title="Authentication"></tss>
                <tss id='BB' title="Identification"></tss>
        </tssS>
        <requisites>
                <requisite id='FIA_UAU' title='User authentication'>
                        <covers id='AA'>justificacio</covers><content></
content>
                </requisite>
                <requisite id='FIA_UAI' title='User identification'>
                        <covers id='BB'>justificacio</covers><content></
content>
                </requisite>
        </requisites>
</sfrTSS>
```

02_1 Applicable to Security Target

Different Solutions to minimizing ST problems: **Mapping Coverage**

Example: Incorrect traceability between SFR and TSS.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method='text'/>
<xsl:template match='/sfrTSS'>
<!-- Are all tss implemented as requirements? →
        <xsl:for-each select='tssS/tss'>
                <xsl:variable name='id' select='@id'/>
                <xsl:if test='count(/sfrTSS/requisites/requisite/covers[@id=$id])=0'>Error: tss

                    "<xsl:value-of select='/sfrTSS/requisites/requisite[@id=$id]/@title'/>" Not
covered!
                </xsl:if>
        </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

02_2 Applicable to evidence of Development Stage

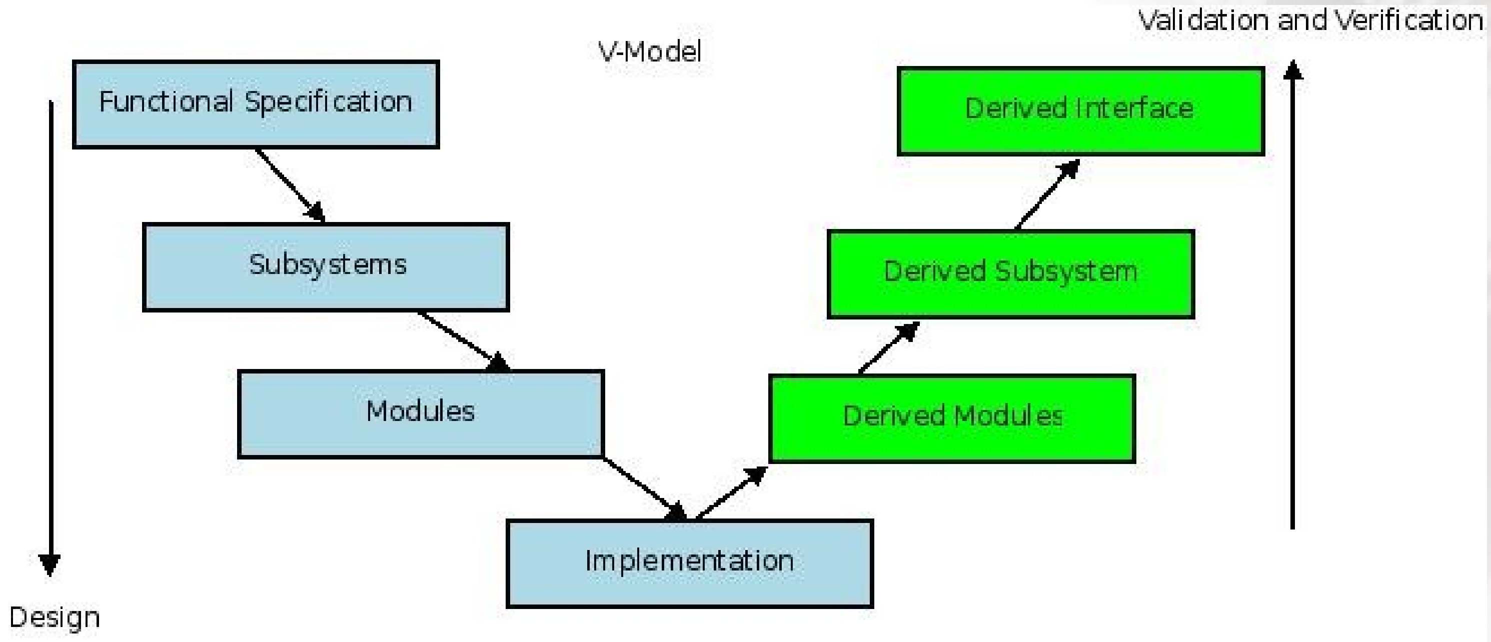
LGAI

Based on Doxygen the Lab offers a tool that can trace, verify and detect the lack of correctness in top-down implementation:

•Extraction of derived modules or subsystems from source code and check against developers' instantiation

• Verification, isolation and security domains by means of cross-references rules

• Visual representation of design and easily linked with other evidence through indexation



02_2 Applicable to evidence of Development Stage

Example: Doxygen generates xml files and it can be parsed and manipulated with xslt.

```
<?xml version="1.0" ?>
<design>
        <tssS>
                <tss id='AA' title="Authentication"></tss>
                <tss id='BB' title="Identification"></tss>
        </tssS>
        <modules>
                <module id='mod1' title="Module1">
                        <covers id='AA' relation=enforcing>justification</covers>
                        <covers id='BB' relation=supporting>justification</covers>
                </module>
                <module id='mod2' title="Module2">
                        <covers id='BB' relation=enforcing>justification</covers>
                </module>
        </modules>
</design>
```

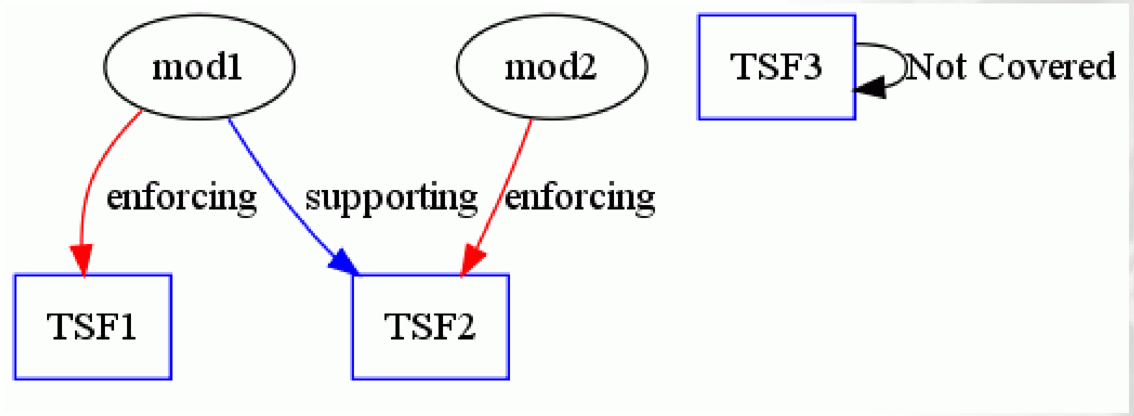
02_2 Applicable to evidence of Development Stage

Example: Doxygen generates xml files and it can be parsed and manipulated with xslt.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method='text'/>
<xsl:template match='/sfrTSS'>
<!-- Are all tss covered by modules? →
        <xsl:for-each select='design/tssS/tss'>
                <xsl:variable name='id' select='@id'/>
                <xsl:if test='count(/design/modules/module/covers[@id=$id])=0'> "<xsl:value-
of  select='/design/tssS/tss[@id=$id]/@title'/>" not covered by modules
                </xsl:if>
        </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

02_2 Applicable to evidence of Development Stage

Example: Graphviz can generate visual linking between modules and TSS.



Consequently, derived modules can be checked and reviewed against developers' instantiation.

An extra benefit is that if the source code is well-defined in terms of tags, the output can be used as a proof of non-bypassing:

Example: Access control variable is managed only for module A and no other modules can modify it. This expression is easily implemented in terms of rules.

02_3 Applicable to Guidance

By using labeling tags it is possible to index every statement of guidance regarding ADV or ST evidence. That is why the lab uses three relationship levels :

- Enforcing
- Supporting
- Non-interfering

Example: Indexation between guidance statement and other CC evidence terms.

```
<!--Definition of Statements in ST-->
<!DOCTYPE dependency [
<!ELEMENT dependency (id,statement,depen+)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT statement (#PCDATA)>
<!ELEMENT depen (type,related)>
<!ATTLIST type (enforcing|supporting|non-interfering) "enforcing" xml:lang CDATA 'en'
#REQUIRED>
<!ELEMENT related (#PCDATA)>
]>
```

02_4 Applicability to Life Cycle support

ALC_DVS is a great candidate to be automated. Thereby, most findings can be achieved and stored logically, for subsequent post-processing.

Each EAL requires a different level of protection. Our automated tool generates scripts that can be adapted to claimed EAL.

Features of capabilities to be checked:

- Confidentiality of evidence in Configuration List:
  - In case of local storage, it is encrypted or protected by local ACL
  - In case of remote storage, it is encrypted or protected by domain ACL
    - Communication is encrypted
    - Secure deletion is used
    - Secure backup is used

02_4 Applicable to Life Cycle support

Features of capabilities to be checked (cont):

Integrity of evidence in Configuration List:

- In case of local storage, integrity checker mechanism is used.
- In case of remote storage, integrity checker mechanism is used.
- Communication is protected against unauthorized modification.
- A secure mechanism is used to protect against unauthorized modification in backup.

The equipment used for handling information:

- is updated
- is isolated
- allows information bridging

02_5 Applicable to Testing

By using labeling tags it is possible to index every statement of testing regarding ADV, ADG and ASE evidence. That is why the lab uses three relationship levels:

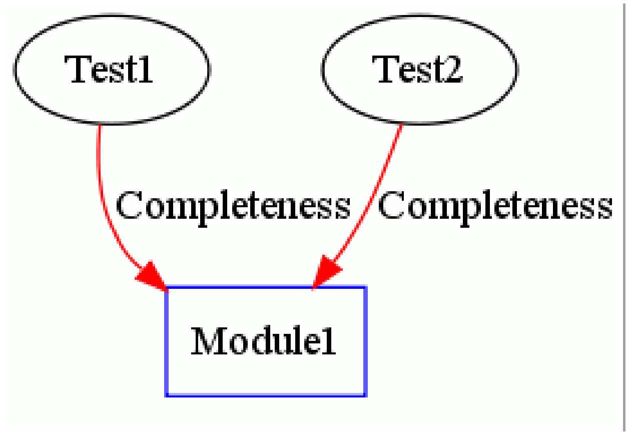
- It is based on three levels of coverage, 1- enforcing, 2- supporting and 3- non-interfering
- It gives a template with all the requirements claimed in CC

```
 <!--Definition of Statements in Test Documentation-->
<!DOCTYPE test [
<!ELEMENT test
(id,initialProcedures,procedures,expecteResults,actualResults,coverage+)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT initialProcedures (#PCDATA)>
<!ELEMENT procedures (#PCDATA)>
<!ELEMENT expecteResults (#PCDATA)>
<!ELEMENT actualResults (#PCDATA)>
<!ELEMENT coverage (sfr,type)>
<!ATTLIST type (enforcing|supporting|non-interfering) "enforcing" #REQUIRED>
 <!ELEMENT sfr (#PCDATA)>
]>
```

02_5 Applicable to Testing

Whit this template it is easy to incorporate new features that can be used for completeness and accuracy at different levels.

- On the basis of information extracted from ADV, implicitly testing will be able to be mapped against:
  - ADV_FSP, test can be mapped to interfaces
  - ADV_TDS at subsystem level
  - ADV_TDS at module level
- And, by using graphviz (open-source tool), get a visual interpretation of test related to ADV



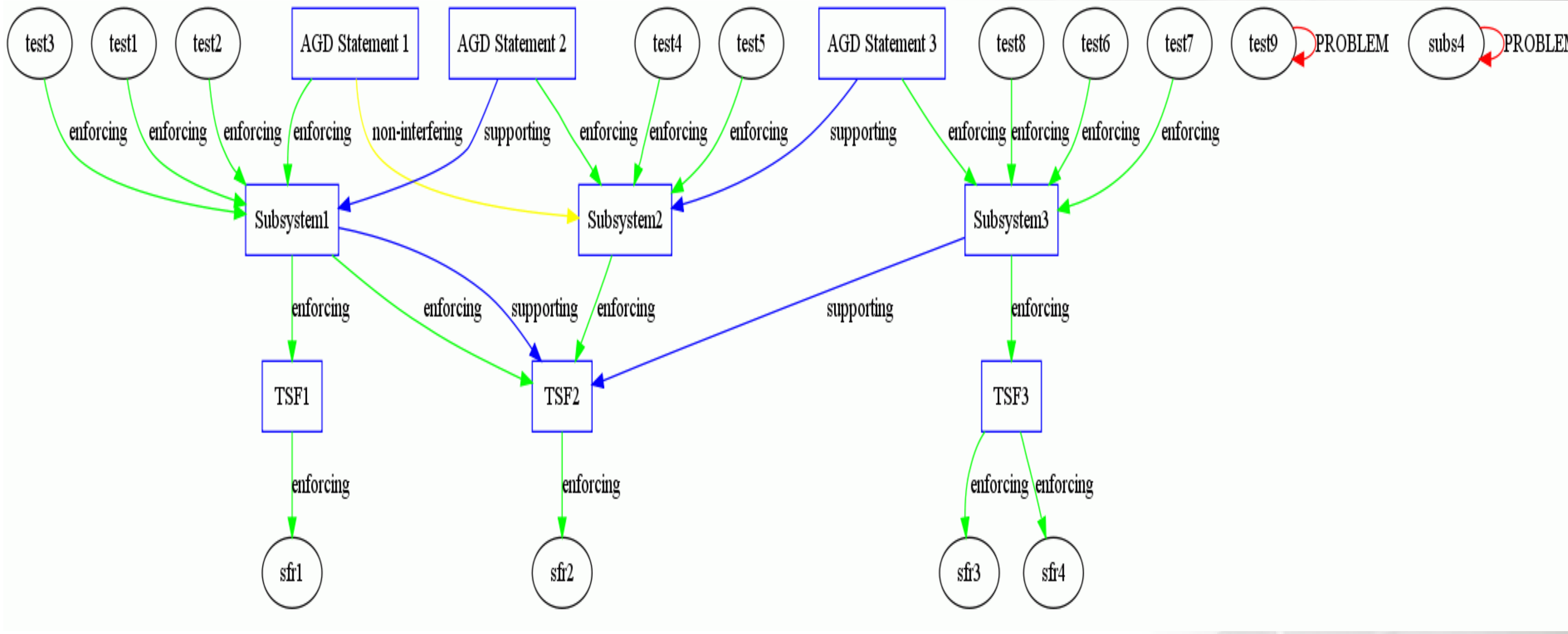
02_5 Applicable to Testing

Based in well-correctness of test's templates and evidences for ATE, it's possible to analysis test against ADV

```
<!--Definition of Statements in coverage-->
<!DOCTYPE coverage [
<!ELEMENT coverage (test, interface, depen)+ >
<!ELEMENT test %test%.%id%>
<!ELEMENT interface %adv%.%fsp%>
<!ELEMENT depen (type,related)>
<!ATTLIST type (enforcing|supporting|non-interfering) "enforcing"
#REQUIRED>
<!ELEMENT related (#PCDATA)>
<!-- For checking correctness of interface coverage-->
<!ELEMENT correctness ()>
<!ELEMENT testcoverage (%test%.%coverage%,%adv%.%fsp%,covertype)>
<!ATTLIST covertype (Correct|nonCorrect)>
]>
```

This technique allows formal language for checking coverage of testing depth and scope

02_6 Challenge



“Uniquely” content validation !!!!

03_ Conclusion

This technique has many advantages for evaluators and schemes because it:

- Offers a common language for coverage and completeness analysis
- Allows traceability between evidence of assurance classes
- Is easily integrated with CC XML structure
- Introduces a common language that can be used in work units
- Allows more effort in AVA analysis

Besides, it also has the following advantages for developers:

- Possible identification of problems in advance
- It is easily adaptable with developers’ tools
- It is time and cost saving
- Gives assurance of well-structured methodologies

Applus⊕
LGAI

Questions ?

Applus⊕

Ismael Kane
Common Criteria Senior Evaluator
ikane@appluscorp.com