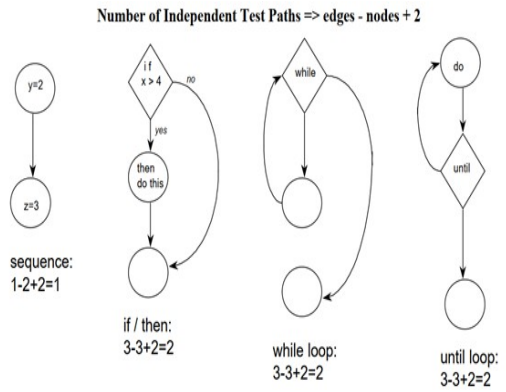
Cyclomatic complexity (pas nécessairement orienté objet):

La complexité cyclomatique (McCabe) est utilisée pour évaluer la complexité d'un algorithme dans une méthode. Il s'agit du nombre de cas de tests nécessaires pour tester de manière exhaustive la méthode. La formule de calcul de la complexité cyclomatique est le nombre d'arêtes moins le nombre de nœuds plus 2. Pour une séquence où il n'y a qu'un seul chemin, aucune alternative ou choix, un seul cas de test est nécessaire. Cependant, une boucle " if " a deux choix : si la condition est vraie, un chemin est testé ; si la condition est fausse, un chemin alternatif est testé.

Exemples de calcul de la complexité cyclomatique :



La complexité cyclomatique ne peut pas être utilisée pour mesurer la complexité d'une classe en raison de l'héritage, mais la complexité cyclomatique des méthodes individuelles peut être combinée avec d'autres métriques pour évaluer la complexité de la classe.

En surveillant les niveaux de complexité cyclomatique dans le code source, on peut identifier les parties du logiciel qui pourraient nécessiter un examen plus approfondi et une attention particulière lors des tests. Réduire la complexité cyclomatique en refactorisant le code ou en le réorganisant peut aider à rendre le logiciel plus compréhensible, plus facile à maintenir et moins sujet aux erreurs.

WMC (Weighted method Class):

Définition:

WMC mesure la complexité d'une classe individuelle. Si toutes les méthodes sont considérées comme également complexes, alors WMC est simplement le nombre de méthodes définies dans chaque classe.

WMC = $\sum_{i=1}^{i=n} ii$, où une classe C1 possède M1, … Mn, méthodes avec une complexité c1, …, cn respectivement

Utilités :

C&K [Chidamber and Kemerer, 1994] suggèrent que WMC est destiné à mesurer la complexité d'une classe. Par conséquent, il est un indicateur de l'effort nécessaire pour développer et maintenir une classe.

Commentaires :

Le WMC (Weighted Methods per Class) est un indicateur de l'effort nécessaire pour développer une classe, car une classe contenant une seule méthode volumineuse peut prendre autant de temps à développer qu'une classe contenant un grand nombre de petites méthodes. Par conséquent, il devrait simplement être considéré comme une mesure de la taille de la classe.

Les classes comportant un grand nombre de méthodes nécessitent davantage de temps et d'efforts pour être développées et entretenues, car cela aura un impact sur les sous-classes héritant de toutes les méthodes.

Classes avec un grand nombre de méthodes ont tendance à être plus spécifiques à une application, limitant ainsi la possibilité de réutilisation.

Niveau d'héritage :

La relation d'héritage est perçue comme un compromis. Elle complique la compréhension et la maintenabilité des systèmes, rendant difficile le changement de l'interface d'une superclasse.

L'utilisation de l'héritage multiple n'est généralement pas recommandée en raison des collisions de noms et du manque de compréhensibilité dans les langages qui permettent aux méthodes d'être héritées de plusieurs

superclasses, comme C++. Des langages tels que Java ne prennent en charge que l'héritage simple.

DIT : (Depth of inheritence tree) :

Définition :

La profondeur d'une classe dans la hiérarchie d'héritage est le nombre maximum d'étapes depuis le nœud de la classe jusqu'à la racine de l'arbre, et elle est mesurée par le nombre de classes ancestrales. Plus une classe est profonde dans la hiérarchie, plus elle est susceptible d'hériter d'un grand nombre de méthodes, ce qui rend plus complexe la prédiction de son comportement. Les arbres plus profonds constituent une complexité de conception plus grande, car plus de méthodes et de classes sont impliquées, mais ils offrent également un potentiel plus important pour la réutilisation des méthodes héritées.

Commentaires :

Les profondeurs d'héritage importantes indiquent des objets complexes qui peuvent être difficiles à tester et à réutiliser. Les profondeurs d'héritage faibles peuvent indiquer un code fonctionnel qui n'exploite pas pleinement le mécanisme d'héritage.

[Cartwright and Shepperd, 1996] ont trouvé une corrélation positive entre la métrique DIT et le nombre de problèmes signalés par les utilisateurs, remettant en question l'utilisation efficace de l'héritage.

L'utilisation des cadres (frameworks) affecte la DIT (Depth of Inheritance Tree). Les cadres comprennent plusieurs niveaux d'héritage et servent souvent de base pour les objets d'application.

La DIT (Depth of Inheritance Tree) ne peut pas être une mesure valide de la réutilisation, car il peut arriver facilement qu'une valeur élevée de DIT réutilise le même nombre ou moins de méthodes qu'une hiérarchie peu profonde et large.

2-NOC (Number Of Children) :

Le nombre d'enfants est le nombre de sous-classes immédiates subordonnées à une classe dans la hiérarchie. Plus le nombre d'enfants est élevé, plus il est probable qu'il y ait une abstraction incorrecte de la classe parente et cela peut être un cas d'utilisation abusive de la sous-classification.

Plus grand le NOC , plus il est difficile de modifier la classe mère car cela affecte tous ses enfants et entraîne une forte dépendance vis-à-vis de la classe de base.