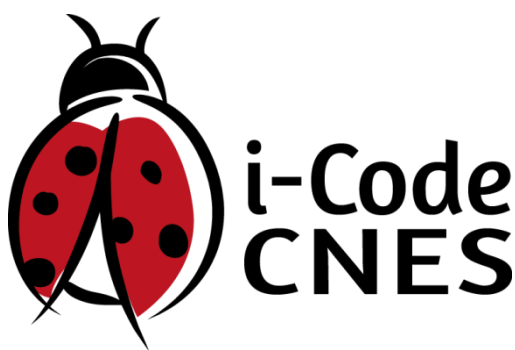


# Manuel Utilisateur

## i-Code CNES

DNO/DA /AQ - 2017.0002478

Version 3.0.1



<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 2/51
--------------------------	---------------------------------------	--

## HISTORIQUE DES MODIFICATIONS

Version	Date	Chapitres modifiés / Raison / Nature de l'évolution
1.0.0	05/12/2014	Initialisation du document
1.1.0	23/01/2015	Prise en compte des modifications des standards Fortran 77, Fortran 90 et des règles communes
2.0.0	03/06/2015	Intégration des règles Shell
2.0.1	03/09/2015	Ajout de l'export xml
2.0.2	21/12/2015	Ajout des marqueurs de violations de métriques et de la fonctionnalité double-clic
2.0.3	14/03/2016	Fusion des modifications des documentations 1.0, 2.0.1, 2.0.2
2.0.4	16/12/2016	Mise à jour suite à la validation
3.0.0	28/08/2017	Mise à jour pour la version 3.0 d'i-Code CNES Ajout d'une rubrique Limitations.
3.0.1	24/05/2018	Mise à jour pour la release 3.0.1

[illegible]

## SOMMAIRE

<b>1. INTRODUCTION .....</b>	<b>5</b>
<b>2. I-CODE CNES IDE ET PLUG-IN ECLIPSE .....</b>	<b>5</b>
<b>2.1. CONFIGURATION DE L'OUTIL .....</b>	<b>5</b>
2.1.1. CHOIX DU TYPE DE CONFIGURATION .....	6
2.1.2. FILTRAGE DES CHECKERS .....	7
2.1.3. ACTIVATION/DESACTIVATION DES CHECKERS .....	7
2.1.4. CONFIGURATION DE LA CRITICITE DES REGLES .....	8
2.1.5. CONFIGURATION DES SEUILS DES METRIQUES .....	9
2.1.6. REINITIALISATION DES CONFIGURATIONS .....	10
<b>2.2. VERIFICATION DU RESPECT DES REGLES DE CODAGE ET CALCUL</b>	
<b>DES METRIQUES .....</b>	<b>10</b>
2.2.1. LANCEMENT DE L'ANALYSE .....	10
2.2.2. AFFICHAGE DES RESULTATS .....	11
2.2.3. PARCOURS DES VIOLATIONS DETECTEES DANS LE CODE .....	15
2.2.4. EXPORT DES RESULTATS .....	16
<b>3. I-CODE CNES COMMAND LINE .....</b>	<b>18</b>
<b>3.1. EXPORTER LES RESULTATS .....</b>	<b>18</b>
3.1.1. FORMAT XML .....	18
3.1.2. FORMAT CSV .....	19
<b>3.2. DETAILLER L'EXECUTION .....</b>	<b>19</b>
<b>3.3. AFFICHAGE DE L'AIDE .....</b>	<b>19</b>
<b>4. AJUSTER LES RESSOURCES UTILISEES PAR I-CODE CNES .....</b>	<b>19</b>
<b>5. TABLES DES VIOLATIONS DETECTEES .....</b>	<b>21</b>
5.1. REGLES COMMUNES .....	21
5.2. REGLES SPECIFIQUES .....	26
5.2.1. FORTRAN 77 .....	26
5.2.2. FORTRAN 90 .....	29
5.2.3. SHELL .....	37
<b>6. TABLES DES METRIQUES CALCULEES .....</b>	<b>40</b>
<b>7. MESSAGES UTILISATEUR .....</b>	<b>43</b>
<b>7.1. MESSAGES DES VIOLATIONS .....</b>	<b>43</b>
7.1.1. REGLES COMMUNES .....	43
7.1.2. FORTRAN 77 .....	45
7.1.3. FORTRAN 90 .....	47
7.1.4. SHELL .....	49
<b>8. LIMITATIONS .....</b>	<b>51</b>
8.1. POWERSHELL NE PERMET PAS LE LANCEMENT D'ANALYSE .....	51
8.2. SHELL : LES CHAINES DE CARACTERES DOIVENT ETRE	
DELIMITEES PAR DES GUILLEMETS .....	51

## 1. INTRODUCTION

Ce document constitue le manuel utilisateur de l'outil i-Code CNES. Il vise à décrire le fonctionnement et l'utilisation de ce plug-in eclipse.



Avant d'utiliser i-Code CNES, il est fortement conseillé de :

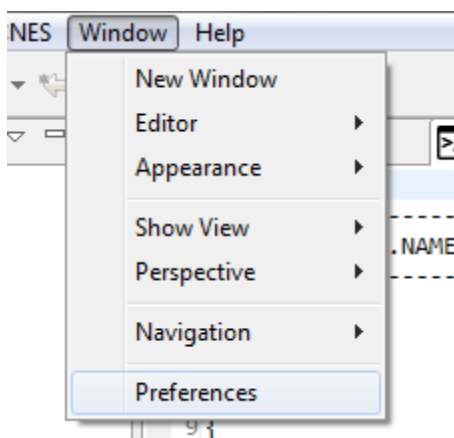
- prendre connaissance de l'environnement eclipse. La documentation d'eclipse est disponible sur le site en référence [R1].
- prendre connaissance des règles normatives du CNES sur les langages à analyser [R2] [R3] [R4]

## 2. I-CODE CNES IDE ET PLUG-IN ECLIPSE

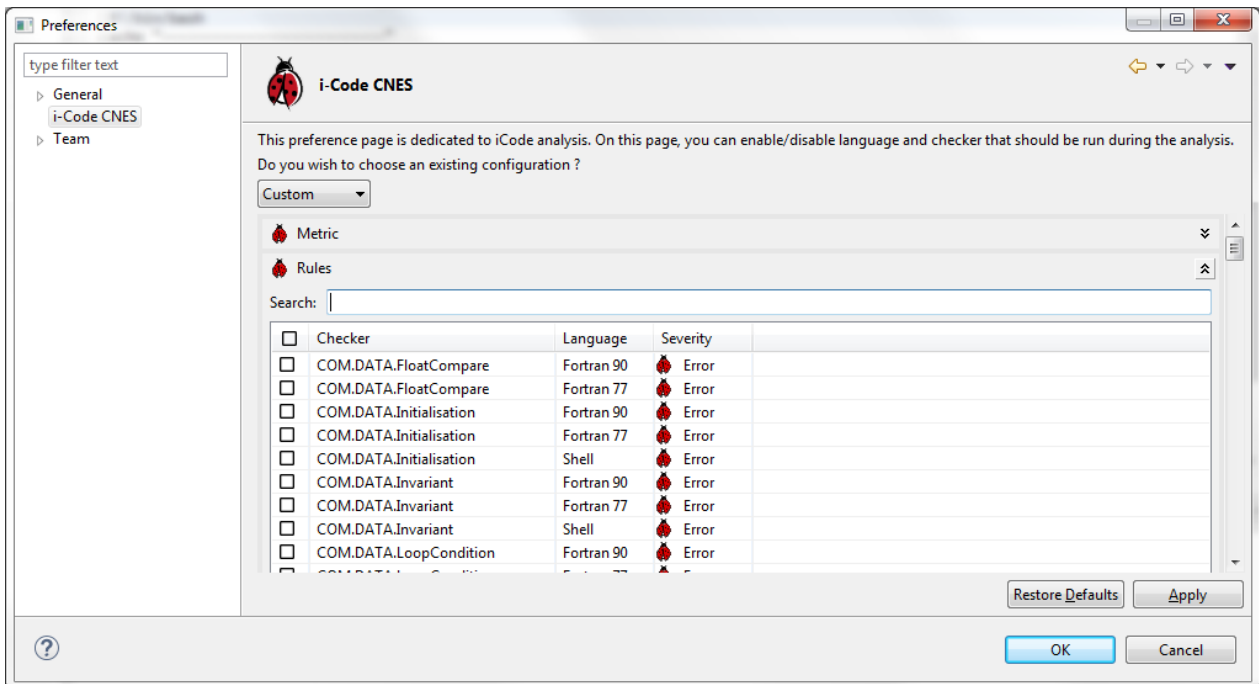
### 2.1. CONFIGURATION DE L'OUTIL

L'ensemble de la configuration de l'outil est accessible dans les pages de préférences **d'Eclipse**.

Pour ouvrir les pages de préférence d'Eclipse, cliquer sur Window > Préférences



Dans la fenêtre des préférences, un item « i-Code CNES » est visible dans la partie gauche, c'est dans cette partie que la configuration i-Code CNES peut être modifiée.

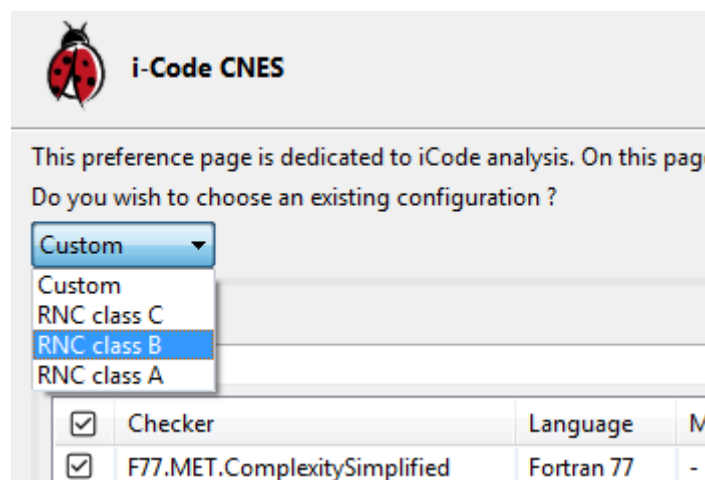


Pour visionner l'ensemble des règles disponibles, il faut dérouler l'onglet des métrique « Metrics » ou bien des règles « Rules » en cliquant sur un des volets.

### 2.1.1. CHOIX DU TYPE DE CONFIGURATION

Les préférences i-Code CNES proposent à l'utilisateur soit de définir lui-même sa propre configuration, soit d'utiliser une configuration par défaut proposée par le plug-in.

Au-dessus des volets « Metrics » et « Rules » un menu déroulant propose de sélectionner la configuration souhaitée.



La configuration **Custom** permet de :

- Activer/désactiver les règles souhaitées ;

- Définir la criticité de chaque règle ;
- Définir des seuils pour les métriques.

Le choix d'une autre configuration bloque les modifications.

### 2.1.2. FILTRAGE DES CHECKERS

Il est possible de filtrer les règles de manière à ne faire apparaître que celle d'un langage, d'une partie d'identifiant ou d'un identifiant en particulier. Pour cela, il suffit de saisir l'identifiant ou le nom de langage désiré dans la barre de recherche du volet.

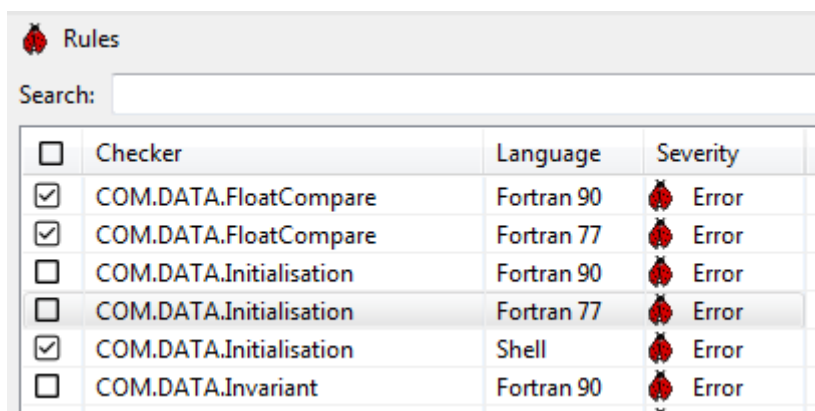
Rules				
Search: Shell				
<input type="checkbox"/>	Checker	Language	Severity	
<input type="checkbox"/>	COM.DATA.Initialisation	Shell	Error	
<input type="checkbox"/>	COM.DATA.Invariant	Shell	Error	
<input type="checkbox"/>	COM.DATA.LoopCondition	Shell	Error	
<input type="checkbox"/>	COM.DATA.NotUsed	Shell	Error	
<input type="checkbox"/>	COM.DESIGN.ActiveWait	Shell	Error	
<input type="checkbox"/>	COM.FLOW.Abort	Shell	Error	
<input type="checkbox"/>	COM.FLOW.BooleanExpression	Shell	Error	
<input type="checkbox"/>	COM.FLOW.CaseSwitch	Shell	Error	
<input type="checkbox"/>	COM.FLOW.Exit	Shell	Error	

### 2.1.3. ACTIVATION/DESACTIVATION DES CHECKERS



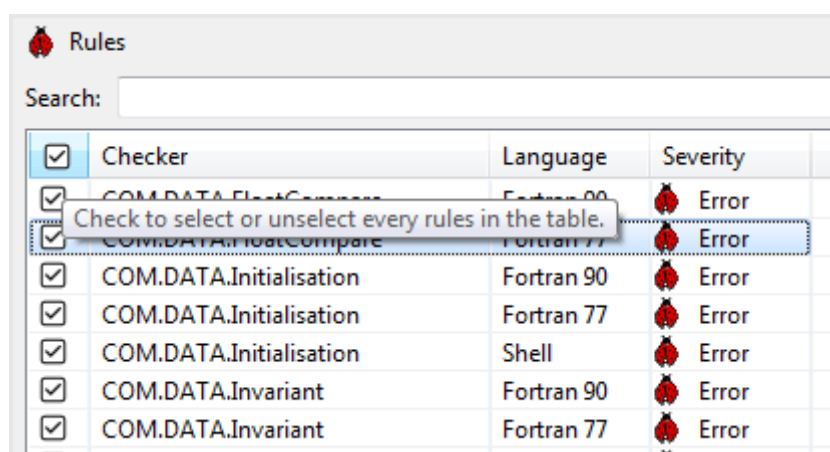
Cette fonctionnalité n'est disponible que lorsque la configuration « Custom » est appliquée.

L'activation des checkers se fait cochant ou en décochant ces derniers du tableau.



<input type="checkbox"/>	Checker	Language	Severity
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 77	Error
<input type="checkbox"/>	COM.DATA.Initialisation	Fortran 90	Error
<input type="checkbox"/>	COM.DATA.Initialisation	Fortran 77	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Shell	Error
<input type="checkbox"/>	COM.DATA.Invariant	Fortran 90	Error
<input type="checkbox"/>	COM.DATA.Invariant	Fortran 77	Error

Pour sélectionner ou désélectionner tous les checkers simultanément, il faut sélectionner/désélectionner la checkbox de la colonne.



<input checked="" type="checkbox"/>	Checker	Language	Severity
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 77	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Fortran 77	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Shell	Error
<input checked="" type="checkbox"/>	COM.DATA.Invariant	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.Invariant	Fortran 77	Error

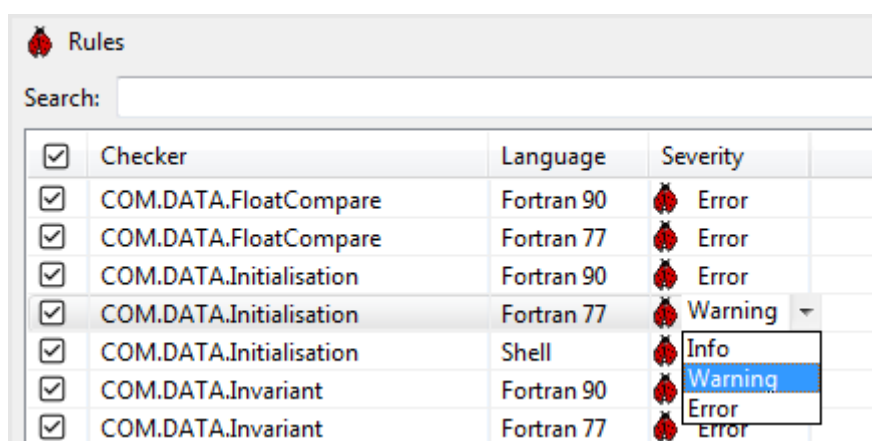
#### 2.1.4. CONFIGURATION DE LA CRITICITE DES REGLES



Cette fonctionnalité n'est disponible que lorsque la configuration « Custom » est appliquée.

Le niveau de criticité des règles est modifiable dans la colonne « Severity » des préférences. Pour cela, il faut cliquer sur la cellule de la colonne « Severity » de la règle que l'on souhaite modifier. Un menu déroulant proposant trois niveaux de criticités apparaît, il suffit ensuite de sélectionner le niveau souhaité.





<input checked="" type="checkbox"/>	Checker	Language	Severity
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.FloatCompare	Fortran 77	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Fortran 90	Error
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Fortran 77	Warning
<input checked="" type="checkbox"/>	COM.DATA.Initialisation	Shell	Info
<input checked="" type="checkbox"/>	COM.DATA.Invariant	Fortran 90	Warning
<input checked="" type="checkbox"/>	COM.DATA.Invariant	Fortran 77	Error

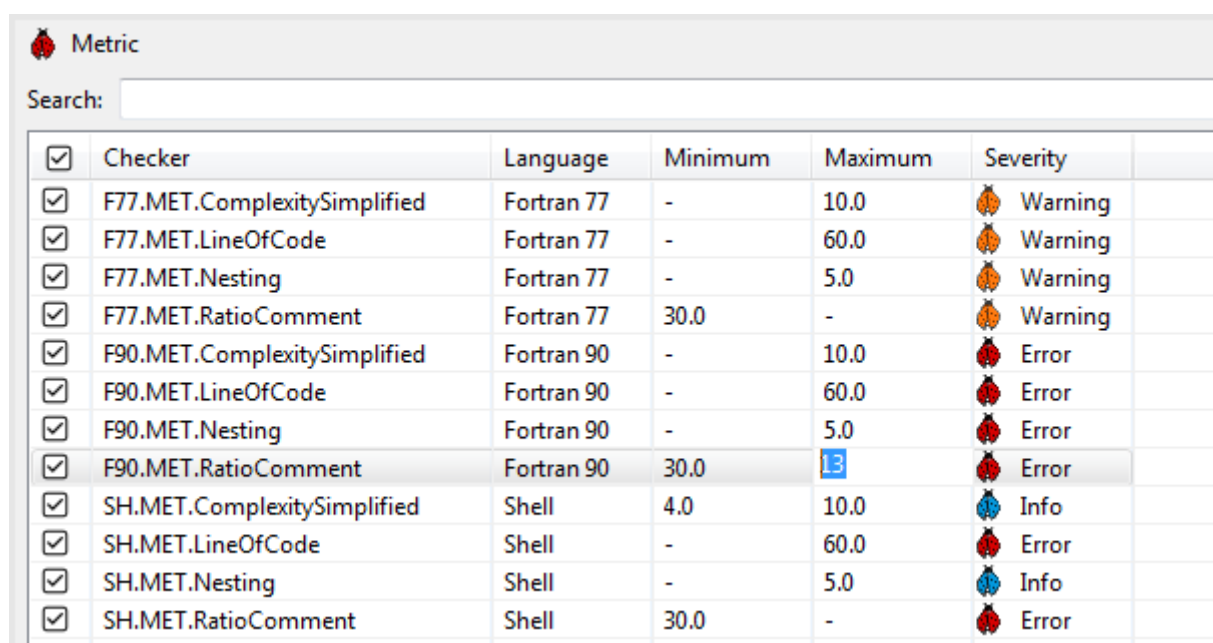
La criticité définie est ensuite visible dans la cellule modifiée.

### 2.1.5. CONFIGURATION DES SEUILS DES METRIQUES



Cette fonctionnalité n'est disponible que lorsque la configuration « Custom » est appliquée.

Le tableau du volet des métriques « Metrics » présente deux colonnes supplémentaires de configurations « Minimum » et « Maximum ». L'édition de ces seuils se fait en sélectionnant la cellule de la colonne « Minimum » ou « Maximum » de la règle à paramétrer.

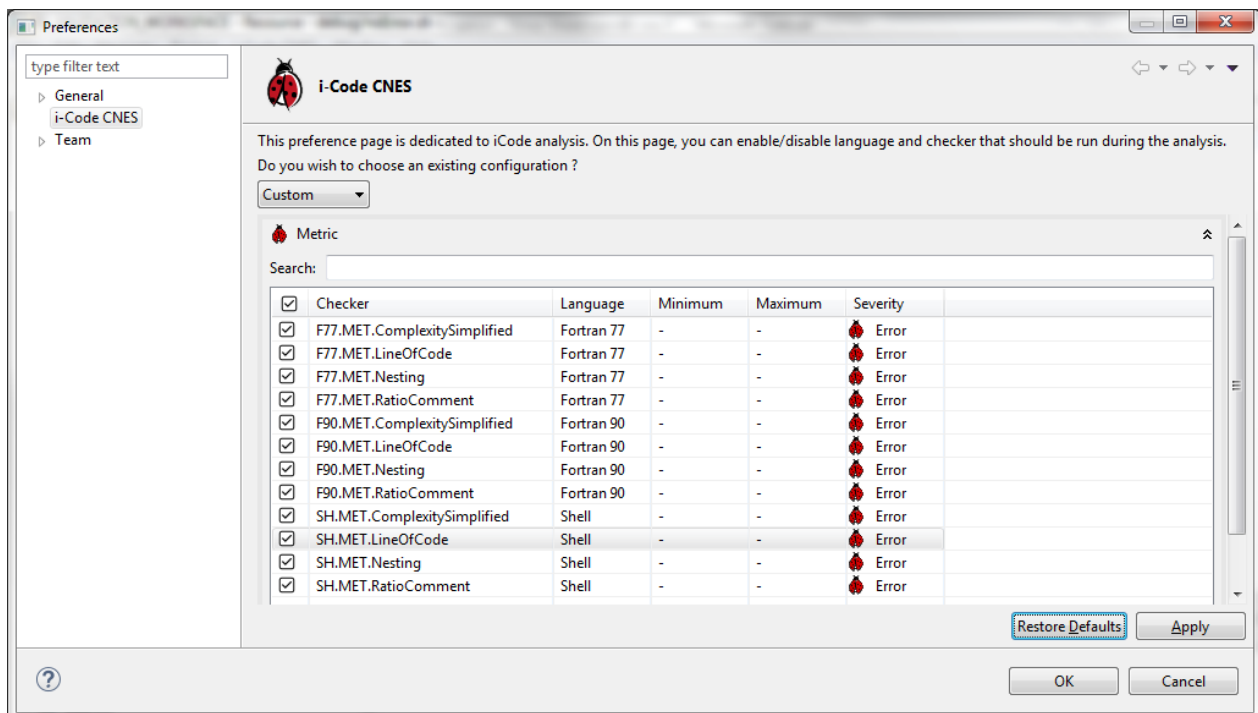


<input checked="" type="checkbox"/>	Checker	Language	Minimum	Maximum	Severity
<input checked="" type="checkbox"/>	F77.MET.ComplexitySimplified	Fortran 77	-	10.0	Warning
<input checked="" type="checkbox"/>	F77.MET.LineOfCode	Fortran 77	-	60.0	Warning
<input checked="" type="checkbox"/>	F77.MET.Nesting	Fortran 77	-	5.0	Warning
<input checked="" type="checkbox"/>	F77.MET.RatioComment	Fortran 77	30.0	-	Warning
<input checked="" type="checkbox"/>	F90.MET.ComplexitySimplified	Fortran 90	-	10.0	Error
<input checked="" type="checkbox"/>	F90.MET.LineOfCode	Fortran 90	-	60.0	Error
<input checked="" type="checkbox"/>	F90.MET.Nesting	Fortran 90	-	5.0	Error
<input checked="" type="checkbox"/>	F90.MET.RatioComment	Fortran 90	30.0	13	Error
<input checked="" type="checkbox"/>	SH.MET.ComplexitySimplified	Shell	4.0	10.0	Info
<input checked="" type="checkbox"/>	SH.MET.LineOfCode	Shell	-	60.0	Error
<input checked="" type="checkbox"/>	SH.MET.Nesting	Shell	-	5.0	Info
<input checked="" type="checkbox"/>	SH.MET.RatioComment	Shell	30.0	-	Error

Après le lancement d'une analyse, lorsque le seuil minimum n'est pas atteint ou le seuil maximum est dépassé, les résultats seront affichés avec pour criticité celle paramétrée. Si aucun seuil « Minimum » ou « Maximum » n'est configuré pour une métrique, cette dernière sera tout de même calculée et les résultats seront affichés au niveau de criticité « Information ».

### 2.1.6. REINITIALISATION DES CONFIGURATIONS

Pour revenir à la configuration par défaut, il faut cliquer sur le bouton « Restore Defaults ».



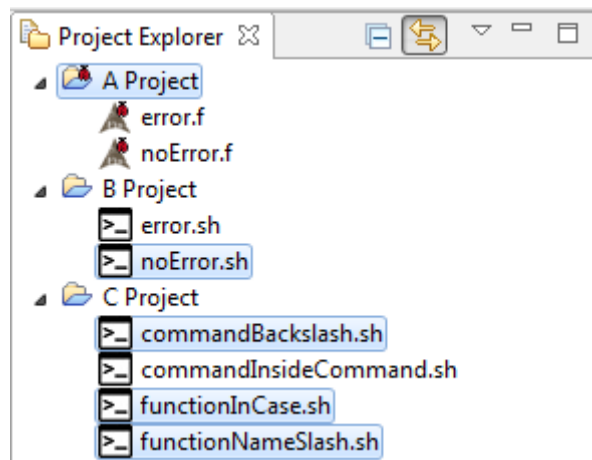
## 2.2. VERIFICATION DU RESPECT DES REGLES DE CODAGE ET CALCUL DES METRIQUES

i-Code CNES permet de vérifier certaines règles de codage des standards de Fortran 77 [R2], Fortran 90 [R3], règles communes [R4] et Shell [R5], de visualiser les résultats dans un tableau et d'accéder rapidement à la ligne incriminée. Il permet également de calculer 4 métriques, et d'exporter l'ensemble des résultats aux formats xml et csv.

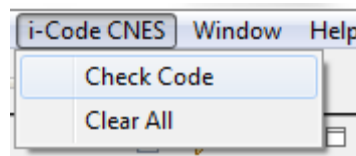
### 2.2.1. LANCEMENT DE L'ANALYSE

**Prérequis :** L'ensemble des fichiers à analyser sont disponibles dans *Project Explorer* Eclipse.

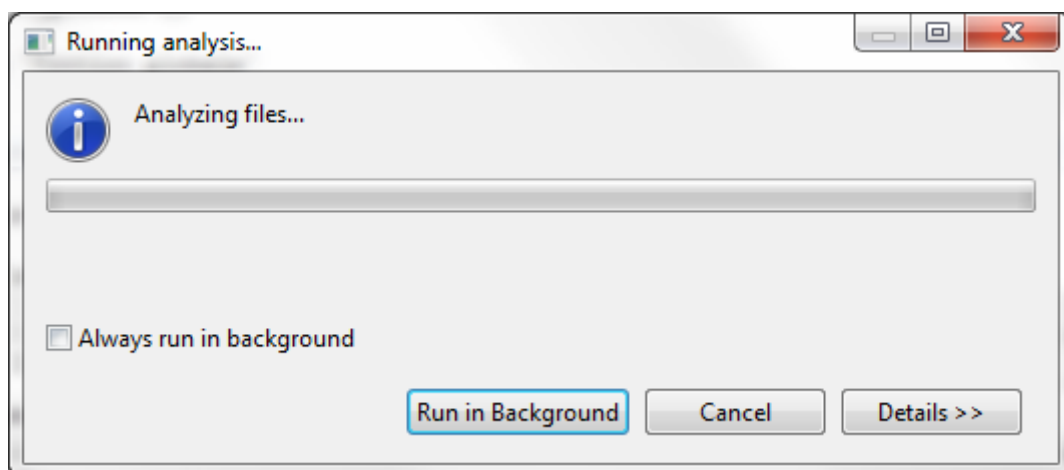
**Etape 1 :** Sélectionner l'ensemble des fichiers que vous souhaitez analyser dans le *Project Explorer*. Vous pouvez sélectionner tout un projet, certains fichiers uniquement, et soumettre n'importe quelle extension de fichier à l'analyseur. Seuls les fichiers pouvant être analysés le seront.



Etape 2 : Lancer l'analyse via le menu i-Code CNES > Check Code.



Une barre de progression apparaît. A la fin de l'analyse, les résultats s'affichent dans la vue « I-Code CNES Violations » et « i-Code CNES Metrics »

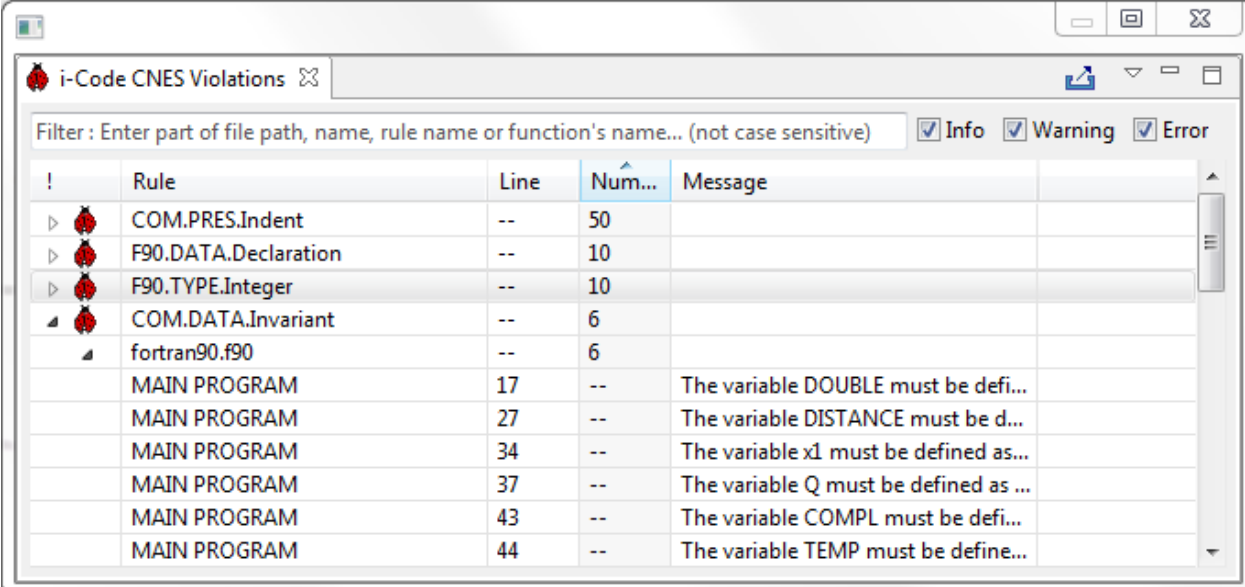


## 2.2.2. AFFICHAGE DES RESULTATS

Le plug-in i-Code CNES propose deux vues d'affichage des résultats. Les résultats des métriques calculées sont affichées dans la vue *i-Code CNES Metrics*, et l'affichage des violations remontées par les checkers se fait dans la vue *i-Code CNES Violations*.

Ces vues sont ouvertes et complétées après le lancement d'une analyse (*voir partie précédente*).

### 2.2.2.1. LA VUE I-CODE CNES VIOLATIONS



The screenshot shows a software window titled "i-Code CNES Violations". It features a search filter at the top: "Filter : Enter part of file path, name, rule name or function's name... (not case sensitive)". Below the filter are three checkboxes: "Info" (checked), "Warning" (checked), and "Error" (checked). The main area contains a table with the following columns: "!", "Rule", "Line", "Num...", and "Message". The table lists several violations, including "COM.PRES.Indent" at line 50, "F90.DATA.Declaration" at line 10, "F90.TYPE.Integer" at line 10, "COM.DATA.Invariant" at line 6, and "fortran90.f90" at line 6. Below these, there are several "MAIN PROGRAM" entries with messages about variable definitions (e.g., "The variable DOUBLE must be defi...", "The variable DISTANCE must be d...", etc.).

!	Rule	Line	Num...	Message
▶	COM.PRES.Indent	--	50	
▶	F90.DATA.Declaration	--	10	
▶	F90.TYPE.Integer	--	10	
▲	COM.DATA.Invariant	--	6	
▲	fortran90.f90	--	6	
	MAIN PROGRAM	17	--	The variable DOUBLE must be defi...
	MAIN PROGRAM	27	--	The variable DISTANCE must be d...
	MAIN PROGRAM	34	--	The variable x1 must be defined as...
	MAIN PROGRAM	37	--	The variable Q must be defined as ...
	MAIN PROGRAM	43	--	The variable COMPL must be defi...
	MAIN PROGRAM	44	--	The variable TEMP must be define...

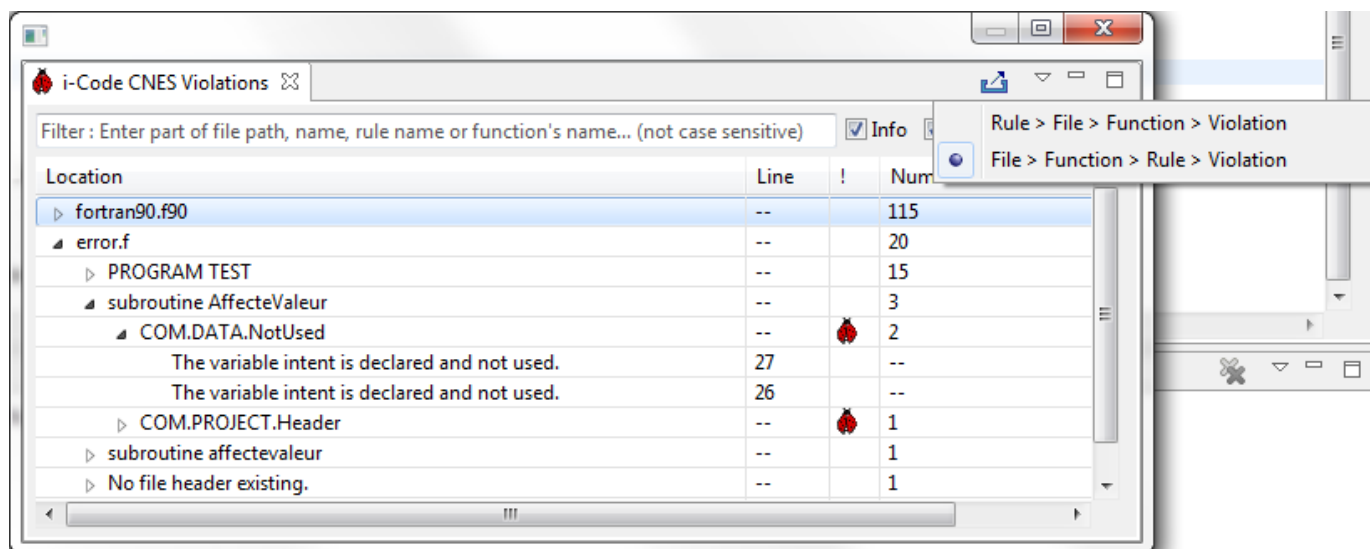
La signification des colonnes de la vue est la suivante :

- *Rule* : nom de la règle qui provoque l'erreur
- *Line* : Numéro de ligne où est détectée l'erreur
- *Number of violations* : nombre de violations total.
- *Message* : détails sur l'erreur (variable qui lance l'erreur, petite description, etc.)

L'ensemble des violations d'une même règle est présenté sous forme arborescente. Le premier niveau correspond au fichier présentant la violation, le deuxième niveau la méthode et la ligne.

#### 2.2.2.1.1. Changer d'arborescence

L'arborescence de l'affichage des règles peut être modifiée en sélectionnant l'onglet de la barre d'outil de la vue :



Il est donc possible de choisir une arborescence :

- Plus orientées pour les analyses de code : Règles > Fichiers > Fonctions > Violations
- Plus orientée pour les développeurs : Fichiers > Fonctions > Règles > Violations

Ces arborescences possèdent les mêmes fonctionnalités d'affichage, seules l'arborescence et l'organisation des colonnes changent.

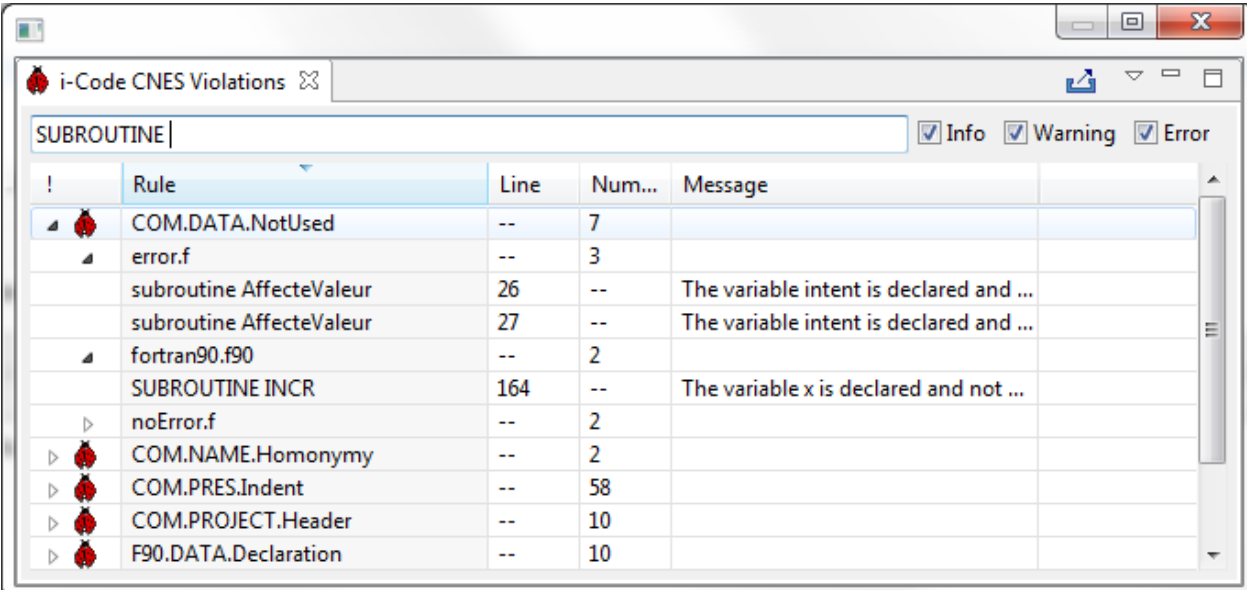
#### 2.2.2.1.2. Filtrer les résultats

Il est possible de filtrer les résultats des tableaux de deux façons :

##### Barre de recherche

La barre de recherche permet de filtrer les règles, le nom des fichiers ou encore la localisation de la violation. Une violation est affichée, avec tous ses éléments parents lorsque la recherche inclue :

- L'identifiant ou une partie de l'identifiant de la règle ;
- Le nom, ou une partie du nom du fichier analysé ;
- La localisation, ou une partie du nom de la localisation de la fonction où se trouve la violation.



!	Rule	Line	Num...	Message
!	COM.DATA.NotUsed	--	7	
!	error.f	--	3	
	subroutine AffecteValeur	26	--	The variable intent is declared and ...
	subroutine AffecteValeur	27	--	The variable intent is declared and ...
!	fortran90.f90	--	2	
	SUBROUTINE INCR	164	--	The variable x is declared and not ...
	noError.f	--	2	
!	COM.NAME.Homonymy	--	2	
!	COM.PRES.Indent	--	58	
!	COM.PROJECT.Header	--	10	
!	F90.DATA.Declaration	--	10	

### Sélection des criticités affichées

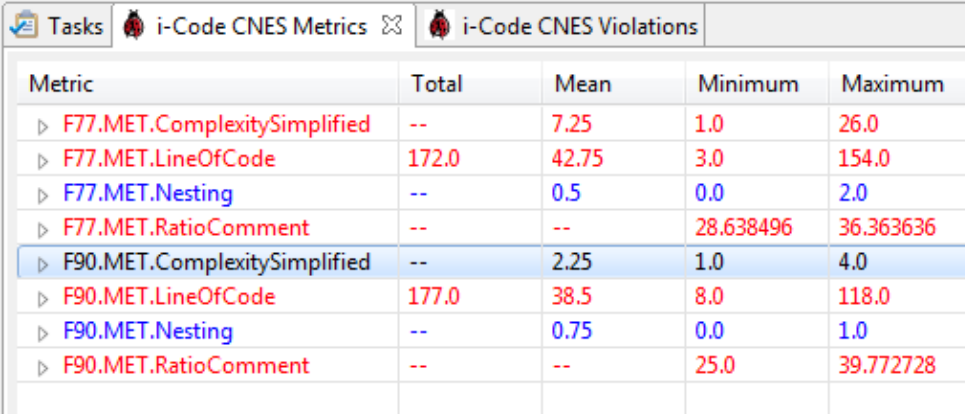
Trois cases à cocher permettent d'afficher ou de masquer les violations en fonction de leur criticité.

#### 2.2.2.1.3. Trier les résultats

Il est possible de trier les résultats des tableaux. Pour cela, il suffit de cliquer sur l'intitulé de la colonne.

#### 2.2.2.2. LA VUE I-CODE CNES METRICS

La vue i-Code CNES Metrics présente l'ensemble des métriques calculées .



Metric	Total	Mean	Minimum	Maximum
▷ F77.MET.ComplexitySimplified	--	7.25	1.0	26.0
▷ F77.MET.LineOfCode	172.0	42.75	3.0	154.0
▷ F77.MET.Nesting	--	0.5	0.0	2.0
▷ F77.MET.RatioComment	--	--	28.638496	36.363636
▷ F90.MET.ComplexitySimplified	--	2.25	1.0	4.0
▷ F90.MET.LineOfCode	177.0	38.5	8.0	118.0
▷ F90.MET.Nesting	--	0.75	0.0	1.0
▷ F90.MET.RatioComment	--	--	25.0	39.772728

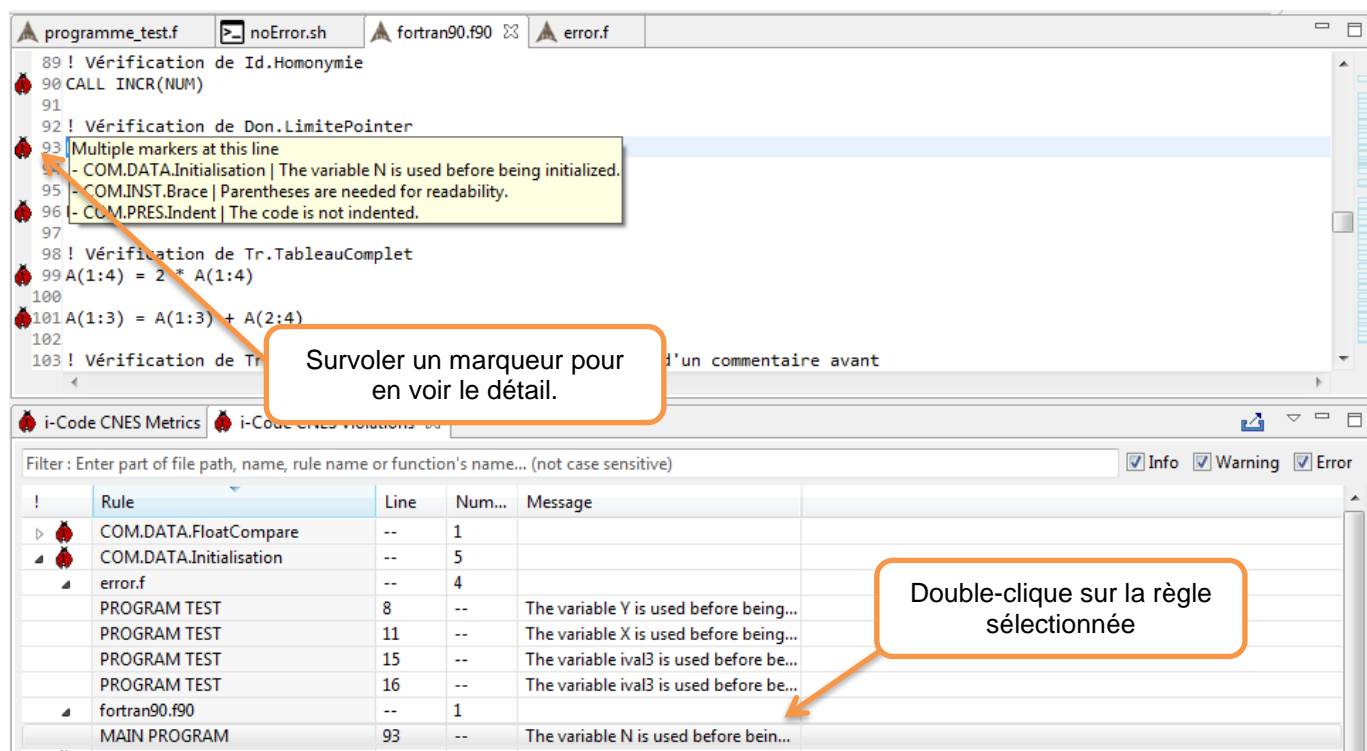
Les dépassements de seuil sont identifiés en rouge dans le tableau.

La signification des colonnes est la suivante :

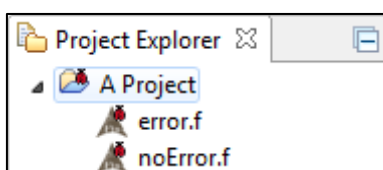
- *Metric* : nom de la métrique qui est analysée
- *Total* : addition de toutes les valeurs de cette métrique pour tous les fichiers analysés
- *Mean* : moyenne des métriques pour les fichiers analysés
- *Minimum* : valeur minimum dans tous les fichiers
- *Maximum* : valeur maximum dans tous les fichiers

### 2.2.3. PARCOURS DES VIOLATIONS DETECTEES DANS LE CODE

La table des résultats présente le fichier et la ligne d'une violation. I-Code CNES permet d'ouvrir directement le fichier à la ligne correspondant à la violation par un double-clic sur la violation. Survoler un marqueur permet d'en voir le détail.



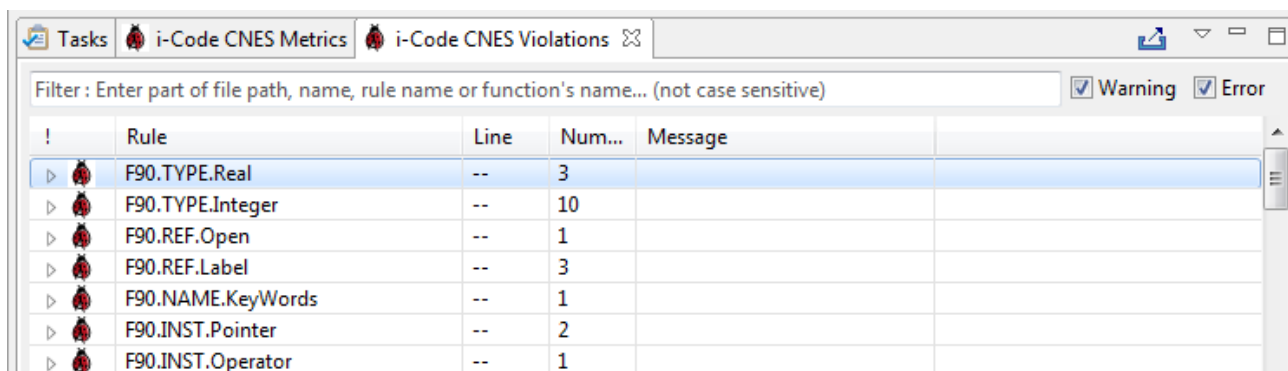
Dans La vue Project Explorer, le marqueur apparait sur les fichiers présentant des violations aux règles de codage.



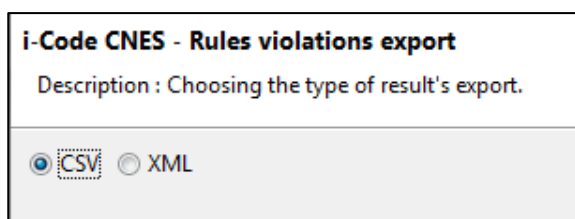
## 2.2.4. EXPORT DES RESULTATS

Les résultats de l'analyse peuvent être exportés dans un fichier au format csv.

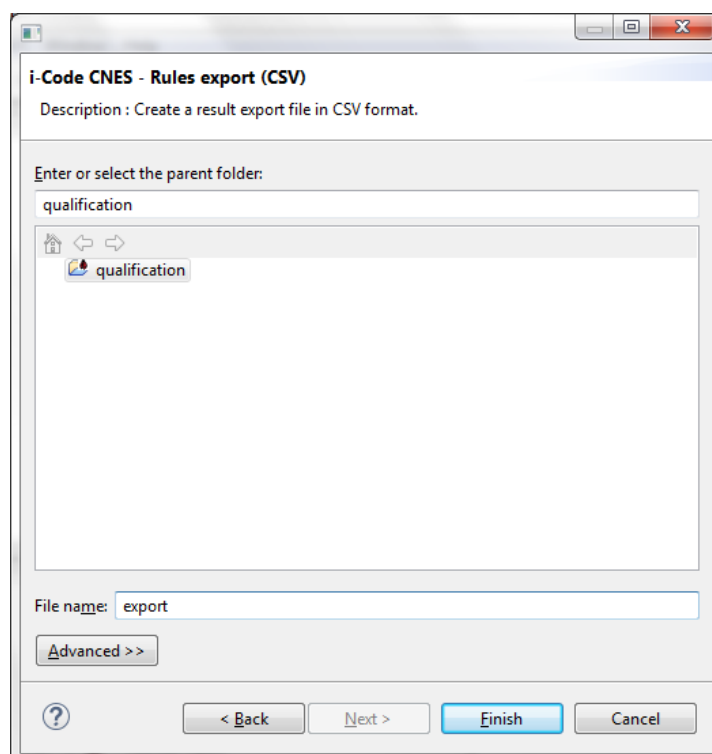
1. Cliquer sur le bouton d'export en haut à droite de la vue « i-Code CNES Violations » ou « i-Code CNES Metrics »



2. Sélectionner le format d'export csv ou xml :

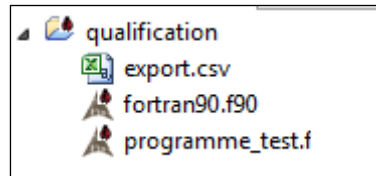


3. Indiquer le nom du fichier d'export et sa localisation.





4. Le fichier est créé dans le projet sélectionné



### 3. I-CODE CNES COMMAND LINE

L'application i-Code CNES en ligne de commande, sous le nom `icode.exe` permet de réaliser les mêmes actions que le plug-in i-Code pour Eclipse.

Pour lancer une analyse, lancer l'exécution de `icode.exe` en indiquant le chemin vers l'exécutable avec les commandes suivantes :

```
icode <fichiers> [<options>]
```

**<fichiers> :**

Les fichiers doivent-être indiqués avec une expression régulière pour indiquer le chemin vers le(s) fichier(s) ou dossier(s) à analyser.

*Exemple :* `icode *.f90 ; icode ./tmp/myfiles.f77 ;`

Pour faciliter l'appel d'`icode.exe`, il est possible d'accéder à l'exécutable via :

- un alias sous Linux ;
- une variable d'environnement sous Windows.

Des fonctionnalités supplémentaires peuvent-être utilisées avec i-Code CNES en ligne de commande. Elles sont détaillées dans les sous-parties suivantes.

#### 3.1. EXPORTER LES RESULTATS

Pour exporter les résultats d'une analyse, il faut utiliser la commande suivante :

```
icode -f <format> <parametres> -output <fichier sortie> <fichiers analysés>
```

**<format> :** Le format d'export ;

**<parametres> :** Ils peuvent-être optionnel ou non selon le format d'export. Ils sont précisés séparément dans les sous-parties suivantes.

**Fichier sortie :** Le fichier d'export des résultats ;

**Fichiers analysés :** Le ou les fichiers à analyser.

##### 3.1.1. FORMAT XML

L'export au format XML permet d'entrer des paramètres supplémentaires pour le fichier d'export. Ces derniers sont optionnels et sont remplacés par leur valeur par défaut lorsqu'ils ne sont pas précisés.

Pour lancer un export dans le format XML il faut préciser « xml » après l'option `f`.

Les paramètres optionnels de l'export XML sont les suivants :

- `-project <nom>` : Le nom du projet analysé.
- `-projectVersion <version>` : La version du projet analysé.

<p style="text-align: center;">CNES DNO/DA/AQ</p>	<p style="text-align: center;"><b>Manuel Utilisateur i-Code CNES</b></p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 19/51</p>
---	--	--

- `-author <nom>` : Nom de l'auteur de l'analyse.
- `-configID <id>` : Identifiant de la configuration d'analyse utilisée.

#### Exemples :

- `icode -f xml -output result.xml *f`
- `icode -f xml -author Me -output result.xml *f ;`
- `icode -f xml -author Me -project MyProject -output export.xml ./tmp/file.sh`

Une fonctionnalité de transformation du fichier XML vers HTML est disponible, pour cela il est nécessaire de préciser en options supplémentaire `-html`, et optionnellement pour exporter ces résultats vers un fichiers `.html`, utiliser l'option `-htmlOutput <fichier>`.

#### Exemples :

- `icode -f xml -output result.xml -html *f`
- `icode -f xml -author Me -output result.xml -html -htmlOutput file.html *f ;`

### 3.1.2. FORMAT CSV

Ce format d'export ne possède pas de paramètres supplémentaires, il doit être précisé en utilisant « csv » après l'option `-f`.

#### Exemples :

- `icode -f csv -output result.xml *f ;`
- `icode -f csv -output export.xml ./tmp/file.sh`

### 3.2. DETAILLER L'EXECUTION

L'option `-v` (pour Verbose) permet d'obtenir le détail de l'exécution de l'export et de l'analyse.

### 3.3. AFFICHAGE DE L'AIDE

L'aide est accessible en lançant la commande `icode -help`.

## 4. AJUSTER LES RESSOURCES UTILISEES PAR I-CODE CNES

Dans sa version client par défaut, les produits i-Code CNES sont paramétrés pour que la JVM ne leur alloue pas plus de 1024Mo.

Néanmoins, l'analyseur d'i-Code a été réalisé de manière à limiter l'utilisation de la mémoire à 90% de la mémoire maximum allouée à i-Code CNES. Modifier la taille arbitraire de 1024Mo permet de lancer i-Code CNES sur des systèmes plus légers ou d'améliorer les performances des analyses réalisées sur des systèmes performants.

Si besoin, il est possible de modifier ce paramétrage dans les fichiers suivants :

- **i-Code CNES IDE** : Le fichier `icode/IDE.ini` se trouvant dans le dossier `icodeIDE`
- **i-Code command line** : Le fichier `icode.ini` se trouvant dans le dossier `icode`
- **Le plug-in Eclipse i-Code** : Le fichier `eclipse.ini` se trouvant dans le dossier Eclipse.

<p><b>CNES</b> DNO/DA/AQ</p>	<p><b>Manuel Utilisateur i-Code CNES</b></p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 20/51</p>
----------------------------------	--	--

Dans chacun de ces fichier la ligne suivante est à modifier avec la valeur souhaitée :

-Xmx1024M

Par exemple :

-Xmx512M

-Xmx2048M

Il est recommandé d'allouer au moins 512M à i-Code CNES pour son bon fonctionnement.

CNES DNO/DA/AQ	Manuel Utilisateur i-Code CNES	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 21/51
-------------------	--------------------------------	---

## 5. TABLES DES VIOLATIONS DETECTEES

### 5.1. REGLES COMMUNES

Règle	Vérification	Couverture du standard
COM.DATA.Array	Obligation d'utiliser les tableaux à deux dimensions de manière à avoir "ligne x colonne"	Non
COM.DATA.DeclarationOrder	Fortran 77 : Non applicable	Non
	Fortran 90 : Les paramètres des fonctions doivent être déclarés, en premier les données d'entrée, après entrée/sortie puis les données de sortie. La déclaration se fait au travers du mot clé 'INTENT' puis 'IN' pour les données d'entrée, INOUT pour les entrées/sorties et OUT pour les sorties. S'il n'y a pas de paramètre INTENT, ce n'est pas possible de faire la vérification.	Oui
	Shell : non applicable	Non
COM.DATA.FloatCompare	Comparaison d'égalité/inégalité (.EQ., ==, .NE., /=) interdite entre des nombres réels (REAL, DOUBLE PRECISION ou COMPLEX).	Oui
	Shell : non applicable	Non
COM.DATA.Initialisation	Les variables doivent être initialisées avant d'être utilisées. Quand une variable est utilisée dans le code, le programme vérifie qu'elle est initialisée (nom de la variable puis signe d'égalité), sinon, il renvoie une erreur.	Oui
	Shell : Non implémenté	Non
COM.DATA.Invariant	Les données déclarées dans une subroutine, fonction, etc et jamais modifiées (pas d'occurrence de la variable puis signe d'égalité) doivent être définies comme constantes	Oui
	Shell : Non implémenté	Non
COM.DATA.LoopCondition	Interdiction de modifier les données de condition de sortie des boucles à l'intérieur de celle-ci	Oui
COM.DATA.NotUsed	Fortran : Toute variable déclarée doit être utilisée, sinon une erreur est remontée.	Oui
	Shell : Toute variable déclarée doit être utilisée.	
	Limitation : Les assignations dans les options de la commande awk peuvent engendrer des faux-positifs. Une variable est considérée comme utilisée si elle on l'utilise avec \${variable}.	
COM.DATA.Using	Interdiction de réutiliser un objet local dans des traitements de type différent.	Non

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 22/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture du standard
COM.DESIGN.Alloc	Fortran : L'allocation et la desallocation des ressources doit être dans le même niveau (fonction, sous-routine, ...). Chaque fois que le mot 'DEALLOCATE' est trouvé, le programme vérifie qu'il y a le mot 'ALLOCATE' et que les deux utilisent la même ressource.	PC <sup>1</sup>
	Shell : non applicable	Non
COM.DESIGN.ActiveWait	Fortran : dans une boucle, les instructions SLEEP, WAIT et PAUSE sont interdites.	Oui
	Shell : La boucle WHILE [1] et le mot READ sont interdits	Oui
COM.FLOW.Abort	Fortran : Le mot STOP est interdit.	Oui
	Shell : Les mots KILL, PKILL et KILLALL sont interdits Limitation : Ne prend pas en compte les reutrn, break et exit, même s'ils peuvent interrompre l'exécution de commandes.	
COM.FLOW.BooleanExpression	Dans une instruction conditionnelle (IF, DO) il n'est pas possible de définir plus de cinq expressions conditionnelles (AND, OR, NEQV, XOR, EQV, NOT, LT, <, LE, <=, GT, >, GE, >=, EQ, ==, NE, /=).	Oui
COM.FLOW.CaseSwitch	Fortran77 : Non Applicable	Non
	Fortran90 : Obligation de finir l'instruction SWITCH avec DEFAULT, afin de traiter tous les cas possibles.	Oui
	Shell : Obligation de finir l'instruction CASE avec *), afin de traiter tous les cas possibles	Oui
COM.FLOW.CheckArguments	Fortran : Obligation de contrôler les paramètres passés à un programme	Non
	Shell : voir SH.FLOW.CheckArguments	
COM.FLOW.CheckCodeReturn	Fortran : Obligation de tester tous les retours de fonction	Oui
	Shell : renommée en SH.FLOW.CheckCodeReturn	Non
COM.FLOW.CheckUser	Fortran : Obligation de vérifier l'identité de l'utilisateur qui exécute un programme	Oui
	Shell : voir SH.FLOW.CheckUser	Non
COM.FLOW.Exit	Fortran : Interdiction d'implémenter plusieurs points de sortie dans les fonctions, procédures ou méthodes.	Oui

<sup>1</sup> Les ressources vérifiées sont les blocs de mémoire, pas les fichiers. Si le développeur encapsule les allocations et desallocations dans les sous-routines, l'application remonte une erreur.

CNES DNO/DA/AQ	Manuel Utilisateur i-Code CNES	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 23/51
-------------------	--------------------------------	---

Règle	Vérification	Couverture du standard
	Shell : Non implémenté	Non
COM.FLOW.ExitLoop	Interdiction d'implémenter plus d'une sortie dans les boucles.	Oui
COM.FLOW.FileExistence	Fortran : Avant d'ouvrir ou créer (mot clé OPEN, READ, WRITE) un fichier, doit apparaître l'instruction INQUIRE avec le flag EXIST sur le même fichier.	Oui
	<u>Shell :</u> Avant d'accéder à un fichier ( > nom_fichier), il faut apparaître la vérification : if [ ! -f \$nom_fichier ]  <u>Limitations :</u> <ul style="list-style-type: none"> <li>Pas de détection dans les commandes \$(...) ou `...`</li> <li>Une variable (or redirections standards) sur laquelle est réalisée une redirection peut être interprétée comme un fichier</li> </ul>	Oui
COM.FLOW.FilePath	Dans l'instruction OPEN, il est interdit d'utiliser directement le nom du fichier (fichier.txt). Le chemin d'accès doit être défini au travers d'une variable qui contient le chemin vers le fichier.	Oui
	Shell : Non implémenté	Non
COM.FLOW.Recursion	Fortran77 : Non Applicable	Non
	Fortran90 : Interdiction d'utiliser la récursivité. En Fortran, une fonction récursive est définie comme suit :  RECURSIVE FUNCTION (params)	Oui
	Shell : Non implémenté	Non
COM.INST.BoolNegation	La double négation est interdite sur les expressions booléennes. Les négations sont définies avec le mot .NOT. donc les expressions suivantes ne sont pas permis :  <div> <div>.NOT. (.NOT. a)</div> <div>-&gt; ( a )</div> </div> <div> <div>.NOT. ( a .AND. .NOT. b )</div> <div>-&gt; .NOT. a .OR. b</div> </div>	Oui
	Shell : Non implémenté	Non
COM.INST.Brace	Fortran : Toute expression doit être parenthésée, ainsi le nombre de parenthèses ouvertes doit être supérieur ou égal au nombre d'opérateurs utilisés (+, -, *, /, **) <div> <div>a + b * c</div> <div>-&gt; a + ( b * c )</div> </div>	Oui
	Shell : Non implémenté	Non
COM.INST.CodeComment	Fortran : Interdiction de commenter le code. Tout mot clé (ASSIGN, BACKSPACE, BLOCK DATA, CALL, ...) dans une ligne de commentaire est un erreur.. Le header (commentaires juste	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 24/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture du standard
	avant ou juste après de la déclaration de la fonction ou sous-routine) peut contenir ces mots.	
	<p>Shell : Interdiction de commenter le code. Le premier mot après le symbole de commentaire (#), ne doit pas être un mot clé (cd, date,...), ni une affectation de variable (var=).</p> <p>Limitation : un texte courant commençant par un mot clé du langage sera incorrectement détecté comme du code commenté.</p> <p>Ex : # date de mise à jour # set the starting value</p>	Oui
COM.INST.GOTO	Fortran: L'instruction GO TO est interdite.	Oui
	Shell: non applicable	Non
COM.INST.Line	Fortran 77 : non applicable	Non
	<p>Fortran 90 et Shell : Chaque ligne doit contenir maximum une expression. En Fortran il est possible d'implémenter plusieurs instructions dans une ligne grâce au point-virgule.</p> <p style="text-align: center;"> <math>a = b + c ; d = e * f</math>                      <math>\rightarrow a = b + c</math>  <math>d = e * f</math> </p>	Oui
	<p>Shell: Interdiction de réaliser plusieurs instructions sur une même ligne.</p> <p>Limitation : L'utilisation de mots clés dans les messages de commandes non encadrés de guillemets peut engendrer des erreurs sur l'analyse du fichier.</p>	
COM.INST.LoopCondition	Dans une instruction de boucle, les comparaisons d'égalité ( .EQ., ==) ou de différence ( !=, .NE.) sont interdites.	Oui
	Shell : Non implémenté	Non
COM.NAME.Homonymy	Fortran : Une variable doit avoir un nom unique. Chaque fois qu'une variable est trouvée, le programme vérifie que le nom de cette variable n'est pas déjà utilisé dans le programme.	Oui
	Shell : Non implémenté	Non
COM.PRES.Data	Obligation de commenter par une description détaillée les objets importants.	Non
COM.PRES.Indent	Le code doit être indenté par espaces. Une ligne doit commencer à la même colonne que la ligne précédente. Après l'instruction DO, IF, WHILE, WHERE, SELECT et TYPE la ligne doit commencer	Oui



CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 25/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture du standard
	<p>dans une colonne supérieure, à l'exception de la fin de l'expression (dénnoté par END). Exemple :</p> <pre> DO i = 2, nb      somme = somme + x(i)      IF (isnan(somme)) THEN          print *, 'somme is a NaN'          moy = -1.0      END IF  END DO </pre>	
	<b>Shell : Non implémenté</b>	<b>Non</b>
COM.PRES.LengthLine	Une ligne de code doit contenir un maximum de 100 caractères. Le caractère 101 doit être écrit à la ligne suivante.	<b>Oui</b>
	<b>Shell : Non implémenté</b>	<b>Non</b>
COM.PROJECT.Analyser	Obligation de passer un outil d'analyse statique sur tous les codes sources d'un projet.	<b>Non</b>
COM.PROJECT.CodeCloning	Interdiction de dupliquer / cloner du code.	<b>Non</b>
COM.PROJECT.Header	<p>Obligation de définir et d'appliquer les entêtes/cartouche de chaque module et fonctions en début de projet.</p> <p><i>Shell : nommer COM.PRES.Header</i></p>	<b>Oui</b>
COM.PROJECT.Warnings	Obligation d'afficher tous les warnings et de les corriger	<b>Non</b>
COM.TYPE.Expression	Fortran : Dans une expression (une expression est définie par un opérateur comme +, -, *, /, **), Les variables doivent être de même type : soit REAL, soit INTEGER, etc.	<b>Oui</b>
	<b>Shell : Non implémenté</b>	<b>Non</b>
	<i>Shell : Les variables d'une même expression doit être du même type : integer ou char/string définis selon première initialisation.</i>	<b>Non</b>

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 26/51
-------------------	---------------------------------------	---

## 5.2. REGLES SPECIFIQUES

### 5.2.1. FORTRAN 77

Règle	Vérification	Couverture au standard
F77.BLOC.Common	Interdiction d'utiliser des COMMON blanc.	Oui
F77.BLOC.Else	Dans une instruction IF, le dernier ELSE IF doit toujours être suivi d'un ELSE.	Oui
F77.BLOC.File	Obligation d'utiliser les instructions OPEN et CLOSE pour accéder aux fichiers.	Non
F77.BLOC.Function	Obligation d'utiliser les parenthèses d'argument pour l'instruction FUNCTION, même s'il n'y a pas d'argument	Oui
F77.BLOC.Loop	Les boucles DO imbriquées doivent avoir des indicateurs de fermeture différents. Ce n'est pas possible de partager le label.	Oui
F77.DATA.Array	Obligation de déclarer explicitement les dimensions des tableaux. Par contre, est possible d'utiliser la notation * pour la dernière dimension mais toujours avec la justification d'un commentaire avant.  A(*), A(4, *), A(4, *, *), A(4, 4, *), mais A(*, 4)	Oui
F77.DATA.Common	Obligation d'utiliser l'instruction INCLUDE pour déclarer les COMMON dans les unités de programmes qui les référencent.	Oui
F77.DATA.Double	Dans une initialisation de constante ou dans l'évaluation d'une expression arithmétique, l'utilisateur souhaite que cette constante soit évaluée en double précision, la présence d'un exposant double précision (lettre D) est obligatoire.	Oui
F77.DATA.Initialisation	Obligation d'initialiser toutes les variables avant leurs utilisations avec l'instruction DATA ou BLOCKDATA.	Oui
F77.DATA.IO	les unités logiques implicites définies par * sont interdites.  READ (*,f) [iolist], READ f [,iolist], WRITE (*,f) [iolist], PRINT f [,iolist]	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 27/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
F77.DATA.Loopedo	Obligation d'utiliser un type ENTIER comme paramètre de contrôle des boucles DO.	Oui
F77.DATA.Parameter	Interdiction d'utiliser des constantes, expression calculé ou appel de fonction comme paramètres de fonction.  CALL function (3, x*y, f(z), var)	Oui
F77.ERR.OpenRead	Obligation des tester le statut de retour des instructions OPEN et READ, de préférence à l'aide du paramètre " IOSTAT = ", et vérifier le value de cette variable.	Oui
F77.INST.Assign	Interdiction d'utiliser l'instruction ASSIGN.	Oui
F77.INST.Dimension	Interdiction d'utiliser l'instruction DIMENSION.	Oui
F77.INST.Equivalence	Interdiction d'utiliser l'instruction EQUIVALENCE.	Oui
F77.INST.Function	Il faut utiliser l'instruction FUNCTION avec une déclaration explicite de type, à la définition de la fonction.	Oui
F77.INST.If	Interdiction d'utiliser le IF arithmétique :  IF (Expression arithmétique) e1,e2,e3  Où les « eN » sont des étiquettes.	Oui
F77.INST.Include	Avec une instruction INCLUDE, le fichier inclus, ne peut pas inclure instructions exécutables (ASSIGN, GOTO, IF, ELSE, CONTINUE, STOP, PAUSE ; DO, READ, WRITE, PRINT, REWIND ;BACKSPACE, ENDFILE, OPEN, CLOSE, INQUIER, CALL, RETURN, END)	Oui
F77.INST.Pause	Interdiction d'utiliser l'instruction PAUSE.	Oui
F77.INST.Return	L'instruction RETURN(i) est interdite dans les sous-programmes.	Oui
F77.INST.Save	Interdiction d'utiliser l'instruction SAVE hormis pour des variables locales avec une justification par commentaire.	Oui
F77.MET.Line	Interdiction de dépasser 72 caractères par ligne.	Oui
F77.NAME.GenericIntrinsic	Obligation d'utiliser les noms génériques des fonctions intrinsèques.	Oui
F77.NAME.Intrinsic	Interdiction de réutiliser les noms des fonctions intrinsèques. Quand une fonction définie par le développeur a le même nom que une fonction intrinsèque (définie aux standards), l'application lance une erreur	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 28/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
F77.NAME.KeyWords	Interdiction de réutiliser les mots-clés u Fortran77 pour les variables	Oui
F77.NAME.Label	Restriction des étiquettes aux instructions FORMAT et CONTINUE.	Oui
F77.PROTO.Declaration	Obligation de déclarer les fonctions externes (lesquelles qu'elles soient pas dans le même fichier) par le mot EXTERNAL avant de leur utilisation.	Oui
F77.REF.IO	Obligation d'identifier les unités logiques par un nom symbolique.  <b>READ (5, *) NOMBRE -&gt; READ (STDIN, *) NOMBRE</b>	Oui
F77.REF.Open	Obligation de définir les paramètres FILE, STATUS et POSITION de l'instruction OPEN	Oui
F77.REF.Parameter	Interdiction de transmettre comme paramètre d'une subroutine els variables que son déjà dans un bloc COMMON accessible par la subroutine et le programme qui l'appel.  <b>COMMON /CONTROL/ A, B, C, D</b>  <b>...</b>  <b>PROGRAM ESSAI</b>  <b>CALL MY_SUB1 (A, B, C, D)</b>  <b>...</b>  <b>END PROGRAM ESSAI</b>  <b>....</b>  <b>SUBROUTINE MY_SUB1 (C _A, B, _C, C_D)</b>	Oui
F77.TYPE.Basic	Obligation d'utiliser les types standards (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, CHARACTER) uniquement. Autres typs non standards seront considérées erreurs <b>INTEGER*4, LOGICAL*n</b>	Oui
F77.TYPE.Hollerith	Les données et les constantes de type HOLLERITH sont interdites.  Une donnée Hollerith est de la forme : numeroH par exemple 0.8H	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 29/51
-------------------	---------------------------------------	---

### 5.2.2. FORTRAN 90

Règle	Vérification	Couverture au standard
F90.DESIGN.Free	Obligation de libérer la mémoire allouée dans le même niveau conceptuel.	<b>Non</b>
F90.DESIGN.Include	L'include d'un fichier est interdit. Si l'INCLUDE contient un fichier écrit en F90, le programme retourne une erreur.  INCLUDE 'file_to_include.F90'	<b>Oui</b>
F90.DESIGN.Interface	Le contenu des modules doit être limité aux clauses USE, PRIVATE et PUBLIC  MODULE interface_syslog  Implicit none  PRIVATE  Interface  Subroutine f_syslog(cdata)  USE message_syslog  Type(opendata_type)  End subroutine  End interface  PUBLIC f_syslog  End module interface_syslog	<b>Oui</b>
F90.DESIGN.IO	Le nombre d'unité dans une fonction OPEN doit dépendre d'une autre fonction ou un tableau.  Integer :: f_unit = 15                    integer :: f_unit  OPEN (UNIT = f_unit, ...) -> f_unit = getNumber()  OPEN(UNIT = f_unit, ...)	<b>PC<sup>2</sup></b>
F90.DESIGN.Obsolete	Cette règle vérifie les clauses suivantes :  - Ne pas utiliser le GOTO calculé.	

<sup>2</sup> La règle demande de vérifier ces trois cas : 1) Des primitives d'allocation et de libération de numéros d'unité. 2) Une primitive de réservation d'un numéro d'unité donné, pour permettre l'utilisation de sous-programmes Fortran 77 utilisant des numéros fixés. 3) Des constantes nommées pour l'entrée et la sortie standard. Notre vérification inclut les options 2 et 3.

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 30/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	<ul style="list-style-type: none"> <li>- Ne pas utiliser la syntaxe : CHARACTER*N</li> <li>- Ne pas utiliser le IF arithmétique</li> <li>- Dans une boucle DO, ne pas utiliser de variables réelles ni comme indice, ni pour les bornes de l'intervalle de contrôle, ni pour le pas d'incrément.</li> <li>- Ne pas utiliser de terminaison de boucle DO autre que END DO ou CONTINUE.</li> <li>- Ne pas faire de branchements sur ENDIF.</li> <li>- Ne pas utiliser l'instruction PAUSE.</li> <li>- Ne pas utiliser l'instruction GOTO assigné.</li> <li>- Ne pas utiliser l'affectation d'étiquette de FORMAT</li> <li>- Ne pas utiliser le descripteur H (Hollerith) dans les formats.</li> </ul>	Oui
F90.BLOC.File	<p>Tout fichier ouvert doit être fermé. Le programme cherche l'instruction CLOSE pour chaque fichier ouvert en amont.</p> <pre> OPEN ( unit = f_unit, ...)  ...  CLOSE ( unit = f_unit, ...) </pre>	Oui
F90.DATA.Array	<p>La dimension du tableau doit être respectée. Il faut obligatoirement utiliser les paramètres qui ont été déclarés à la création du tableau lors de l'appel dans une fonction ou une procédure.</p> <p>Exemple :</p> <pre> Subroutine s1(tab)      -&gt;      Subroutine s1(tab, x) Integer :: tab(2)       Integer :: tab(x) </pre>	Oui
F90.DATA.ArrayAccess	<p>Dans un tableau d'indirection n'est pas possible de spécifier plusieurs fois le même élément.</p> <pre> Integer,dimension(3) :: a  Integer,dimension(3) :: b  a = ( / 1,1,3 / ) </pre>	Oui

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 31/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	$b(a) = ( / 1,2,3 / )$	
F90.DATA.Constant	<p>Les constantes qui apparaissent dans plusieurs sous-programmes doivent être définies dans un module.</p> <pre> Subroutine s1()     Real PI = 3,141519 End subroutine s1                                 module precision                                 -&gt;    real PI = 3.141519 Function f1()                                end module precision     Real PI = 3,141519 End function </pre>	Oui
F90.DATA.ConstantFloat	<p>Les constantes littérales numériques doivent être suivies par le paramètre de sous-type</p> <pre> Integer,parameter :: DOUBLE=selected_real_kind(15) Real (DOUBLE) :: x = 0.1_DOUBLE </pre>	Oui
F90.DATA.Declaration	<p>Il est obligatoire de mettre l'instruction IMPLICITE NONE après chaque en-tête de méthode. De plus chaque variable devra au préalable être déclaré avant leur utilisation. Cette règle n'est pas prise en compte pour les fonctions et les tableaux.</p>	Oui
F90.DATA.Float	<p>Est interdit d'utiliser le format * en sortie (instruction WRITE) pour les nombres flottants</p> <pre> Real :: X Write ( std_out, * ), x </pre>	Oui
F90.DATA.Parameter	<p>Les fonctions intrinsèques SELECTED_REAL_KIND et SELECTED_INT_KIND doivent être regroupées dans un module.</p> <pre> MODULE precision     Integer, parameter :: DOUBLE = SELECTED_REAL_KIND(15) END MODULE </pre>	Oui
F90.ERR.Allocate	<p>L'allocation (ALLOCATE) et libération (DEALLOCATE) doit contenir le paramètre STAT. Ensuite, le valeur du STAT doit être testé après l'instruction.</p>	Oui

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 32/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	ALLOCATE(x, STAT = iom)  IF (iom > 0) THEN ...	
F90.ERR.OpenRead	<p>Les instructions OPEN et READ qui travaille avec des fichiers doivent contenir le paramètre IOSTAT, et vérifier le value de cette variable. Pour vérifier ça, l'instruction OPEN doit contenir l'attribut FILE. Une exemple qui ne respecte pas la règle est :</p> <p><b>OPEN (UNIT = FILE_UNIT, FILE = C_ARG)</b></p> <p>Parce que il n'y a pas d'attribut IOSTAT, et le value de IOSTAT n'est pas testé. Le même exemple correct sera :</p> <p><b>OPEN (UNIT = FILE_UNIT, FILE = C_ARG, IOSTAT=IOS)</b></p> <p><b>IF ( IOS .NE. 0) ...</b></p> <p>Pour l'instruction READ, comme l'attribut FILE n'existe pas, la vérification sera sur ces reads que le UNIT désigne une unité logique qui a été ouverte par un OPEN avant a le code. En continuation avec l'antérieur exemple :</p> <p><b>READ (*, *) //pas vérifiable, lecture du clavier</b></p> <p><b>READ (UNIT = FILE_UNIT, FMT = 9011, IOSTAT = IOS) // verifiable</b></p> <p><b>IF (IOS .LT. 0) ...</b></p>	Oui
F90.INST.Associated	<p>Entre le mot ASSOCIATED et la déclaration il faut avoir l'instruction NULLIFY.</p> <pre> graph TD     decl[déclaration] --&gt; eq1[=&gt;]     decl --&gt; nullify[nullify]     decl --&gt; assoc1[associated]     eq1 --&gt; assoc2[associated]     assoc2 -- V     nullify --&gt; eq2[=&gt;]     nullify --&gt; assoc3[associated]     eq2 --&gt; assoc4[associated]     assoc4 -- V     assoc3 -- V     assoc1 -- X </pre> <p>Exemple :</p> <p>Real, pointer :: ptr</p> <p>...</p>	Oui



<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 33/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	NULLIFY(ptr) ASSOCIATED(ptr)	
F90.INST.Entry	L'instruction ENTRY est interdite Subroutine s1 ENTRY rien	Oui
F90.INST.Equivalence	Interdiction d'utiliser l'instruction EQUIVALENCE. INTEGER total (3,2) INTEGER sum (6) EQUIVALENCE (sum, total)	Oui
F90.INST.If	L'instruction IF logique est interdite quand est suivi par un mt autre que EXIT, CYCLE, GOTO et RETURN. IF (x == 0) THEN GO TO 1000 End IF	Oui
F90.INST.Intent	Chaque paramètre des sous-programmes doit avoir le mot clé INTENT à sa déclaration. Function f1(x, y, z) Integer, INTENT(IN) :: x Integer, INTENT(IN) :: y Integer, INTENT(OUT) :: z	Oui
F90.INST.Nullify	Après une desallocation il y a obligation d'utiliser l'instruction NULLIFY sur la même unité logique. DEALLOCATE (C, stat = iom ) NULLIFY ( C )	
F90.INST.Only	Interdiction d'utiliser le mot clé ONLY sans commentaire avant qui explique son utilisation. Use mes_fonctions_intrinseques , ONLY :: getuid -> my_getuid	Oui
F90.INST.Operator	Ne pas utiliser la notation ancienne pour les opérateurs relationnels. Substituer .EQ., .NE., .LT., .LE., .GT., .GE. pour ==, /=, <, <=, >, >=	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 34/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
F90.INST.Pointer	<p>Le POINTER est interdit à exception des cas suivantes :</p> <ul style="list-style-type: none"> <li>- pour créer des structures de données complexes (i.e. liste de chaînes, arbre, etc.) ;</li> <li>- pour manipuler des références à des tableaux (référence à un tableau alloué dans un sous-programme, recopie par échange de pointeurs,...) et à des parties de tableaux;</li> <li>- pour utiliser de l'allocation dynamique dans les composants de types dérivés.</li> </ul> <p>Le programme lance, donc, une erreur quand il trouve l'attribut POINTER et il référence une variable simple.</p> <p>real, pointer :: ppi</p>	Oui
F90.NAME.GenericIntrinsic	<p>Ne pas se servir des fonctions intrinsèques spécifiques (INT,IFIX,IDINT,REAL,FLOAT,SNGL,ICHAR,CHAR,AINT,DINT,ANINT,DNINT,NINT,IDNINT,IABS,ABS,DABS,CABS,MOD,AMOD,DMOD,ISIGN,SIGN,DSIGN,IDIM,DIM,DDIM,DPROD,MAX0,AMAX1,DMAX1,AMAX0,MAX1,MIN0,AMIN1,DMIN1,AMIN0,MIN1,AIMAG,CONJG,SQRT,DSQRT,CSQRT,EXP,DEXP,CEXP,ALOG,DLOG,CLOG,ALOG10,DLOG10,SIN,DSIN,CSIN,COS,DCOS,CCOS,TAN,DTAN,ASIN,DASIN,ACOS,DACOS,ATAN,DATAN,ATAN2,DATAN2,SINH,DSINH,COSH,DCOSH,TANH,DTANH), utiliser les génériques (INT,REAL,AINT,ANINT,NINT,ABS,MOD,SIGN,DIM,MAX,MIN,SQRT,EXP,LOG,LOG10,SIN,COS,TAN,ASIN,ACOS,ATAN,ATAN2,SINH,COSH,TANH).</p> <p>resultat = AMOD (argument, diviseur)</p> <p>resultat = MOD (argument, diviseur)</p>	Oui
F90.NAME.KeyWords	<p>Les variables du code ne peuvent être nommées comme les mots clés en Fortran (ALLOCATABLE, ALLOCATE ; ASSIGN, BACKSPACE, ...). De plus les noms de fonctions doivent être différents des fonctions intrinsèques (ABS, ACHAR, ACOS, ...)</p> <p>integer, parameter :: DATA</p> <p>integer, parameter :: MY_DATA</p>	Oui
F90.PROTO.Overload	<p>La surcharge des opérateurs est interdite. Celle-ci est définie par l'instruction INTERFACE OPERATOR (symbole) et ensuite par leur utilisation.</p>	Oui
F90.REF.ARRAY	<p>Dans une expression, quand on veut représenter l'utilisation total d'un tableau, il est obligatoire de se servir de la notation ( : )</p> <p>Y = A*X + B</p> <p>Y(:) = A(:)*X + B où Y et A sont des tableaux</p>	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 35/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
F90.REF.Interface	<p>Le sous-programme appelé doit être visible</p> <pre> subroutine Pas_1 call Mouv(3.0,resul) end subroutine Pas_1 subroutine Mouv(oper0, resul) ... end subroutine Mouv end subroutine Pas_1 </pre> <p>subroutine Pas_1 call Mouv(3.0,resul) contains subroutine Mouv(oper0, resul) -&gt; subroutine Mouv(oper0, resul) ... end subroutine Mouv end subroutine Pas_1</p>	Oui
F90.REF.Label	<p>L'END doit être suivi par le type (FUNCTION, SUBROUTINE, ...) et le nom</p> <pre> function f ... end function </pre> <p>function f ... end function f</p>	Oui
F90.REF.Open	<p>Toute instruction OPEN doit avoir les paramètres FILE, STATUS, IOSTAT et POSITION</p> <pre> OPEN (UNIT=f_unit, FILE=c_args, STATUS='old', POSITION='rewind', IOSTAT=ios) </pre>	Oui
F90.REF.Variable	<p>Une même variable doit être référencée sous le même nom dans un sous-programme.</p> <pre> Call incr( i ) ... subroutine incr( j ) i = i + 1 end subroutine incr </pre> <p>call incr( i ) ... subroutine incr( j ) j = j + 1 end subroutine incr</p>	Oui
F90.TYPE.Derivate	<p>Toute déclaration de type doit être dans un module.</p>	Oui
F90.TYPE.Integer	<p>Les paramètres INTEGER doivent être suivis par l'expression SELECTED_INT_KIND.</p> <pre> Integer, parameter :: LONG </pre>	Oui

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 36/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	integer, parameter :: LONG = SELECTED_INT_KIND(5)	
F90.TYPE.Real	<p>Les paramètres REAL doivent être suivis par l'expression SELECTED_REAL_KIND.</p> <p>Real, parameter :: LONG</p> <p>real, parameter :: LONG = SELECTED_REAL_KIND(5)</p>	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 37/51
-------------------	---------------------------------------	---

### 5.2.3. SHELL

Règle	Vérification	Couverture au standard
SH.DATA.Constant	Obligation de définir les constantes en utilisant <code>'typeset -r'</code> .	Non
SH.DATA.IFS	Interdiction de modifier la variable IFS	Oui
SH.DATA.Integer	À la déclaration d'un integer doit apparaitre <code>'typeset -i'</code>	Oui
SH.DESIGN.Bash	La première ligne d'n script doit être <code># !/bin/bash</code> , <code># !/bin/ksh</code> ou <code># !/bin/false</code> .	Oui
SH.DESIGN.Exit	Interdiction pour un programme de prendre fin avant les programmes qu'il a lancés en tâches de fond.	Non
SH.DESIGN.Options	Le cas <code>getopts</code> doit être suivi par un case ou ses options sont évalués.	Oui
SH.ERR.Args	Obligation d'afficher un message particulier lorsqu'un script ne reconnaît pas une option. Ce message doit inclure le synopsis d'utilisation du script.	Non
SH.ERR.Help	Les options doivent être gérées par un <code>getopts</code> ou <code>getopt</code> suivi par un case. Une de ces options doit être <code>-h</code> ou <code>-help</code> .  Limitation : si un "h)" apparaît dans une chaîne de caractères dans le traitement du case, alors que l'option <code>-h</code> n'est pas gérée, il n'y aura pas de violation.	Oui
SH.ERR.NoPipe	Avant le premier pipe d'un script il faut avoir <code>'set -o pipefail'</code>  <u>Limitations</u> : <ul style="list-style-type: none"> <li>Un symbole " " apparaissant dans une chaîne de caractères ou dans les options d'une commande provoquera une violation, sauf suivant un <code>printf</code> ou un <code>sed</code>.</li> <li>Les utilisations du pipe qui suivent un caractère "#" ne seront pas détectées (ex : <code>grep -v ^#   grep "\$POINT_DE_MONTAGE"</code>)</li> </ul>	Oui
SH.ERR.String	Dans les cas de <code>if</code> ou <code>while</code> avec les traitements de chaînes de caractères, il faut avoir le traitement de chaînes vides, désignée par <code>[[ ]]</code>  <u>Limitation</u> :  L'appel de <code>ls</code> à l'intérieur d'une commande <code>\$(...)</code> ou <code>`...`</code> est susceptible d'être ignoré.	Oui
SH.FLOW.CheckArguments	Dans une fonction, la première instruction, sera la vérification des paramètres. L'instruction sera de la forme suivante : <code>if [ \$# -ne 0 ]</code>	Oui

CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 38/51
-------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	Limitation : Un argument d'une même fonction sera signalé autant de fois qu'il est présent dans la fonction	
SH.FLOW.CheckCodeReturn	<b>Shell : Non implémenté</b>  <i>Pour toutes les fonctions existantes et appelées dans le script, faudra de vérifier son return avec l'aide de \$#</i>  <i>La fonction CD est aussi considérée.</i>	<b>Non</b>
SH.FLOW.CheckUser	Si le script demande des droits d'administrateur, après la vérification de l'utilisateur il faut demander l'action directe de l'admin.	<b>Oui</b>
SH.INST.Basename	Ne pas se servir de \$0, utiliser 'basename \$0'	<b>Oui</b>
SH.INST.Continue	<b>Shell : Non implémenté</b>	<b>Non</b>
SH.INST.Copy	Obligation d'utiliser des arguments de même nature pour les commandes 'cp'. La nature des arguments est soit de type fichier ou soit de type répertoire.	<b>Non</b>
SH.INST.Find	L'instruction LS est interdite	<b>Oui</b>
SH.INST.GetOpts	Cette règle vérifie les options passées comme paramètre dans le script. L'application lance un warning à la dernière ligne s'il ne trouve pas le mot GETOPS dans le script. Ce celui qui permet contrôler les paramètres passes.  <u>Limitation :</u>  Après l'appel de la commande getopt, la violation de \$1 est ignoré jusqu'à la déclaration d'une nouvelle fonction dans le fichier.	<b>Partiel</b>
SH.INST.Interpreter	La première ligne d'un script doit être l'accès à l'interpréter	<b>Oui</b>
SH.INST.Keywords	Une variable ne peut pas être le nom d'un mot clé (if, while, then...)  <u>Limitation :</u>  L'appel d'option de commandes possédant un mot clef lèvent de faux-positifs (ex : l'option if de la commande dd).	<b>Oui</b>
SH.INST.Logical	Après le symbole logique && ou    seulement est permises des instructions 'echo' et 'exit'	<b>Oui</b>
SH.INST.Move	Interdiction d'écraser un fichier en utilisant la commande 'mv'.	<b>Non</b>
SH.INST.POSIX	Utiliser les commandes POSIX aux scripts. Liste dans le RNC-CNES-Q-HB-80-516  <u>Limitation :</u>	<b>Oui</b>

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 39/51
--------------------------	---------------------------------------	---

Règle	Vérification	Couverture au standard
	L'appel de la commande écho suivi d'une redirection (ex : echo &>2) peut engendrer des erreurs sur l'analyse.	
SH.INST.SetShift	Les instructions SET et SHIFT sont interdites.	Oui
SH.INST.Variables	Toute variable doit être écrite comme \${nom_variable}	Oui
SH.IO.Redirect	Les redirection non standards doivent être commentées, et l'lien doit contenir le nom explicite de la redirection	Oui
SH.MET.LimitAWK	Pour toute instruction AWK, ne pas dépasser les 5 actions (dénotées par {...} )	Oui
SH.MET.LimitSed	Pour toute instruction SED, ne pas dépasser les 5 actions (dénotées par -e, --expression, -f, ...) et dans chaque action ne pas dépasser les 5 lignes.	Oui
SH.MET.PipeLine	Toute pipeline de ligne de commandes doit être commentée à priori.	Oui
SH.REF.Export	Shell : Non implémenté	Non
SH.REF.Inheritance	Interdiction pour les scripts SHELL d'hériter des alias ou des fonctions définies par l'utilisateur et d'utiliser des commandes interactives dans un script.	Non
SH.SYNC.Signals	Shell : Non implémenté	Non

<p>CNES DNO/DA/AQ</p>	<p>Manuel Utilisateur i-Code CNES</p>	<p>Réf : DNO/DA/AQ - 2017.0002478</p> <p>Date : 14/09/2017</p> <p>Page : 40/51</p>
---------------------------	---------------------------------------	--

# 6. TABLES DES METRIQUES CALCULEES

Dans le cas des métriques, on identifie les vérifications faites pour un fichier et celles faites pour une fonction – dans le cas où elles sont effectuées

Métrique	Vérification sur fichier
MET.Nesting	<p>Nombre maximum de niveaux d'imbrications de if, switch,while, for, ... dans une fonction/méthode. Ce nombre est 0 s'il y a aucune imbrication..</p> <pre> SUBROUTINE changer_coordonnees(Deplacement, NbPoints, Points) ! ! --- Cette routine effectue une modification de coordonnees sur un tableau de poin ! --- en appliquant un déplacement sur les 3 axes x, y et z !     IMPLICIT NONE      INTEGER :: c          ! colonne     INTEGER :: l          ! ligne      INTEGER, parameter :: NbDim = 3     INTEGER, intent(in) :: NbPoints      DOUBLE PRECISION, intent(inout), dimension(NbDim,NbPoints) :: Points     DOUBLE PRECISION, intent(in), dimension(NbDim) :: Deplacement      ! On applique a chaque point une valeur de déplacement selon les 3 axes 1   do c=1, NbDim, 1 2       do l=1, NbPoints, 1             Points(c, l) = Points(c, l) + Deplacement(c)         end do     end do      END SUBROUTINE </pre> <p>NB.Imbric = 2</p>
MET.Cyclomati c	<p>C'est le nombre de décisions du code (nb if, case, while, catch...). .</p>



<p>CNES DNO/DA/AQ</p>	<p>Manuel Utilisateur i-Code CNES</p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 41/51</p>
---------------------------	---------------------------------------	--

	<pre> SUBROUTINE changer_coordonnees(Deplacement, NbPoints, Points) ! ! --- Cette routine effectue une modification de coordonnees sur un tableau de poin ! --- en appliquant un déplacement sur les 3 axes x, y et z !       IMPLICIT NONE        INTEGER :: c          ! colonne       INTEGER :: l          ! ligne        INTEGER, parameter :: NbDim = 3       INTEGER, intent(in) :: NbPoints        DOUBLE PRECISION, intent(inout), dimension(NbDim,NbPoints) :: Points       DOUBLE PRECISION, intent(in), dimension(NbDim) :: Deplacement        ! On applique a chaque point une valeur de déplacement selon les 3 axes 1      do c=1, NbDim, 1 2          do l=1, NbPoints, 1               Points(c, l) = Points(c, l) + Deplacement(c)           end do       end do        END SUBROUTINE NB.Cycomatique = 2 </pre>
<p>MET.LineOfCode</p>	<p>C'est le nombre total les lignes dans du code source du composant logiciel sauf les lignes vides et les lignes de commentaires</p>

<p>CNES DNO/DA/AQ</p>	<p>Manuel Utilisateur i-Code CNES</p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 42/51</p>
---------------------------	---------------------------------------	--

	<pre> 1  SUBROUTINE changer_coordonnees(Deplacement, NbPoints, Points) 2  ! 3  ! --- Cette routine effectue une modification de coordonnees sur un tableau de poin 4  ! --- en appliquant un déplacement sur les 3 axes x, y et z 5  ! 6  IMPLICIT NONE 7 8  INTEGER :: c          ! colonne 9  INTEGER :: l          ! ligne 10 11  INTEGER, parameter :: NbDim = 3 12  INTEGER, intent(in) :: NbPoints 13 14  DOUBLE PRECISION, intent(inout), dimension(NbDim,NbPoints) :: Points 15  DOUBLE PRECISION, intent(in), dimension(NbDim) :: Deplacement 16 17  ! On applique a chaque point une valeur de déplacement selon les 3 axes 18  do c=1, NbDim, 1 19      do l=1, NbPoints, 1 20          Points(c, l) = Points(c, l) + Deplacement(c) 21      end do 22  end do 23 24  END SUBROUTINE 25 26 . </pre> <p>NB.Line = 15</p>	
<p>MET.RatioCom ment</p>	<p>C'est la proportion de commentaires dans le code source du composant logiciel. L'entête se prend en compte dans le calcul de le taux de commentaire (tant si elle est avant ou après la définition).</p>	

<pre> 1  SUBROUTINE changer_coordonnees(Deplacement, NbPoints, Points) 1  ! 2  ! --- Cette routine effectue une modification de coordonnees sur un tableau de poin 3  ! --- en appliquant un deplacement sur les 3 axes x, y et z 2  !       IMPLICIT NONE  3      INTEGER :: c          ! colonne 4      INTEGER :: l          ! ligne  5      INTEGER, parameter :: NbDim = 3 6      INTEGER, intent(in) :: NbPoints  7      DOUBLE PRECISION, intent(inout), dimension(NbDim,NbPoints) :: Points 8      DOUBLE PRECISION, intent(in), dimension(NbDim) :: Deplacement  5  ! On applique a chaque point une valeur de deplacement selon les 3 axes 9  do c=1, NbDim, 1 10     do l=1, NbPoints, 1 11         Points(c, l) = Points(c, l) + Deplacement(c) 12     end do 13 end do 14 15 END SUBROUTINE </pre>	<p>RATE.Comment = 5 / 15 = 0.33</p>
---	-------------------------------------

## 7. MESSAGES UTILISATEUR

Il y a deux types de messages différents dans l'outil : les messages qui correspondent à des violations et les messages d'informations dans une fenêtre émergente.

### 7.1. MESSAGES DES VIOLATIONS

Les messages qui concernent les violations sont affichés dans la vue *i-Code violations* une fois que l'analyse est terminée. Ces messages donnent un peu d'information sur l'erreur remontée : variable qui provoque l'erreur, petite explication de pourquoi cette erreur.

#### 7.1.1. REGLES COMMUNES

Règle	Message
COM.DATA.DeclarationOrder	The parameters are not defined in the right order. The order shall be: in, in/out, out.
COM.DATA.FloatCompare	It's not allowed to compare float variables( " <i>variable</i> ") with equality.

<p>CNES DNO/DA/AQ</p>	<p><b>Manuel Utilisateur i-Code CNES</b></p>	<p>Réf : DNO/DA/AQ - 2017.0002478</p> <p>Date : 14/09/2017</p> <p>Page : 44/51</p>
---------------------------	--	--

Règle	Message
COM.DATA.Initialisation	The variable " <i>variable</i> " is used before being initialized.
COM.DATA.Invariant	The variable " <i>variable</i> " must be defined as constant.
COM.DATA.LoopCondition	The variable " <i>variable</i> " is modified inside the loop.
COM.DATA.NotUsed	The variable " <i>variable</i> " is declared and not used
COM.DESIGN.ActiveWait	<p>This process contains an active wait.</p> <p>SH : There is an active wait in this point.</p>
COM.DESIGN.Alloc	The resource named " <i>variable</i> " has not been allocated and deallocated in the same algorithmic level.
COM.FLOW.Abort	The keyword STOP is not allowed.
COM.FLOW.BooleanExpression	<p>Using more than five conditions in an expression is not allowed.</p> <p>SH : It is not allowed use five or more conditional expressions in the same instruction.</p>
COM.FLOW.CaseSwitch	<p>A DEFAULT case is needed in a switch case instruction.</p> <p>SH : The default case of the case switch condition is missing.</p>
COM.FLOW.CheckCodeReturn	The return code of the function "function" is not checked.
COM.FLOW.CheckUser	The user identity is not verified in the main program.
COM.FLOW.Exit	There is more than one exit in the function.
COM.FLOW.ExitLoop	There is more than one exit in the loop.
COM.FLOW.FileExistence	<p>The existences of the file "file" must be checked with the instruction INQUIRE before being opened or created.</p> <p>SH : The existence of the file " + name + " has not been checked.</p>
COM.FLOW.FilePath	It is not allowed to use directly the file name. Store the file path in a variable. Use the variable instead.
COM.FLOW.Recursion	The use of recursivity is not allowed.
COM.INST.BoolNegation	Double negation is not allowed.
COM.INST.Brace	Parentheses are needed for readability.
COM.INST.CodeComment	<p>Commented code is not allowed. It shall be suppressed.</p> <p>SH :</p>

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 45/51
--------------------------	---------------------------------------	---

Règle	Message
	<p>The keyword " + yytext() + " is used in a comment.</p> <p>A variable is assigned in a comment.</p>
COM.INST.GOTO	The keyword GOTO is not allowed.
COM.INST.Line	More than one instruction per line is not allowed.
COM.INST.LoopCondition	A loop condition shall be written with inequality (.LE.,<=, or .GT.,>=)
COM.NAME.Homonymy	Names must be unique. The name "variable" is already defined in this file.
COM.PRES.Indent	The code is not indented.
COM.PRES.LengthLine	There are more than 100 characters in this line.
COM.PROJECT.Header	<p>- No file header existing. This module/function should have a header with a brief description.</p> <p>- No file header (file name not found). This module/function should have a header with a brief description.</p> <p>- The module/function should have a header with a brief description.</p> <p>SH : The function should have a header with a brief description.</p>
COM.TYPE.Expression	Mixed types " <i>type_variable_1</i> " with " <i>type_variable_2</i> " in the same expression

### 7.1.2. FORTRAN 77

Règle	Message
F77.BLOC.Common	Unnamed COMMON is not allowed.
F77.BLOC.Else	The IF instruction shall finish with an ELSE after the last ELSE IF.
F77.BLOC.Function	When calling a function, the brackets following the function name are mandatory.
F77.BLOC.Loop	Loops shall have distinct ends.
F77.DATA.Array	The dimension of the array "variable" is not well declared. The * shall be used for the last dimension.
F77.DATA.Common	The INCLUDE instruction shall be used to reference the needed common bloc.

<p style="text-align: center;"><b>CNES</b> DNO/DA/AQ</p>	<p style="text-align: center;"><b>Manuel Utilisateur i-Code CNES</b></p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 46/51</p>
--	--	--

F77.DATA.Double	The double precision variable is not correctly initialized. It misses the character D in its declaration.
F77.DATA.Initialization	The variable "variable" shall be initialized with DATA or BLOCK DATA before its use.
F77.DATA.IO	The use of * with logical units is not allowed.
F77.DATA.LoopDo	The control variable in a loop shall be an integer.
F77.DATA.Parameter	"variable" belongs to parameter types forbidden when calling a function: a constant, an expression to be evaluated, a call to another function
F77.ERR.OpenRead	The status of OPEN/READ shall be tested with the parameter IOSTAT.
F77.INST.Assign	The instruction ASSIGN Is not allowed.
F77.INST.Dimension	The instruction DIMENSION Is not allowed.
F77.INST.Equivalence	The instruction EQUIVALENCE is not allowed.
F77.INST.Function	It misses the type declaration in FUNCTION header.
F77.INST.If	The arithmetic if is not allowed.
F77.INST.Include	The executable instruction "variable" is not allowed in the include file.
F77.INST.Pause	The instruction PAUSE is not allowed.
F77.INST.Return	The instruction RETURN(i) is not allowed.
F77.INST.Save	The instruction SAVE is only permitted for local variables
F77.MET.Line	There are more than 72 characters in this line.
F77.NAME.GenericIntrinsic	It should be used the generic name of the intrinsic function instead of "variable"
F77.NAME.Intrinsic	It is not allowed to use the name of an intrinsic function.
F77.NAME.KeyWords	The variable "variable" is a keyword in Fortran77 language.
F77.NAME.Label	The use of labels is not allowed except with the instructions FORMAT and CONTINUE.
F77.PROTO.Declaration	The function "variable" shall be declared.
F77.REF.IO	The logical entities shall be declared using a symbolic name.
F77.REF.Open	The instruction OPEN shall be called with the parameters FILE, STATUS and POSITION.

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 47/51
--------------------------	---------------------------------------	---

F77.REF.Parameter	It is not allowed to provide as a parameter the variables of an accessible bloc COMMON. The variable "variable" is used in a wrong way.
F77.TYPE.Basic	"variable" is not a basic type. Basic types are INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL and CHARACTER.
F77.TYPE.Hollerith	Type Hollerith is not allowed. "variable" shall be a CHARACTER.

### 7.1.3. FORTRAN 90

Règle	Message
F90.BLOC.File	The file "variable" is not correctly closed.
F90.DATA.Array	The dimension's array must be declared as parameters' function.
F90.DATA.ArrayAccess	Array " variable1" initialized using other array named "variable2 " with repeated values.
F90.DATA.Constant	The constants shall be declared and initialized in a module.
F90.DATA.ConstantFloat	Float constant "variable" shall be declared using the subtype_parameter: <name>_<subtype_parameter>
F90.DATA.Declaration	The variable must be declared.  The sequence IMPLICIT NONE must be declared after the method.
F90.DATA.Float	It is not allowed to use the format * for reals like "variable".
F90.DATA.Parameter	It misses the use of intrinsic function SELECTED_REAL_KIND or SELECTED_INT_KIND for the subtype specification.
F90.DESIGN.Include	Is it possible to use a module instead of this inclusion?
F90.DESIGN.Interface	Interface Module shall only contain: INTERFACE, USE, IMPLICIT instructions as well as PRIVATE or PUBLIC declaration.
F90.DESIGN.IO	The value of the logic unity should be a integer or a variable initialised directly.
F90.DESIGN.Obsolete	The instruction calculated GOTO is not allowed.  The instruction PAUSE is not allowed.  The alternate return statement is not allowed.  There is a branch on an END IF statement. It is not allowed.  The use of CHARACTER* is not allowed.

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 48/51
--------------------------	---------------------------------------	---

	<p>The instruction HOLLERITH is not allowed inside FORMAT. Error in "variable" used.</p> <p>The instruction ASSIGN contains the label for the FORMAT instruction.</p> <p>Arithmetical IF is not allowed.</p> <p>A DO loop shall end with END DO.</p> <p>The variable "variable" is a real used in a do loop. Use only INTEGER.</p> <p>Each loop shall have its own END DO. Shared END DO is forbidden.</p>
F90.ERR.Allocate	The status of the ALLOCATE or DEALLOCATE instruction is not checked
F90.ERR.OpenRead	<p>- There is no parameter IOSTAT in the OPEN/READ instruction.</p> <p>- The return of IOSTAT is no checked in the OPEN/READ instruction.</p>
F90.INST.Associated	The pointer « variable » is not set to null before the use of the instruction ASSOCIATED.
F90.INST.Entry	The instruction ENTRY is not allowed.
F90.INST.Equivalence	The instruction EQUIVALENCE is not allowed.
F90.INST.If	Logical IF (without THEN and ENDIF) is only allowed with EXIT, CYCLE, GOTO, RETURN statements.
F90.INST.Intent	It misses the attribute INTENT for the parameter "variable"
F90.INST.Nullify	It misses the instruction NULLIFY after the DEALLOCATION of "variable".
F90.INST.Only	The instruction ONLY must be preceded by a comment.
F90.INST.Operator	The symbolic notation (==, /=, <=, <, >=, >) must be used instead of (.EQ., .NE., .LT., .LE., .GT., .GE.). Error in "variable".
F90.INST.Pointer	This use of POINTER is not allowed.
F90.NAME.GenericIntrinsic	Use the generic name of the intrinsic functions instead of "variable".
F90.NAME.KeyWords	The variable "variable" is a keyword in Fortran90 language.
F90.PROTO.Overload	Overloading operator is not allowed. Overload of "variable"
F90.REF.ARRAY	It should be used the notation(:) to specify the entire use of the arrays: "list_variables".



CNES DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 49/51
-------------------	---------------------------------------	---

F90.REF.Interface	The function "function" is not visible in this point.
F90.REF.Label	It misses the name of the subprogram. It must finish with END TYPE_PROGRAM NAME.
F90.REF.Open	It misses one or more parameters In OPEN instruction. Mandatory parameters are FILE, STATUS, IOSTAT, POSITION.
F90.REF.Variable	The variable "variable" is used with different names inside the subprogram.
F90.TYPE.Derivate	The variable " " must be defined inside the module structure.
F90.TYPE.Integer	It misses the declaration SELECTED_INT_KIND in the initialisation of "variables"
F90.TYPE.Real	It misses the declaration SELECTED_REAL_KIND in the initialisation of "variables"

#### 7.1.4. SHELL

Règle	Message
SH.DATA.IFS	The environment variable IFS can't be modified.
SH.DATA.Integer	The integer variables must be defined using the typeset -i declaration.
SH.DESIGN.Bash	The first line must declare the interpreter (/bin/bash, /bin/ksh or /bin/false)
SH.DESIGN.Options	It is mandatory to use getopts and getopt and to provide the -h, -help, -v and -version options at least.
SH.ERR.Help	The help option (-h or --help) must be implemented.
SH.ERR.NoPipe	When the pipe is used in the script the option set -o pipefail is mandatory.
SH.ERR.String	The empty strings must be taken into account
SH.FLOW.CheckArguments	The number of parameters received has not been checked.
<i>SH.FLOW.CheckCodeReturn</i>	<i>The function's return function_name has not been checked.</i>
SH.FLOW.CheckUser	The user has not been checked.
SH.INST.Basename	The use of the keyword basename before \$0 is mandatory.
<i>SH.INST.Continue</i>	<i>The keyword CONTINUE is not allowed.</i>
SH.INST.Find	The use of LS is not allowed. Use FIND instead.
SH.INST.GetOpts	It is mandatory to use getopts & getopt in the script.

<b>CNES</b> DNO/DA/AQ	<b>Manuel Utilisateur i-Code CNES</b>	Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 50/51
--------------------------	---------------------------------------	---

Règle	Message
SH.INST.Interpreter	The first line must declare the interpreter
SH.INST.Keywords	The keywords <i>variable</i> cannot be used as a variable.
SH.INST.Logical	The abbreviation    and && must be followed only by ECHO or EXIT.
SH.INST.POSIX	The keyword <i>POSIX_word</i> is not allowed.
SH.INST.SetShift	The keyword SET/SHIFT is not allowed.
SH.INST.Variables	The variable <i>variable_name</i> is not correctly declared (must be declared using \${ } or " " notation )
SH.IO.Redirect	Thenon-standard redirection must be preceded by a comment.
SH.MET.LimitAWK	The AWK expression has more than 5 actions
SH.MET.LimitSed	The SED expression has more than 5 actions/lines
SH.MET.PipeLine	Every pipeline must be preceded by a comment.
<i>SH.REF.Export</i>	<i>The keyword EXPORT is no allowed.</i>
<i>SH.SYNC.Signals</i>	<i>The keyword TRAP must be followed by a variable, not an integer.</i>

<p style="text-align: center;"><b>CNES</b> DNO/DA/AQ</p>	<p style="text-align: center;"><b>Manuel Utilisateur i-Code CNES</b></p>	<p>Réf : DNO/DA/AQ - 2017.0002478 Date : 14/09/2017 Page : 51/51</p>
--	--	--

## 8. LIMITATIONS

En cas de problème lié à l'exécution d'i-Code CNES, un message redirige vers la plateforme des issues **GitHub** de i-Code CNES à cette adresse : <https://github.com/dupuisa/i-CodeCNES/issues/>.

Si la problématique rencontrée n'est pas renseignée dans les limitation ci-dessous, cette plateforme permet de saisir des issues afin que les problèmes puissent être corrigés.

Les limitations suivantes sont présentes dans i-Code CNES et peuvent altérer l'expérience utilisateur.

### 8.1. POWERSHELL NE PERMET PAS LE LANCEMENT D'ANALYSE

Sous Windows, l'invite de commande Powershell ne permet pas de lancer des analyses. Il faut utiliser l'invite de commande cmd.exe (Démarrer > Exécuter > cmd.exe).

Issue : <https://github.com/dupuisa/i-CodeCNES/issues/73>

### 8.2. SHELL : LES CHAINES DE CARACTERES DOIVENT ETRE DELIMITEES PAR DES GUILLEMETS

Les langages shell sont très permissifs contrairement à la plupart des autres langages, il est souvent possible en paramètres de commandes d'utiliser des chaînes de caractères sans les entourer de guillemets. Néanmoins, l'analyseur d'i-Code CNES ne sait pas interpréter ces chaînes de caractères.

Issue: <https://github.com/dupuisa/i-CodeCNES/issues/31>

♦♦♦♦ FIN DU DOCUMENT ♦♦♦♦