

A decorative border composed of a grid of colored squares surrounds the central text. The colors include blue, orange, light blue, light orange, grey, white, teal, and dark blue.

NDepend en action

Nestor K.
Ing. Log.,Msc.



INTRODUCTION



PRINCIPAUX AXES



DÉMO



CONCLUSION



QUESTIONS



→ Créé en 2004

→ Commercialisé depuis 2007

→ Clientèle (> 6000)

→ Force

- Analyse statique de code .NET
- Intégré à Visual Studio
- Intégré à TFS 2017/2018
- Supporte TeamCity, SonarQube



 Diagrammes

 Métriques applicatives

 Assurance qualité

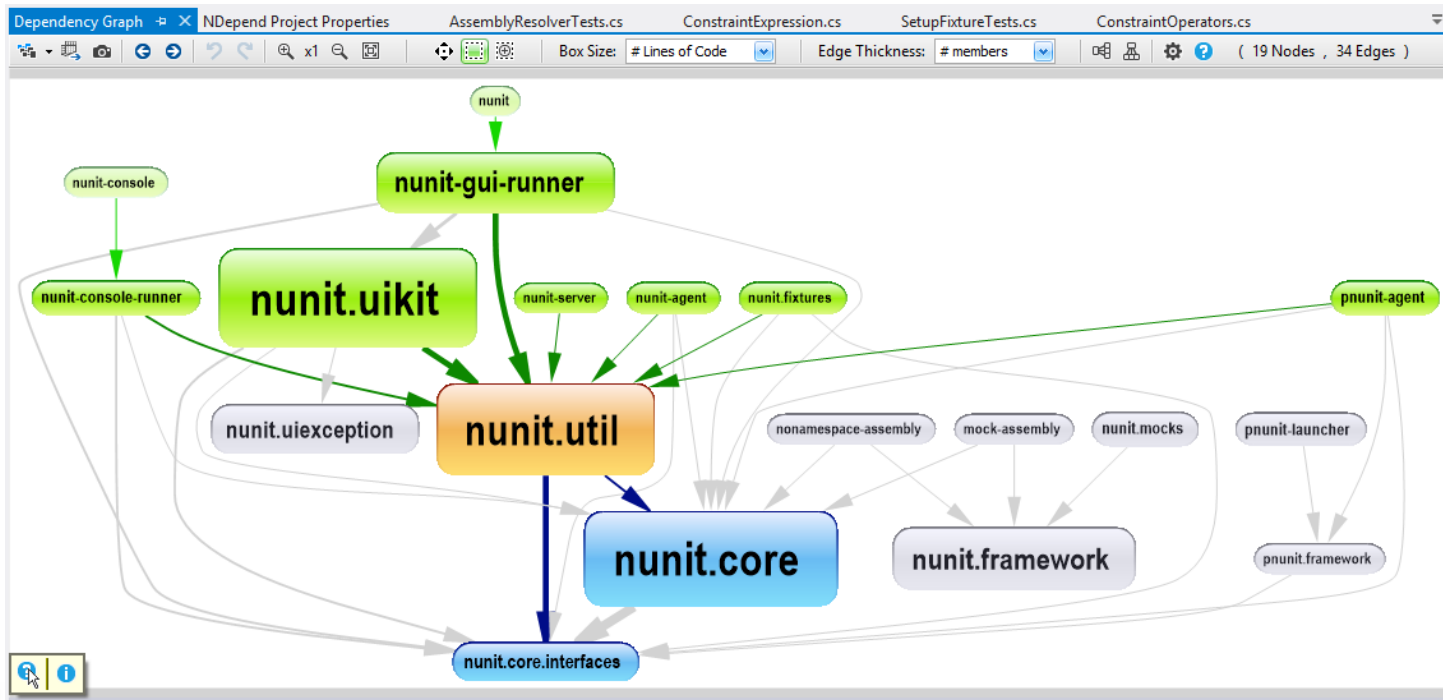
 Règles organiques

 Couverture de tests

 Dette technique

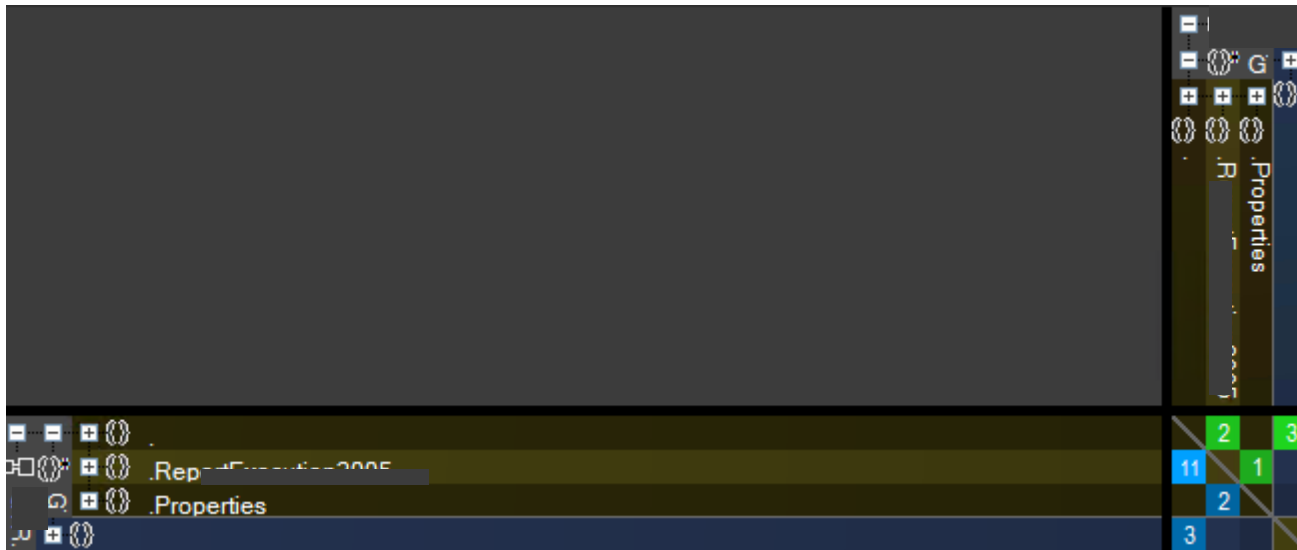


Graphe de dépendance





Matrice de dépendance



	2	3
11	1	2
3	2	3



Matrice de dépendance (suite)

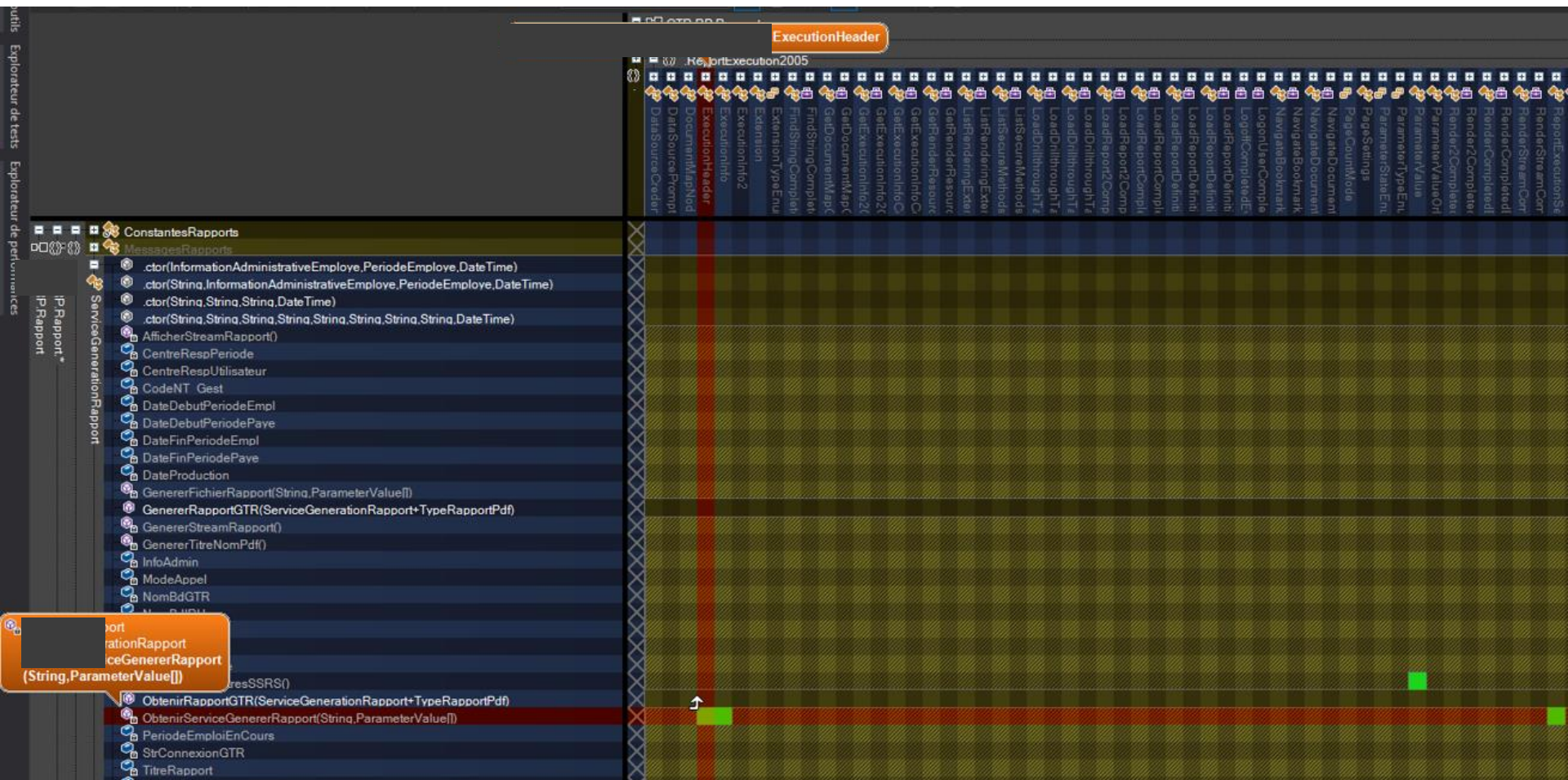




Diagramme d'abstraction / instabilité

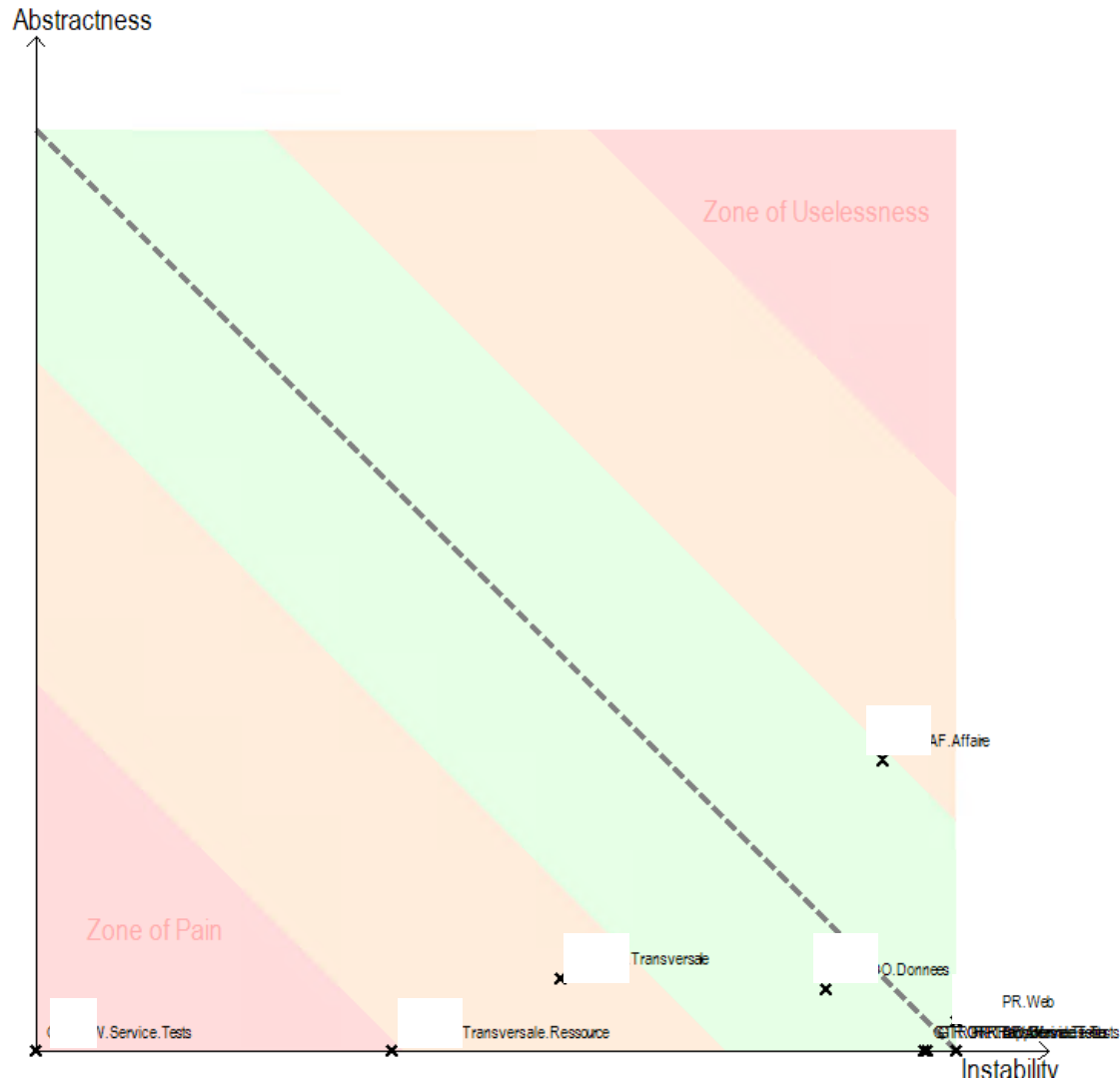




Tableau de bord

Choose Baseline **same** 1d 2d 3d 4d 5d 7d 14d 30d 60d 90d any define none

Lines of Code

11 514 no diff

1 041 (NotMyCode) no diff

Estimated Dev Effort 339d no diff

Debt

7.19% no diff

Rating **B** 7d 3h effort to reach **A**

Debt 24d no diff

The technical-debt is incomplete because no coverage data specified.

[Explore Debt](#)

Quality Gates

		2
⚠	Warn	0
◇	Pass	6

Types

407 no diff

11 Assemblies no diff

58 Namespaces no diff

2 120 Methods no diff

410 Fields no diff

357 Source Files no diff

1 330 Third-Party Elements no diff

Coverage

N/A because no coverage data specified

[Import Code Coverage Data](#)

Rules

🚫	violated	0
○	Ok	68

Comment

48.57% no diff

10 872 Lines of Comment no diff

Method Complexity

76 Max no diff

1.74 Average no diff

Issues

All 974

🏠	Medium	675
🏠	Low	188

[Group issues by rules](#)





Métrique du code (12)

- Nombre de lignes de code
- Nombre de ligne de commentaires de code
- Pourcentage de commentaires
- Quantité d'instruction d'IL
- Nombre d'assemblies
- Nombre d'espaces de nom
- Nombre de types
- Nombre de méthodes
- Nombre de champs
- Pourcentage de couverture de test
- Nombre de lignes de code couvert
- Nombre de lignes de code non couvert

Lines of Code

11 514  no diff

1 041 (NotMyCode)  no diff

Estimated Dev Effort 339d  no diff

Comment

48.57%  no diff

10 872 Lines of Comment  no diff



Métrique sur les assemblies (18)

- Couplage afférant
- Couplage efférent
- Cohésion relationnelle
- Instabilité
- Abstraction

Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relation Cohesion
						77.17	-	19	12	0.14
						58.27	-	110	146	1.22
						40.89	-	26	157	2.33
						41.93	-	18	207	1.37
						53.55	-	2	55	0.57
						35.33	-	0	116	0.23
						70.02	-	0	107	0.25
						42.17	-	3	93	1.6
						37.72	-	0	273	1.05

Métrique sur les namespace(13)

- Couplage afférent au niveau de l'espace de noms
- Couplage efférent au niveau de l'espace de noms



Métrique sur les types (22)

- Complexité cyclomatique
- Nombre d'interfaces implémentées
- Profondeur d'héritage
- Taille de l'instance
- Couplage afférent au niveau du type (Ca)
- Couplage efférent au niveau du type (Ce)
- Manque de cohésion des méthodes (LCOM)

Métrique sur les méthodes (19)

- Couplage afférent au niveau du type (Ca)
- Couplage efférent au niveau du type (Ce)
- Nombre de paramètres / variables
- Nombre de surcharge
- Pourcentage de couverture

Métrique sur les champs (2)

- Couplage afférent au niveau du type (Ca)
- Taille de du champ



9 0 2


Quality Gates

2 Quality Gate(s) that Fail

- ▶ **Critical Rules Violated**
- ▶ **Debt Rating per Namespace**

9 Quality Gate(s) that Pass

- ▶ Percentage Coverage
- ▶ Percentage Coverage on New Code
- ▶ Percentage Coverage on Refactored Code
- ▶ Blocker Issues
- ▶ Critical Issues
- ▶ New Blocker / Critical / High Issues
- ▶ Percentage Debt
- ▶ New Debt since Baseline
- ▶ New Annual Interest since Baseline

 *Quality Gate Fail:* Critical Rules Violated

9 rules matched

	9 rules	issues	Full Name
Avoid types too big		12 issues	Rule
Avoid methods too big, too complex		9 issues	Rule
Avoid methods with too many parameters		5 issues	Rule
Do not hide base class methods		1 issue	Rule
Avoid namespaces mutually dependent		10 issues	Rule
Avoid non-readonly static fields		6 issues	Rule
Attribute class name should be suffixed with 'Attribute'		3 issues	Rule
Avoid having different types with same name		9 issues	Rule
Don't call your method Dispose		1 issue	Rule



1 4 3

Code Smells

- 3 Critical Rule(s) violated
 - ▶ **Avoid types too big**
 - ▶ **Avoid methods too big, too complex**
 - ▶ **Avoid methods with too many parameters**
- 4 Rule(s) violated
 - ▶ Avoid types with too many methods
 - ▶ Avoid types with too many fields
 - ▶ Avoid methods potentially poorly commented
 - ▶ Avoid types with poor cohesion

3 10 1

Object Oriented Design

- 1 Critical Rule(s) violated
 - ▶ **Do not hide base class methods**
- 10 Rule(s) violated
 - ▶ Avoid interfaces too big
 - ▶ Class shouldn't be too deep in inheritance tree
 - ▶ Class with no descendant should be sealed if possible
 - ▶ Overrides of Method() should call base.Method()
 - ▶ A stateless class or structure might be turned into a static type
 - ▶ Non-static classes should be instantiated or turned to static
 - ▶ Methods should be declared static if possible
 - ▶ Constructor should not call a virtual method
 - ▶ Don't assign static fields from instance methods
 - ▶ Avoid empty interfaces

1 method

baseMethodsHidden

Debt

Severity

Full Name

ListeArgumentsPourDescriptionErreur

1 method

10min

High

:intite .HoraireTravailEffort .AffichageJourPeriod

()

()



6 6 0

Design

6 Rule(s) violated

- ▶ Types with disposable instance fields must be disposable
- ▶ Avoid namespaces with few types
- ▶ Nested types should not be visible
- ▶ Instances size shouldn't be too big
- ▶ Attribute classes should be sealed
- ▶ Do implement methods that throw NotImplementedException

3 1 0

Dead Code

1 Rule(s) violated

- ▶ Potentially Dead Methods

2 5 1

Architecture

1 Critical Rule(s) violated

- ▶ **Avoid namespaces mutually dependent**

5 Rule(s) violated

- ▶ Avoid namespaces dependency cycles
- ▶ UI layer shouldn't use directly DB types
- ▶ UI layer shouldn't use directly DAL layer
- ▶ Assemblies with poor cohesion (RelationalCohesion)
- ▶ Namespaces with poor cohesion (RelationalCohesion)

7 3 1

Immutability

1 Critical Rule(s) violated

- ▶ **Avoid non-readonly static fields**

3 Rule(s) violated

- ▶ Fields should be marked as ReadOnly when possible
- ▶ A field must not be assigned from outside its parent hierarchy ty
- ▶ Don't assign a field from many methods



.NET Framework Usage



Naming Conventions

3 Critical Rule(s) violated

- ▶ **Attribute class name should be suffixed with 'Attribute'**
- ▶ **Avoid having different types with same name**
- ▶ **Don't call your method Dispose**

8 Rule(s) violated

- ▶ Instance fields naming convention
- ▶ Static fields naming convention
- ▶ Abstract base class should be suffixed with 'Base'
- ▶ Methods name should begin with an Upper character
- ▶ Avoid methods with name too long
- ▶ Avoid prefixing type name with parent namespace name
- ▶ Avoid naming types and namespaces with the same identifier
- ▶ Properties and fields that represent a collection of items should be named Items.



System

3 Rule(s) violated

- ▶ Mark attributes with AttributeUsageAttribute
- ▶ Do not raise too general exception types
- ▶ Use integral or string argument for indexers



Visibility

5 Rule(s) violated

- ▶ Methods that could have a lower visibility
- ▶ Avoid publicly visible constant fields
- ▶ Fields should be declared as private
- ▶ Constructors of abstract classes should be declared as protected or private
- ▶ Avoid public methods not publicly visible



9 0 2

Quality Gates

2 Quality Gate(s) that Fail


- ▶ **Critical Rules Violated**
- ▶ **Debt Rating per Namespace**

9 Quality Gate(s) that Pass

- ▶ Percentage Coverage
- ▶ Percentage Coverage on New Code
- ▶ Percentage Coverage on Refactored Code
- ▶ Blocker Issues
- ▶ Critical Issues
- ▶ New Blocker / Critical / High Issues
- ▶ Percentage Debt
- ▶ New Debt since Baseline
- ▶ New Annual Interest since Baseline



- ✓ Notation de la dette
- ✓ Effort en j/p (h/p)
- ✓ Nombre d'anomalies liées à la dette







 *Quality Gate Fail:* Debt Rating per Namespace

9 namespaces matched

9 namespaces	debtRating	debtRatio	devTimeInManDay	debtInManDay	issues	Full Name
Transversale	D	20.73	1d 7h	3h 17min	8 issues	e
Transversale.Hash	D	24.68	1d 4h	3h 9min	8 issues	e.Hash
Transversale.Validation.Unitaire	D	24.2	1h 35min	23min	5 issues	e.Validatic
Transversale.Entite.Constantes	D	33.28	34min	11min	5 issues	e.Entite.C terface odele.Par aces s ests.Mock



DÉMO

-  Compatibilité avec TFS 2010 à 2018, CruiseControl, SonarQube, TeamCity, Jenkins, FinalBuilder
-  Licence à vie et moins dispendieuse que ses concurrents
-  Œuvrer pour le moyen et long terme
-  Robustesse du code
-  Maintenabilité du code
-  Pérennisation du système

 Code Javascript et variantes

