

# LOG8430 : Qualité de Conception

# Aujourd'hui

LOG2410

Conception  
OO (SOLID)

Patrons de  
Conception

LOG8371

LOG3430

Qualité de la  
conception

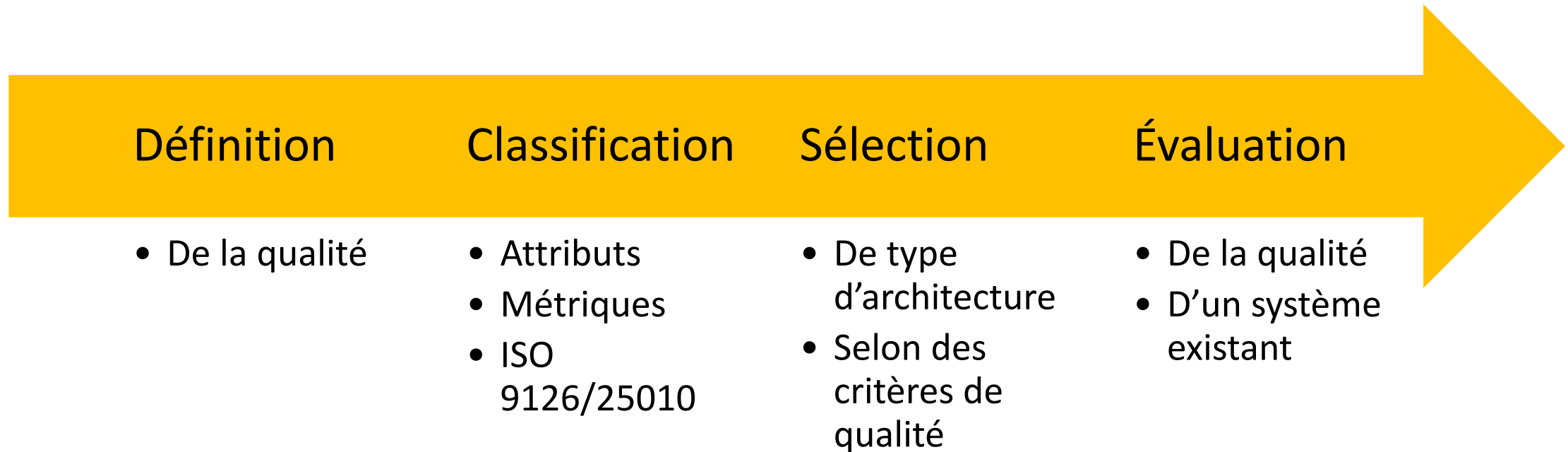
Styles et Patrons d'Architectures Distribuées

Cadriciels

Écosystèmes

Plugiciels

# Agenda



# Après cette séance...

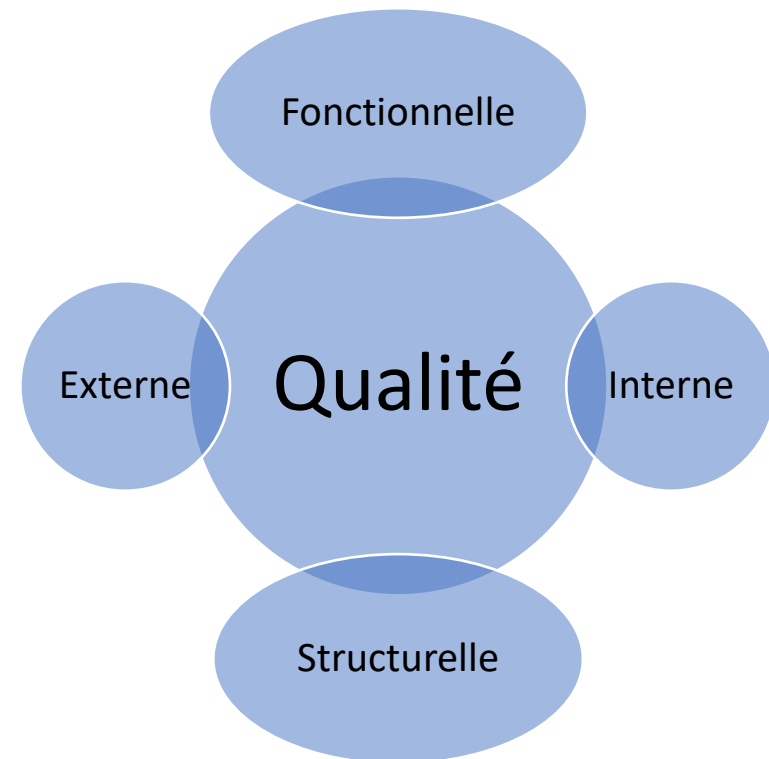
- Vous pourrez reconnaître et prioriser les attributs de qualité selon les exigences du système.
- Vous apprécierez l'importance d'une haute qualité et de maintenir ce niveau de qualité.
- Vous pourrez mesurer la qualité pour évaluer les logiciels et assurer leur qualité, et pour identifier les instances de mauvaise qualité (en combinaison avec le prochain cours).
- Vous acquerez une partie des compétences nécessaires pour l'assurance de la qualité (QA).

# Qu'est-ce que la qualité de la conception?

...beaucoup de choses!

# Définition

- En fait, la qualité inclut plusieurs notions et concepts, qui sont utilisés ensemble pour déterminer et confirmer que le logiciel fonctionne comme il faut et comme désiré.
- On va se concentrer sur la qualité interne structurelle.
- **La qualité est un critère fondamental pour la sélection d'une architecture et pour la conception du logiciel.**



# Divers perspectives de la qualité

- La qualité pour l'utilisateur : Le logiciel est utile et il fonctionne comme prévu dans le contexte d'utilisation ?
- La qualité selon la valeur : Le logiciel est évalué basé sur la valeur qu'il produit.
- La qualité du produit : La qualité est mesurée selon les caractéristiques du produit.
- La qualité pour le fabricant : Le logiciel est évalué selon sa conformité aux exigences.

# Divers perspectives de la qualité

- La qualité pour l'utilisateur : Le logiciel est utile et il fonctionne comme prévu dans le contexte d'usage.
- La qualité selon la valeur : Le logiciel est évalué basé sur la valeur qu'il produit.
- La qualité du produit : La qualité est mesurée selon les caractéristiques du produit.
- La qualité pour le fabricant : Le logiciel est évalué selon sa conformité aux exigences.



# Quelle est la meilleure?



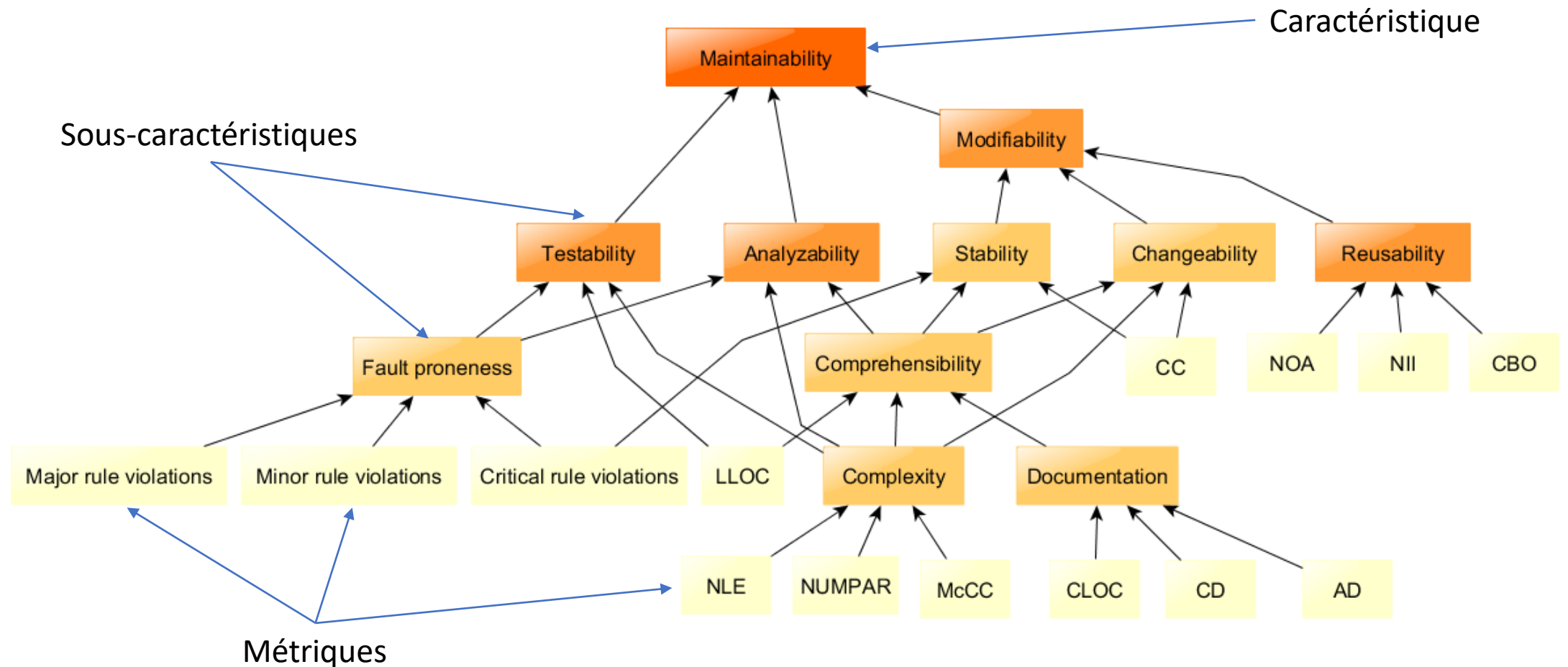
# Avant de comparer/évaluer la qualité

- On a besoin de :
- **Un modèle**
  - Quels sont les critères selon lesquels on va évaluer le logiciel?
  - Quelles sont nos priorités pour la conception, l'implémentation et l'utilisation du logiciel?
- **Des métriques**
  - **Internes**, pendant le développement (mesurées de façon statique)
  - Externes, pendant l'exécution (mesurées de façon dynamique)
  - À l'usage, pendant l'utilisation (mesurées par des humains)
- **Une procédure**
  - Concevoir, planifier, exécuter l'évaluation.
  - Définir le cadre de l'interprétation des résultats.

# Le modèle ISO/IEC 9126 (le vieux standard)

Fonctionnalité	Fiabilité	Utilisabilité	Efficacité	Maintenabilité	Portabilité
<ul style="list-style-type: none"> <li>• Pertinence</li> <li>• Précision</li> <li>• Interopérabilité</li> <li>• Sécurité</li> <li>• Conformité</li> </ul>	<ul style="list-style-type: none"> <li>• Maturité</li> <li>• Tolérance aux pannes</li> <li>• Récupérabilité</li> </ul>	<ul style="list-style-type: none"> <li>• Compréhensibilité</li> <li>• Facilité d'apprentissage</li> <li>• Opérabilité</li> <li>• Attraction</li> </ul>	<ul style="list-style-type: none"> <li>• Utilisation du temps</li> <li>• Utilisation des ressources</li> </ul>	<ul style="list-style-type: none"> <li>• Analysabilité</li> <li>• Changeabilité</li> <li>• Stabilité</li> <li>• Testabilité</li> </ul>	<ul style="list-style-type: none"> <li>• Adaptabilité</li> <li>• Installabilité</li> <li>• Coexistence</li> <li>• Remplaçabilité</li> </ul>

# (Demi-)Évolution



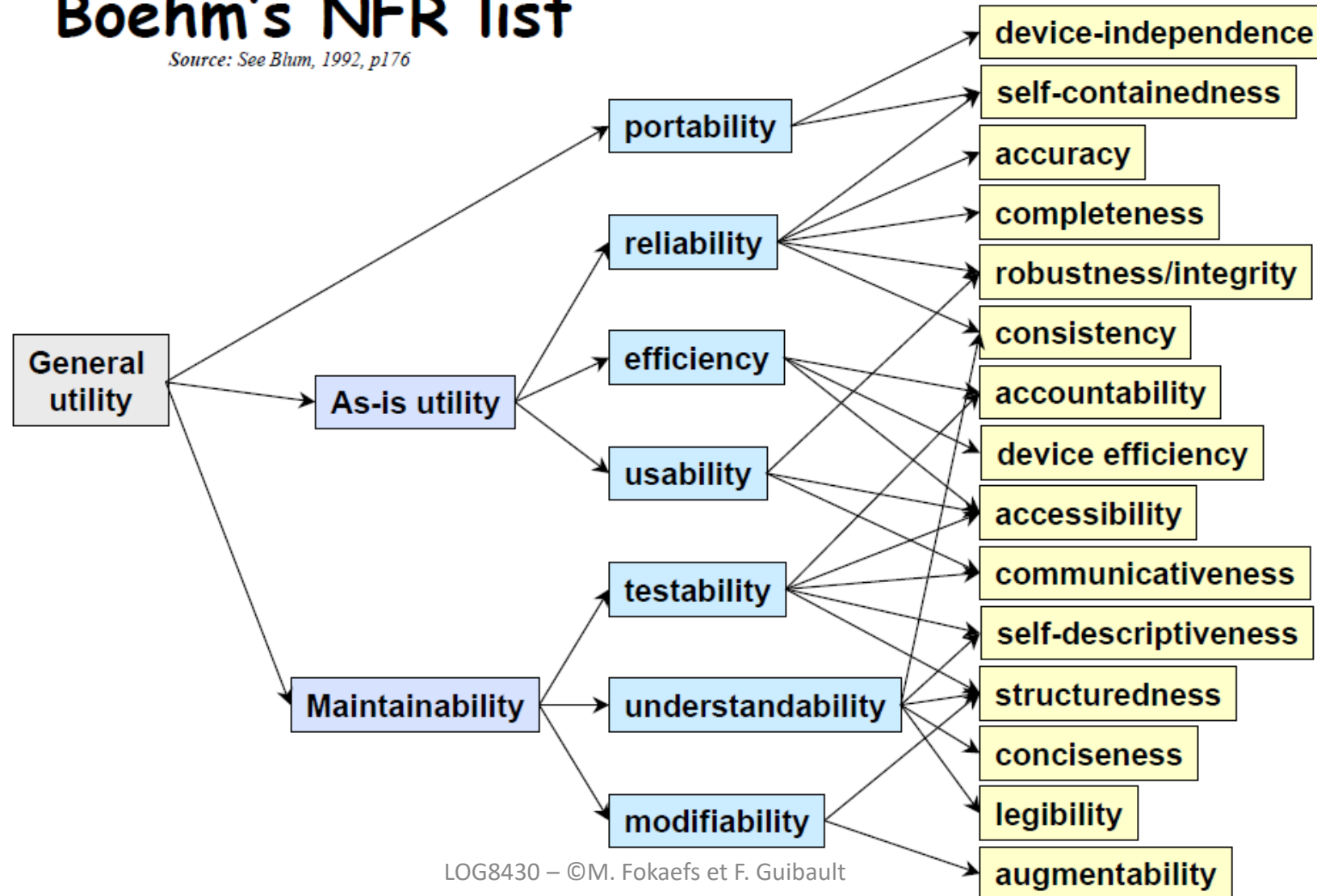
# Le modèle ISO/IEC 25010:2011

Pertinence Fonctionnelle	Fiabilité	Utilisabilité	Efficacité	Maintenabilité	Portabilité	Sécurité	Compatibilité
<ul style="list-style-type: none"> <li>• Complétude</li> <li>• Correction</li> <li>• Adéquation</li> </ul>	<ul style="list-style-type: none"> <li>• Maturité</li> <li>• Disponibilité</li> <li>• Tolérance aux pannes</li> <li>• Récupérabilité</li> </ul>	<ul style="list-style-type: none"> <li>• Accessibilité</li> <li>• Facilité d'apprentissage</li> <li>• Opérabilité</li> <li>• Esthétique d'IU</li> <li>• Protection contre les fautes d'utilisateur</li> <li>• Facilité de reconnaissance</li> </ul>	<ul style="list-style-type: none"> <li>• Utilisation du temps</li> <li>• Utilisation des ressources</li> <li>• Capacité</li> </ul>	<ul style="list-style-type: none"> <li>• Modularité</li> <li>• Réutilisabilité</li> <li>• Analysabilité</li> <li>• Possibilité de modification</li> <li>• Testabilité</li> </ul>	<ul style="list-style-type: none"> <li>• Adaptabilité</li> <li>• Installabilité</li> <li>• Remplaçabilité</li> </ul>	<ul style="list-style-type: none"> <li>• Intégrité</li> <li>• Confidentialité</li> <li>• Responsabilité</li> <li>• Authenticité</li> <li>• Non-répudiation</li> </ul>	<ul style="list-style-type: none"> <li>• Coexistence</li> <li>• Interopérabilité</li> </ul>

# Autres modèles : Boehm

## Boehm's NFR list

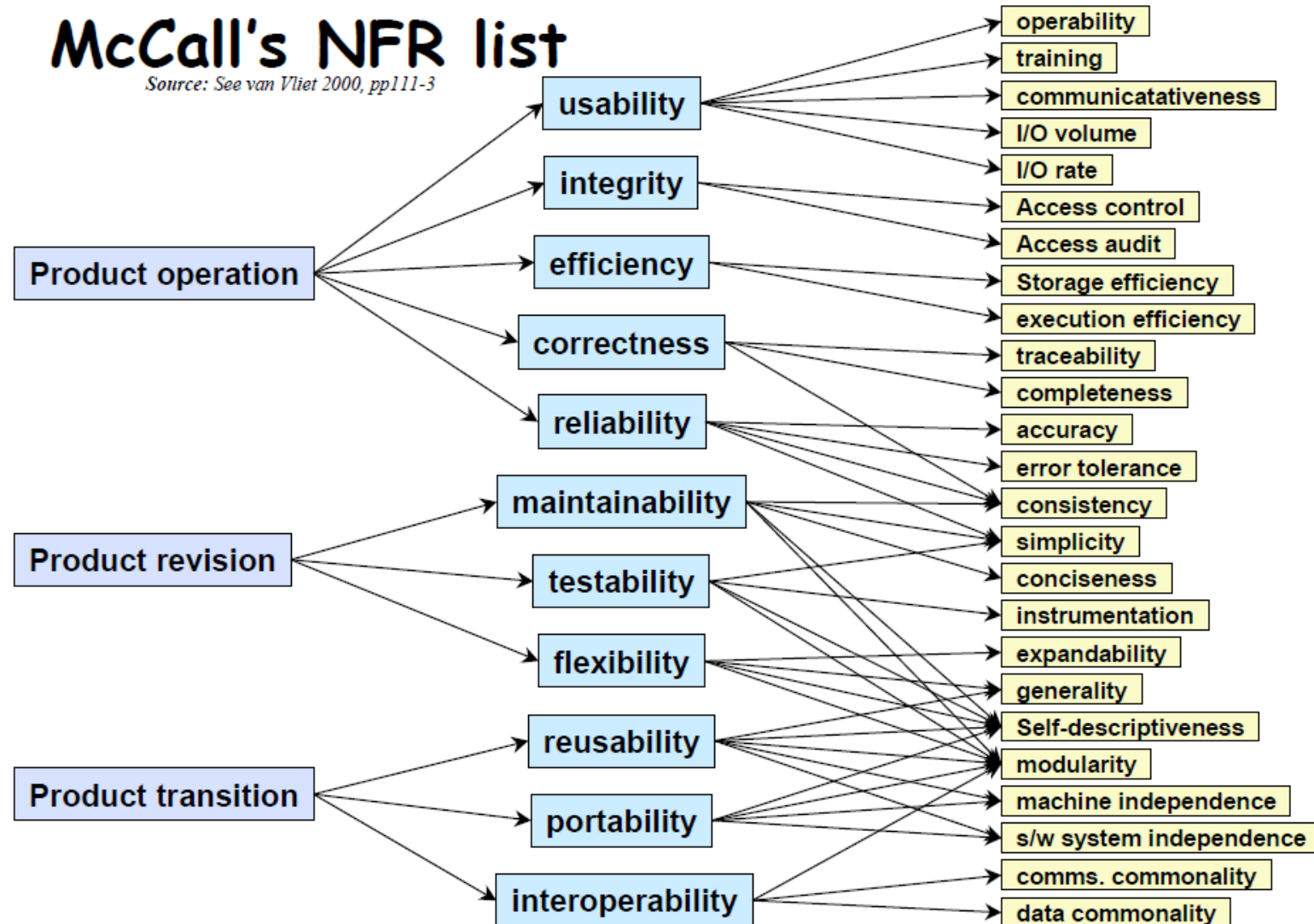
Source: See Blum, 1992, p176



# Autres modèles : McCall

## McCall's NFR list

Source: See van Vliet 2000, pp111-3





# Est-ce qu'on peut mesurer les critères?



# Les Métriques

- Taille
  - Lignes du code (*LOC*)
- Complexité de l'interface
  - Nombre d'attributs et de méthodes (*SIZE2*)
  - Nombre de méthodes locales (*NOM*)
- Complexité structurelle
  - Complexité cyclomatique de McCabe (*CC*)
  - Nombre pondéré de méthodes (*WMC*)
  - Réponse pour une classe (*RFC*)
- Héritage
  - Profondeur de l'arbre d'héritage (*DIT*)
  - Nombre d'enfants (*NOC*)

# Les Métriques (suite)

- Couplage
  - Couplage afférent ( $Ca$ )
  - Couplage entre les objets ( $CBO$ )
  - Autres...
- Cohésion
  - Manque de cohésion entre les méthodes ( $LCOM$ )
  - Cohésion de classe serrée ( $TCC$ )
  - Autres...
- Documentation
  - Manque de documentation ( $LOD$ )

# Taille

- LOC : Elle compte le nombre des caractères de saut de ligne dans une méthode, une classe, une unité de compilation.
  - Il y a plusieurs variantes : lignes du code source (SLOC), lignes du code sauf la documentation et les lignes blanches.
- SIZE2 : Elle compte le nombre d'attributs et de méthodes d'une classe.
- NOM : Elle compte le nombre de méthodes déclarées dans une classe, pas les méthodes dérivées.
- Des valeurs absolues.
- Quand la LOC, la SIZE2 et la NOM augmentent :
  - Compréhensibilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité, Interopérabilité, Sécurité, Tolérance aux pannes, Récupérabilité

# Complexité


- CC : Elle compte le nombre de chemins indépendants dans un morceau de code.
  - Définition1 :  $CC = E - V + 2 * P$ , E=arêtes, V=nœuds, P=composantes connectés
  - Définition2 : compte le nombre d'instructions de branchement (if, while, for, do, switch), les opérandes logiques (||, &&) plus 1 (définition d'Eclipse Metrics et STAN).
- WMC : Elle compte la somme d'une métrique pour toutes les méthodes d'une classe. La métrique peut être la complexité, la LOC ou aucune (unweighted).
  - Définition :  $WMC = \sum_{i=1}^n CC_i$
- RFC : Elle compte le nombre de méthodes publiques d'une classes et toutes les méthodes accédées directement par ces méthodes publiques.
- Des valeurs absolues.
- Quand la CC, la WMC et la RFC augmentent :
  - Compréhensibilité, Opérabilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité, Interopérabilité, Sécurité, Tolérance aux pannes, Récupérabilité

# Héritage

- DIT : Elle calcule la longueur maximale d'une classe dérivée à une classe de base dans la structure d'héritage du système.
- NOC : Elle compte le nombre de classes immédiatement dérivées d'une classe de base.
- Des valeurs absolues.
- Quand la **DIT** augment :
  - Compréhensibilité, Opérabilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité, Interopérabilité, Sécurité, Tolérance aux pannes, Récupérabilité
- Quand la **NOC** augment :
  - Compréhensibilité, Opérabilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité,

# Couplage (entre deux modules A et B)

Degré du couplage



Type	Caractéristiques	Désirabilité
Données	A et B utilisent des données partagées	Haute
Structure des données	A et B utilisent des structures de données partagées	OK
Contrôle	A utilise des méthodes de B	Nécessaire
Externe	A et B utilisent un « outil » commun imposé à l'externe. (format de données, appareil, protocole de communication)	Mal
Commun	A et B utilisent les mêmes variables globales.	Mal
Contenu	A utilise le code de B directement ou change l'état de B directement.	Très Mal!

# Métriques de Couplage

- CBO : Elle compte le nombre de classes qui sont « couplées » avec une classe. Deux classes A et B sont couplées, si la classe B utilise des méthodes de la classe A.
- Ca : Elle compte le nombre de classes externes qui sont couplées (par la définition de la CBO) avec des classes d'un paquet par rapport au couplage d'entrée (c.-à-d. les classes externes utilisent des méthodes des classes internes).
- Des valeurs absolues.
- Quand la CBO augment :
  - Compréhensibilité, Opérabilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Interopérabilité, Sécurité, Tolérance aux pannes, Récupérabilité  
capacité des systèmes à opérer ensemble
- Quand la Ca augment :
  - Opérabilité, Attraction, Facilité d'apprentissage, Stabilité, Remplaçabilité, Interopérabilité, Sécurité, Tolérance aux pannes

# Cohésion (entre les parties d'un module)

Type	Caractéristiques	Désirabilité
Données	Toutes font partie d'une abstraction des données	Très Haute
Fonctionnelle	Toutes résolvent le même problème.	Haute
Séquentielle	La sortie d'une est l'entrée d'une autre.	OK
Communication	Des opérations qui utilisent la même entrée ou la même sortie.	Modéré
Procédurale	Une série d'opérations qui doivent s'exécuter en ordre.	Mal
Temporelle	Des éléments qui fonctionnent en même temps.	Mal
Logique	Des éléments qui font des tâches logiquement semblables.	Très Mal!
Par coïncidence	Des éléments qui n'ont aucune relation conceptuelle mais qui ont du code répliqué.	Très Mal!



# Métriques de Cohésion

- LCOM : Plus de 6 définitions!
- On va utiliser celle de Henderson-Sellers :
 
$$LCOM_{96a} = \frac{(\frac{1}{a} \sum_{j=1}^a \mu(A_j)) - m}{1 - m}$$
  - $a$  est le nombre d'attributs d'une classe,  $m$  est le nombre de méthodes,  $\mu(A_j)$  est le nombre de méthodes qui accèdent à l'attribut  $A_j$ .
- Mais pourquoi est-ce si difficile de calculer LCOM?
  - Il y a des classes sans attributs, ou avec une seule méthode, ou avec un seul attribut.
  - L'héritage complique la situation au niveau des attributs et des méthodes hérités.
  - Il y a les méthodes d'accès et de modification.
- TCC = NDP/NP, NDP = nombre de paires de méthodes qui accèdent aux attributs directement, NP = nombre de paires de méthodes  $(n(n-1)/2)$
- Des pourcentages (valeurs normalisées) **LCOM=0 → Cohésion maximale, TTC=1 → Cohésion maximale**
- Deux des métriques les plus fiables.
- Quand la **LCOM** diminue ou la **TCC** augment :
  - **Maturité, Compréhensibilité, Attraction, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité**

# Documentation

- LOD : En assumant un commentaire par classe et un commentaire par méthode, elle compte le pourcentage des éléments qui ne sont pas commentés.
  - $LOD = 1 - \#comm / (m + c + 1)$ , #comm=nombre de commentaires, m=nombre de méthodes, c=nombre de classes internes.
- Pourcentage (valeur normalisée)
- Quand la **LOD** augment :
  - Compréhensibilité, Opérabilité, Attraction, Maturité, Analysabilité, Possibilité de modification, Testabilité, Adaptabilité, Facilité d'apprentissage, Stabilité, Remplaçabilité

# Matrice des métriques

Software Quality Matrix

		Main property	Functionality					Reliability				Re-Usability				Efficiency			Maintainability				Portability							
		Sub Property	Suitability	Accuracy	Interoperability	Security	Functionality Compliance	Maturity	Fault-tolerance	Recoverability	Reliability Compliance	Understandability for Reuse	Learnability for Reuse	Operability for Reuse - Programmability	Attractiveness for Reuse	Re-Usability Compliance	Time Behavior	Resource Utilization	Efficiency Compliance	Analyzability	Changeability	Stability	Testability	Maintainability Compliance	Adaptability	Installability	Co-existence	Conformance	Replace ability	Portability Compliance
Category	Sub-Category	Metric																												
Complexity	size	LOC																												
	structural C.	SIZE 2																												
		NOM																												
		CC																												
		WMC																												
Architecture & Structure	Inheritance	RFC																												
		DIT																												
	Coupling	NOC																												
		Ca																												
		CBO																												
		CDBC																												
		CDOC																												
		Ce																												
		CF																												
		DAC																												
		I																												
		LD																												
		MPC																												
	PDAC																													
	Cohesion	LCOM																												
		ILCOM																												
TCC																														
Design	Documentation	LOD																												

## Legend

not evaluated	
highly directly, directly related	
highly inversely, inversely related	
not related	

# Bonus : Placement d'Entité

- L'ensemble des entités d'une classe est constitué de tous les attributs et de toutes les méthodes de la classe.
- L'ensemble des entités d'une méthode est constitué de tous les attributs et de toutes les méthodes qui sont utilisés par la méthode.
- L'ensemble des entités d'un attribut est constitué de toutes les méthodes qui accèdent à l'attribut.
- La distance entre une entité,  $e$  (méthode ou attribut), et une classe,  $C$ :

$$distance(e, C) = 1 - \frac{|S_e \cap S_C|}{|S_e \cup S_C|}, \text{ where } S_C = \bigcup \{e_i\}.$$

- Le degré de placement des entités dans une classe:

$$EntityPlacement_C = \frac{\sum_{e_i \in C} distance(e_i, C)}{|\text{entities} \in C|} \div \frac{\sum_{e_j \notin C} distance(e_j, C)}{|\text{entities} \notin C|}$$

# Bonus : Placement d'Entité

- EP est une métrique qui combine les concepts de la cohésion et du couplage.
- Des violations de cette métrique sont résolues par des mouvements des membres de la classe ou par l'extraction des membres vers une nouvelle classe.
- Elle est utilisée dans l'outil JDeodorant (au prochain cours!) pour évaluer l'impact des refactorings sur la qualité du logiciel.

# Liste des métriques (Android Studio (MetricsReloaded))

- Chidamber-Kemerer
  - CBO, DIT, LCOM, NOC, RFC, WMC
- Class count metrics
  - Number of classes, Number of product classes (non-test), Number of test classes (also the recursive version of these metrics for packages)
- Complexity metrics
  - Cognitive Complexity, Essential cyclomatic complexity, Design complexity, Cyclomatic complexity
- Dependency metrics
  - Number of Cyclic Dependencies, Number of Dependencies, Number of Transitive Dependencies, Number of Dependents, Number of Transitive Dependents, Number of Package Dependencies, Number of Dependent Packages
- Junit testing metrics
- Javadoc coverage metrics
- Lines of code metrics
  - CLOC, JLOC, LOC, NCLOC (SLOC), RLOC
- MOOD metrics (project level)
  - Attribute hiding factor (encapsulation), Attribute inheritance factor, Coupling factor, Method hiding factor, Method inheritance factor, Polymorphism factor
- Martin package metrics
  - Abstractness, Afferent coupling, Efferent coupling, Distance from the main sequence, Instability
- Number of files metrics

# List de métriques (Android Studio (MetricsTree))

## 1. Project level:

- Non-Commenting Source Statements, Lines Of Code, Number Of Concrete Classes, Number Of Abstract Classes, Number Of Static Classes, Number Of Interfaces, MOOD metrics set [1]:

## 2. Package level

- Non-Commenting Source Statements, Lines Of Code, Number Of Concrete Classes, Number Of Abstract Classes, Number Of Static Classes, Number Of Interfaces, Robert C. Martin metrics set [2, 3]:

## 3. Method level:

- LOC: Lines Of Code, CC: McCabe Cyclomatic Complexity, Maximum Nesting Depth, Loop Nesting Depth, Condition Nesting Depth, Number Of Loops, LAA: Locality Of Attribute Accesses, FDP: Foreign Data Providers, NOAV: NumberOfAccessedVariables, CINT: Coupling Intensity, CDISP: Coupling Dispersion

## 1. Class level

1. Chidamber-Kemerer metrics set [4]:
2. Lorenz-Kidd metrics set [5]:
  - NOA: Number of Attributes, NOO: Number of Operations, NOAM: Number of Added Methods, NOOM: Number of Overridden Methods
3. Li-Henry metrics set [6]:
  - SIZE2: Number of Attributes and Methods, MPC: Message Passing Coupling, DAC: Data Abstraction Coupling, NOM: Number of Methods
4. Lanza-Marinescu metrics set [7]:
  - ATFD: Access To Foreign Data, NOPA: Number Of Public Attributes, Number Of Accessor Methods, WOC: Weight Of A Class
5. Bieman-Kang metrics set [8]:
  - TCC: Tight Class Cohesion
6. Chr. Clemens Lee metrics set: - NCSS: Non-Commenting Source Statements

# Un modèle hiérarchique pour l'évaluation de la qualité de conception orientée-objet

Bansiya, J. and Davis, C.G., 2002. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, 28(1), pp.4-17.



# ISO 9126 métriques de critères

Design Metrics Descriptions

METRIC	NAME	DESCRIPTION
DSC	Design Size in Classes	This metric is a count of the total number of classes in the design.
NOH	Number of Hierarchies	This metric is a count of the number of class hierarchies in the design.
ANA	Average Number of Ancestors	This metric value signifies the average number of classes from which a class inherits information. It is computed by determining the number of classes along all paths from the "root" class(es) to all classes in an inheritance structure.
DAM	Data Access Metric	This metric is the ratio of the number of private (protected) attributes to the total number of attributes declared in the class. A high value for DAM is desired. (Range 0 to 1)
DCC	Direct Class Coupling	This metric is a count of the different number of classes that a class is directly related to. The metric includes classes that are directly related by attribute declarations and message passing (parameters) in methods.
CAM	Cohesion Among Methods of Class	This metric computes the relatedness among methods of a class based upon the parameter list of the methods [3]. The metric is computed using the summation of the intersection of parameters of a method with the maximum independent set of all parameter types in the class. A metric value close to 1.0 is preferred. (Range 0 to 1)
MOA	Measure of Aggregation	This metric measures the extent of the part-whole relationship, realized by using attributes. The metric is a count of the number of data declarations whose types are user defined classes.
MFA	Measure of Functional Abstraction	This metric is the ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class. (Range 0 to 1)
NOP	Number of Polymorphic Methods	This metric is a count of the methods that can exhibit polymorphic behavior. Such methods in C++ are marked as virtual.
CIS	Class Interface Size	This metric is a count of the number of public methods in a class
NOM	Number of Methods	This metric is a count of all the methods defined in a class.

Design Metrics for Design Properties

Design Property	Derived Design Metric
Design Size	Design Size in Classes (DSC)
Hierarchies	Number of Hierarchies (NOH)
Abstraction	Average Number of Ancestors (ANA)
Encapsulation	Data Access Metric (DAM)
Coupling	Direct Class Coupling (DCC)
Cohesion	Cohesion Among Methods in Class (CAM)
Composition	Measure of Aggregation (MOA)
Inheritance	Measure of Functional Abstraction (MFA)
Polymorphism	Number of Polymorphic Methods (NOP)
Messaging	Class Interface Size (CIS)
Complexity	Number of Methods (NOM)

Computation Formulas for Quality Attributes

Quality Attribute	Index Computation Equation
Reusability	$-0.25 * \text{Coupling} + 0.25 * \text{Cohesion} + 0.5 * \text{Messaging} + 0.5 * \text{Design Size}$
Flexibility	$0.25 * \text{Encapsulation} - 0.25 * \text{Coupling} + 0.5 * \text{Composition} + 0.5 * \text{Polymorphism}$
Understandability	$-0.33 * \text{Abstraction} + 0.33 * \text{Encapsulation} - 0.33 * \text{Coupling} + 0.33 * \text{Cohesion} - 0.33 * \text{Polymorphism} - 0.33 * \text{Complexity} - 0.33 * \text{Design Size}$
Functionality	$0.12 * \text{Cohesion} + 0.22 * \text{Polymorphism} + 0.22 * \text{Messaging} + 0.22 * \text{Design Size} + 0.22 * \text{Hierarchies}$
Extendibility	$0.5 * \text{Abstraction} - 0.5 * \text{Coupling} + 0.5 * \text{Inheritance} + 0.5 * \text{Polymorphism}$
Effectiveness	$0.2 * \text{Abstraction} + 0.2 * \text{Encapsulation} + 0.2 * \text{Composition} + 0.2 * \text{Inheritance} + 0.2 * \text{Polymorphism}$

# Métriques alternatives pour les ISO 9126

## métriques de critères

Design Property	Bansiya metric	Android Studio metric
Design size	Design Size in Classes (DSC)	Cp (Number of product classes)
Hierarchies	Number of Hierarchies (NOH)	Number of Abstract Classes + Number of Interfaces
Abstraction	Average Number of Ancestors (ANA)	SUM(NOC)/NOH
Encapsulation	Data Access Metric (DAM)	AHF
Coupling	Direct Class Coupling (DCC)	CBO
Cohesion	Cohesion Among Methods in Class (CAM)	1-(LCOM/Max(LCOM))
Composition	Measure of Aggregation (MOA)	Number of dependencies
Inheritance	Measure of Functional Abstraction (MFA)	MIF
Polymorphism	Number of Polymorphic Methods (NOP)	NOM*PF
Messaging	Class Interface Size (CIS)	NOM (only public methods)*
Complexity	Number of Methods (NOM)	NOM

# Sélection de type d'architecture selon les critères de qualité

Haoues, M., Sellami, A., Ben-Abdallah, H., & Cheikhi, L. (2017). A guideline for software architecture selection based on ISO 25010 quality related characteristics. International Journal of System Assurance Engineering and Management, 8(2), 886-909.

# Comparaison des architectures logicielles par rapport aux caractéristiques et sous-caractéristiques de qualité (1)

ISO 25010 characteristics	ISO 25010 sub-characteristics	SOA	MDA	CBA	EDA	AOA
Functional Suitability	Functional Completeness	0	0	+	0	0
	Functional Correctness	+	0	+	0	+
	Functional appropriateness	0	0	0	0	0
Reliability	Maturity	–	0	0	–	0
	Availability	0	+	+	–	+
	Fault Tolerance	–	+	+	+	+
	Recoverability	0	0	+	0	+
Usability	Appropriateness	+	0	0	0	0
	Recognizability					
	User interface aesthetics	–	+	0	0	0
	Operability	0	+	0	0	0
	User error protection	–	0	0	0	0
	Learnability	–	0	0	0	0
Compatibility	Accessibility	+	+	0	+	0
	Coexistence	+	+	0	0	0
	Interoperability	+	+	+	+	+

- SOA = Architecture orienté service comprendre, produire, outils qui existent déjà, changements, innovant
- MDA = Architecture orienté modèle adaptatif et facile à changer, interface simple, éviter erreurs, performance, blackboard et pipefilter, pls étapes pour identifier les mots le syntaxe
- CBA = Architecture des composantes fiabilité et sécurité sont les priorités
- EDA = Architecture évènementielle notifier, en temps réel, intégré
- AOA = Architecture orienté aspect autonome, en tout temps, pas faire d'erreurs, plus haute priorité, système fonctionne avec n'importe quelle voiture

# Comparaison des architectures logicielles par rapport aux caractéristiques et sous-caractéristiques de qualité (2)

		SOA	MDA	CBA	EDA	AOA
Performance efficiency	Time Behaviour	—	0	+	+	0
	Resource Utilization	+	+	0	+	+
	Capacity	—	—	—	+	+
Portability	Adaptability	+	+	+	+	+
	Instability	—	—	—	0	+
	Replaceability	+	+	+	0	0
Security	Confidentiality	—	—	0	—	0
	Integrity	—	+	+	—	0
	Non-repudiation	0	—	0	0	0
	Accountability	0	0	0	0	0
	Authenticity	—	—	0	0	0
Maintainability	Modularity	+	+	+	+	+
	Reusability	+	+	+	+	—
	Analyzability	+	0	+	0	0
	Modifiability	+	+	+	—	+
	Testability	—	0	—	0	0

- SOA = Architecture orienté service
- MDA = Architecture orienté modèle
- CBA = Architecture des composantes
- EDA = Architecture événementielle
- AOA = Architecture orienté aspect

# Relations entre les caractéristiques de qualité

	Functional suitability	Performance efficiency	Usability	Compatibility	Reliability	Security	Maintainability	Portability
Functional suitability	+	—	+	0	+	—	+	0
Performance efficiency	0	+	—	—	0	—	—	—
Usability	+	±	+	0	+	0	±	0
Compatibility	0	0	0	+	0	—	±	+
Reliability	+	0	+	0	+	0	+	0
Security	0	—	—	—	+	+	0	0
Maintainability	+	—	0	+	±	±	+	+
Portability	0	—	0	+	0	0	+	+

# Guide de sélection d'architecture

	Functional suitability	Performance efficiency	Usability	Compatibility
1st choice	CBA	EDA	MDA	SOA MDA
2nd choice	SOA AOA	AOA	EDA	CBA EDA AOA
3rd choice	MDA EDA	MDA CBA	CBA AOA	
	Reliability	Security	Maintainability	Portability
1st choice	CBA AOA	CBA	MDA	AOA
2nd choice	MDA	AOA	SOA CBA	EDA
3rd choice	EDA	EDA	EDA AOA	SOA MDA CBA

# La prochaine fois

