POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

LOG8430 : Attributs de qualité, partie 2

Attributs de qualité – partie 2

Définition des attributs de qualité

1. Disponibilité
2. **Interopérabilité**
3. **Modifiabilité**
4. Performance
5. Sécurité
6. Testabilité
7. Utilisabilité
8. Autres attributs de qualité

Attribut de qualité 2: Interopérabilité

L'interopérabilité réfère au degré avec lequel deux systèmes, ou plus, peuvent échanger de l'information utile au travers d'interfaces dans un contexte donné.

Cette propriété implique l'habilité d'interpréter correctement les données échangées (interopérabilité sémantique).

Un système ne peut pas être interopérable de façon isolée.
L'interopérabilité implique d'identifier le contexte:

- Avec quel autre système?
- Quelles données?
- Dans quelles circonstances?

Attribut de qualité 2: Interopérabilité

L'échange d'information entre deux système peut prendre de très nombreuses formes:

- Du plus direct: le programme A appelle le programme B en lui passant des paramètres,
- Au plus implicite: le programme A est écrit en supposant que le programme B produira tel information dans tel format ou aura tel comportement.

Les systèmes ou composantes ont souvent des attentes précises à propos du comportement de leurs partenaires lors d'échanges d'information.

Attribut de qualité 2: Interopérabilité

Dans un contexte d'interopérabilité, le concept d'interface prend un sens beaucoup plus large qu'une entente sur le nom des méthodes et les types de paramètres.

L'interface d'une entité doit inclure un ensemble de suppositions qu'elle est sûr de faire à propos de cette entité.

Exemple: besoin de recalibration périodique du système antimissile utilisé lors de la guerre en Iraq.

Attribut de qualité 2: Interopérabilité

Des situations typiques qui exigent de l'interopérabilité entre systèmes:

- Quand on fournit un service utilisé par plusieurs systèmes qui nous sont inconnus.
  → Par exemple un service de cartographie (Google Maps) ou de prévision météo (Accuweather).
- Quand on assemble une nouvelle fonctionnalité à partir de plusieurs systèmes déjà existants.
  → Par exemple:
  1. un système responsable d'acquérir les données de senseurs dans un environnement,
  2. un système responsable de traiter les données brutes,
  3. un système responsable d'interpréter les données, et
  4. un système responsable de représenter les données et de les distribuer.

  → Par exemple, un service de monitoring du trafic routier.

Attribut de qualité 2: Interopérabilité

Deux aspects importants de l'interopérabilité:

- Découverte: le consommateur d'un service doit découvrir (avant ou durant l'exécution) l'emplacement, l'identité et l'interface d'un service.
- Traitement de la réponse (3 possibilités):
  1. Le service répond au demandeur avec la réponse,
  2. Le service envoie sa réponse à un autre système,
  3. Le service diffuse sa réponse à toutes les parties intéressées.

Ces deux aspects mènent aux scénarios et tactiques pour l'interopérabilité.

Attribut de qualité 2: Interopérabilité

Les systèmes de systèmes (SoS): un groupe de systèmes qui collaborent pour atteindre un objectif commun.

Catégories de SoS:

1. Dirigé: Les objectifs, une gestion centralisée, le financement et l'autorité pour l'ensemble du SoS sont en place. Les systèmes sont subordonnés au SoS.
2. Reconnu: Les objectifs, une gestion centralisée, le financement et l'autorité pour l'ensemble du SoS sont en place. Les systèmes conservent leur propre gestion, financement et autorité, en parallèle à ceux du SoS.
3. Collaboratif: Pas d'objectifs, de gestion centralisée, de financement ou d'autorité au niveau du SoS. Les systèmes fonctionnent volontairement ensemble pour atteindre des intérêts communs ou partagés.
4. Virtuel: Comme collaboratif, mais les systèmes ne se connaissent pas.

Attribut de qualité 2: Interopérabilité

Dans les SoSs dirigés et reconnus, il y a une tentative délibérée de créer un SoS.

La différence clé entre les 2 catégories réside dans le degré d'autonomie des systèmes, plus grand dans le cas d'un système reconnu.

Les SoSs collaboratifs et virtuels sont plus ad hoc. Absence d'autorité globale ou de financement spécifique.

Dans le cas d'un SoS virtuel, absence de connaissance des participants et de l'envergure du SoS entier.

Attribut de qualité 2: Interopérabilité

Le cas des SoSs collaboratifs est très commun:

- Dans le cas de Google Maps
  - Google est le gestionnaire et l'autorité qui finance le service de cartes.
  - Chaque application utilisant le service a ses propres gestionnaire et source de financement.
  - Pas de gestion globale de toutes les applications qui utilisent Google Maps.
  - Les différentes applications collaborent pour assurer le bon fonctionnement de tous.

Un SoS virtuel implique de grands systèmes et est souvent plus ad hoc:

- Exemple du réseau électrique aux ÉU
  - Plus de 3000 compagnies d'électricité.
  - Commission électrique publique dans chaque état.
  - Le département de l'énergie fédéral impose certaines politiques et règles.
  - Plusieurs systèmes du réseau doivent interagir, mais il n'y a pas d'autorité centrale.

Scénario général d'interopérabilité

Source du stimulus

Un système initie une requête.

Stimulus

Un requête pour échanger de l'information entre des systèmes.

Artéfact

Les systèmes qui souhaitent interopérer

Environnement

Les systèmes qui souhaitent interopérer sont découverts à l'exécution ou sont connus avant.

Réponse

La requête demandant d'interopérer résulte en un échange d'information. L'information est comprise par le récepteur tant syntaxiquement que sémantiquement. Autrement la requête est rejetée et les entités appropriées sont averties. Dans tous les cas, la requête peut être enregistrée.

Mesure de la réponse

Le pourcentage de requêtes correctement traitées ou le pourcentage de requêtes correctement rejetées.

Exemple de scénario concret d'interopérabilité

Source du stimulus

Système d'information du véhicule

Stimulus
Position courante envoyée

Artéfact
Système de surveillance du trafic

Environnement
Système connu avant l'exécution

Réponse
Le système de surveillance du trafic combine la position courante avec d'autre information, superpose à une carte Google Maps et retransmet l'information

Mesure de la réponse
L'information de position du véhicule incluses correctement 99.9% du temps.

Attribut de qualité 2: Interopérabilité

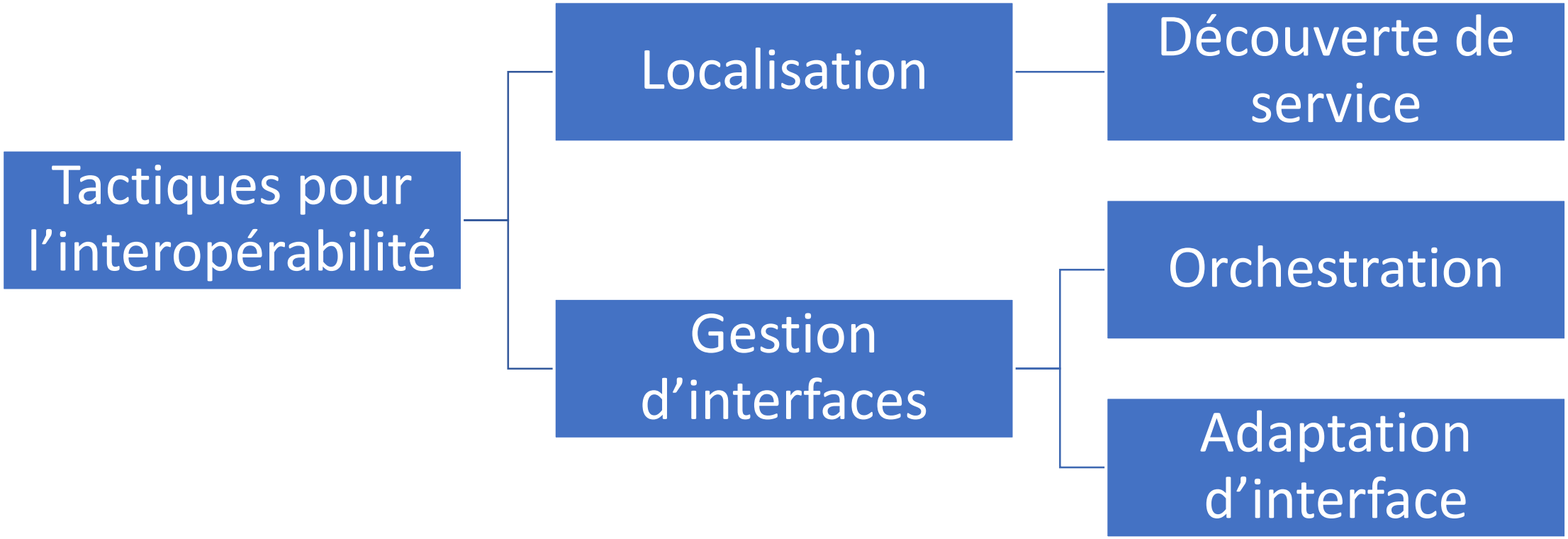
Deux technologies courantes pour permettre à des applications web d'interopérer:

1. SOAP: protocole d'échange de données basé sur XML.
   → Définition de plusieurs standards pour la composition de services, la définition de transactions, la découverte de services, la fiabilité…
2. REST: protocole d'accès à des ressources basé sur la sémantique des opérations de base du protocole HTTP.
   → Opérations de création, lecture, mise à jour et destruction des ressources (CRUD).
   → Méthode d'adressage simple basé sur des URI (*Universal Ressource Identifier*).
   → Interopérabilité sémantique non garantie par l'interface.

Attribut de qualité 2: Tactiques pour l'interopérabilité

Des systèmes interopèrent s'ils sont en mesure de formuler des requêtes et d'échanger de l'information entre eux, qui soit comprise de chacun.

Attribut de qualité 2: Tactiques pour l'interopérabilité



Attribut de qualité 2: Tactiques pour l'interopérabilité: **Localisation**

Une seule tactique pour la localisation de services, utilisée pour trouver un service en cours d'exécution:

- Découverte de service: utilisation d'un service de localisation connu.
- Peut impliquer plusieurs niveaux d'indirection.
- Identification de service par:
  - Type de service,
  - Nom,
  - Localisation,
  - Autres critères.

Attribut de qualité 2: Tactiques pour l'interopérabilité: **Gestion d'interfaces**

Deux tactiques pour la gestion d'interfaces:

- Orchestration:
  - Mise en place d'un mécanisme de contrôle pour coordonner, gérer et séquencer les appels à des services particuliers.
  - Utilisée pour permettre à des services d'interagir de façon complexe.
  - Les engins de gestions des flux de travail sont des exemples d'utilisation de cette tactique.
  - Le patron Médiateur peut assumer cette fonction dans des cas simples.
  - Spécification d'orchestrations complexes à l'aide de BPEL (*Business Process Execution Language*)
- Adaptation d'interfaces:
  - Ajouter ou retirer des capacités à une interface.
  - Capacités telles que traduction, mise en mémoire tampon, lissage.
  - Dissimulation de certaines fonctions à des usagers non fiables.
  - Les patrons Décorateur et Adapteur sont des exemples de cette stratégie.

Attribut de qualité 2: Interopérabilité
Les standards et leurs limites

Pour s'entendre sur la structure et la fonction dans les échanges d'informations, l'industrie tend à adopter des standards.

Les standards jouent un rôle central pour rendre des systèmes interopérables.

Les standards sont utiles et souvent indispensables, mais les attentes engendrées par l'adoption de standards sont souvent irréalistes.

Attribut de qualité 2: Interopérabilité
Les standards et leurs limites

Défis associés à l'utilisation de standards:

1. Implémentation: chaque organisation doit implémenter sa propre version d'un standard. Ces implémentations sont différentes et pas rarement parfaitement compatibles entre-elles.
2. Extensibilité: les standards contiennent fréquemment des points d'extension que les organisations utilisent pour se distinguer, mais qui rendent les informations contenues dans les extensions incompatibles.
3. Cycle de vie: les standards évoluent et changent. Différentes version d'un standard ne sont pas nécessairement compatibles entre-elles. Chaque organisation doit décider quelle version du standard supporter à quel moment.

Attribut de qualité 2: Interopérabilité
Les standards et leurs limites (suite)

Défis associés à l'utilisation de standards:

4. Spécification: les spécifications des standards sont imparfaites. Certains standards sont sous-spécifiés, d'autres sont surspécifiés, instables ou insignifiants.
5. Compétition: des standards distincts couvrent souvent le même domaine d'information. Choisir le bon standard équivaut souvent à choisir un camp au sein de l'industrie.
6. Frein à l'innovation: des standards trop stricts ou limités réduisent la flexibilité et l'expressivité, ce qui peut ralentir les développements.

Attribut de qualité 2: Interopérabilité
Les standards et leurs limites (suite)

Les standards ne peuvent pas servir à décider de l'architecture d'un système.

L'architecture d'un système doit être décidée d'abord. Les standards à supporter sont décidés ensuite.

Les standards peuvent ainsi évoluer sans affecter l'architecture globale du système.

Liste de vérification pour l'interopérabilité
Allocation des responsabilités

- Déterminer quelles responsabilités du système doivent interopérer avec d'autres systèmes.
- S'assurer que des responsabilités ont été ajoutées pour détecter les requêtes exigeant d'interopérer avec d'autres systèmes, connus ou non.
- S'assurer que des responsabilités sont liées à:
  - Accepter une requête
  - Échanger de l'information
  - Rejeter une requête
  - Avertir les entités appropriées
  - Enregistrer les requêtes (pour assurer la non-répudiation dans un environnement non fiable, l'enregistrement des requêtes est essentiel)

Liste de vérification pour l'interopérabilité
Modèle de coordination

- S'assurer que les mécanismes de coordination peuvent rencontrer les requis d'attributs de qualité critiques. Ceci inclut de considérer les aspects suivants:
  - La quantité de trafic réseau créé tant par les systèmes sous votre contrôle que par les systèmes qui ne sont pas sous votre contrôle.
  - La précision temporelle des messages envoyés par votre système.
  - L'actualité des messages envoyés par votre système.
  - La variance temporelle (gigue/jitter) dans l'arrivée des messages.
  - Vérifier que tous les systèmes sous votre contrôle font des suppositions concernant les protocoles et les infrastructures réseau qui sont consistantes avec les systèmes qui ne sont pas sous votre contrôle.

Liste de vérification pour l'interopérabilité
Modèle de données

- Déterminer la syntaxe et la sémantique de toutes les abstractions de données importantes qui peuvent être échangées entre les systèmes qui interopèrent.
- S'assurer que toutes ces abstractions sont consistantes avec les données provenant des systèmes avec lesquels le système interopère. Ceci peut impliquer d'appliquer des transformations sur le modèle de données s'il doit être gardé confidentiel.

Liste de vérification pour l'interopérabilité
Correspondance entre les éléments architecturaux

- Pour l'interopérabilité, la correspondance critique est celle des composantes logicielles sur les processeurs.
- En plus de s'assurer que les composantes qui doivent interopérer avec d'autres systèmes soient installées sur des processeurs qui peuvent communiquer avec le réseau, les principales considérations impliquent la sécurité, la disponibilité et la performance des communications.

Liste de vérification pour l'interopérabilité
Gestion des ressources

- S'assurer que l'interopérabilité avec d'autres systèmes ne peut jamais épuiser une ressource critique du système.
- S'assurer que la charge sur les ressources imposée par les communications liées à l'interopérabilité est acceptable.
- S'assurer que si des ressources doivent être partagées entre les systèmes participants, qu'une politique d'arbitrage adéquate soit en place.

Liste de vérification pour l'interopérabilité
Choix du moment pour effectuer une liaison

- Déterminer les systèmes qui peuvent interopérer et le moment où ils deviennent connus les uns des autres.
- Pour chaque système sur lequel vous avez le contrôle:
  - S'assurer qu'il a une politique pour gérer les liaisons vers d'autres systèmes externes, connus ou non.
  - S'assurer qu'il a un mécanisme pour rejeter les requêtes de liaisons inacceptables et pour enregistrer ces requêtes.
  - Dans le cas de liaisons à l'exécution, s'assurer que les mécanismes vont supporter la découverte des services et protocoles appropriés ou l'envoie d'information selon les protocoles choisis.

Liste de vérification pour l'interopérabilité
Choix de technologie

- Pour chacune des technologies que vous avez choisies, est-elle visible au niveau des interfaces du système? Si oui, quels effets sur l'interopérabilité a-t-elle? Supporte-t-elle, empêche-t-elle ou est-elle neutre quand aux scénarios d'interopérabilité s'appliquant à votre système? S'assurer que ces effets sont acceptables.
- Considérer des technologies qui supportent l'interopérabilité, telle que les services web. Peuvent-elles être utilisées pour satisfaire les requis d'interopérabilité des systèmes sous votre contrôle?

Attribut de qualité 3: Modifiabilité

La modifiabilité réfère au degré de facilité avec lequel un système peut être modifié, avec quel niveau de risque et à quel coût.

Nombre d'études montrent que la plus grande partie du coût d'un système logiciel est lié à son entretien et aux changements effectués après que le système ait été déployé.

Pour planifier en fonction de la modifiabilité, un architecte doit se poser quatre questions

- Qu'est-ce qui peut changer?
- Quelle est la probabilité de changement?
- Quand le changement est-il effectué et par qui?
- Quel est le coût du changement?

Attribut de qualité 3: Modifiabilité
Qu'est-ce qui peut changer?

TOUT!

- Les opérations effectuées par le système,
- La plateforme (matériel, système d'exploitation, intergiciel) sur laquelle le système s'exécute,
- L'environnement dans lequel le système s'exécute:
  - Les systèmes avec lesquels il doit interopérer,
  - Les protocoles utilisés.
- Les attributs de qualité que le système doit posséder:
  - Sa performance, sa fiabilité et même sa capacité à être modifié !
- Sa capacité de traitement:
  - Nombre d'usagers supportés, nombre d'opérations simultanés.

Attribut de qualité 3: Modifiabilité
Quelle est la probabilité de changement?

Il n’est pas possible de planifier pour tous les types de changements.

Malgré le fait que tout peut changer, l’architecte doit prendre les décisions difficiles concernant les changements probables et qui seront facilités et ceux qui sont improbables.

Attribut de qualité 3: Modifiabilité
Quand le changement est-il effectué et par qui?

Auparavant, les changements les plus fréquents étaient ceux faits au code source.

Aujourd'hui, la question du moment où un changement est fait s'entremêle à celle de qui fait le changement.

Des changements peuvent être faits:

- Au code source,
- À la compilation (p.ex. à l'aide d'énoncés de compilation conditionnelle),
- À la construction (p.ex. par le choix de bibliothèques),
- À la configuration (p.ex. en changeant des valeurs de paramètres),
- À l'exécution (p.ex. par des paramètres, plugins, etc.).

Ces changements peuvent être faits par les développeurs, les utilisateurs ou les administrateurs système.

Attribut de qualité 3: Modifiabilité
Quel est le coût du changement?

Rendre un système plus facilement modifiable implique 2 sortes de coûts:

- Le coût d'introduction d'un mécanisme pour rendre le système plus facilement modifiable,
- Le coût d'effectuer la modification à l'aide du mécanisme.

Attribut de qualité 3: Modifiabilité
Quel est le coût du changement?

Le mécanisme de changement le plus simple est d'attendre qu'une demande de changement soit formulée pour modifier le code source:

- Pas de coût pour introduire un mécanisme de modification.
- Le coût du changement est celui de **modifier** le code source, de **tester** et de **déployer** la nouvelle version.

À l'autre extrême, on peut développer un générateur d'applications (p.ex. un générateur d'interfaces utilisateur):

- Le générateur prend en entrée une spécification modifiée du système et génère un nouveau code source.
- Le coût du changement comprend: le coût de **développer le générateur**, le coût de **modifier la spécification**, le coût d'**exécuter le générateur**, de **tester** et de **déployer** la nouvelle version.

Attribut de qualité 3: Modifiabilité
Quel est le coût du changement?

Pour justifier l'introduction d'un mécanisme de modification, les coûts liés à son développement et son utilisation doivent être inférieurs à ceux nécessaires pour modifier le système sans le mécanisme.

La fréquence des changements est un facteur clé pour introduire un mécanisme de modification.

La prédiction de la fréquence des changements reste une estimation entachée d'une grande incertitude:

→ Si moins de modifications que prévues sont effectuées, le mécanisme ne sera pas justifié.

→ Le temps nécessaire au développement des mécanismes de modification pourrait être investi autrement (ajout de fonctionnalité, optimisation, …).

Scénario général de modifiabilité

Source du stimulus

Identité de l'intervenant qui fait le changement:

- Développeur
- Administrateur système
- Utilisateur

Stimulus

Spécification du changement à effectuer.
Ajout/changement ou retrait de fonctionnalité.
Changement à un attribut de qualité du système (performance, capacité, etc.).
Changement de technologie.

Artéfact

Spécification de ce qui doit être changer: composante ou module, plateforme, interface, environnement

Environnement

Moment où le changement peut être effectué: à la conception, à la compilation, à la construction, à l'installation ou à l'exécution.

Réponse

Faire le changement, le tester et le déployer.

Mesure de la réponse

Habituellement, le temps et l'argent qu'ont nécessité le changement. L'étendue des changements (nombre de modules, classes, etc. qui ont été modifiés) ou le nombre de nouveaux défauts introduits. L'effet sur les attributs de qualité.

Exemple de scénario concret de modifiabilité

Source du stimulus

Développeur

Stimulus
Souhaite changer l'interface utilisateur

Artéfact
Code source

Environnement
Durant la conception

Réponse
Changement effectué et tests unitaires passés.

Mesure de la réponse
En 3 heures.

Attribut de qualité 3: Tactiques pour la modifiabilité

Les tactiques pour contrôler la modifiabilité visent à contrôler la difficulté d'effectuer des changements, le temps nécessaire et le coût.

Attribut de qualité 3: Tactiques pour la modifiabilité: contrôler le couplage et la cohésion

Les modules d'un système assument des responsabilités.

Lorsqu'un changement implique de changer un module, ce changement modifie ses responsabilités d'une façon quelconque.

Un changement qui affecte un seul module est généralement plus simple et moins coûteux qu'un changement qui affecte plusieurs modules.

Si les responsabilités de deux modules se chevauchent, un changement dans un module risque grandement d'affecter l'autre.

On mesure le chevauchement par la probabilité qu'un changement dans un module implique un changement dans un autre: c'est le couplage.

Attribut de qualité 3: Tactiques pour la modifiabilité: contrôler le couplage et la cohésion

La cohésion mesure le degré de relation qui existe entre les responsabilités d'un module.

La cohésion mesure l'unité de but dans un module:

- Liée au nombre de scénarios de changement du module,
- Probabilité qu'un scénario de changement qui affecte une responsabilité affecte aussi une autre responsabilité.
- Plus la cohésion est élevée, moins grande est la probabilité qu'un changement affecte plusieurs responsabilités.

Attribut de qualité 3: Tactiques pour la modifiabilité: contrôler le couplage et la cohésion

Paramètres justifiants les tactiques de modifiabilité:

- Taille d'un module: les tactiques qui divisent des modules vont réduire le coût d'une modification au module, dans la mesure où la division reflète le type de changement qui sont probables.
- Couplage: réduire la force du couplage entre deux modules fait décroître le coût de modification de l'un de ces modules. Les tactiques qui réduisent le couplage introduisent des intermédiaires entre les modules.
- Cohésion: si un module a une faible cohésion, on peut améliorer la cohésion en retirant du module des responsabilités qui ne sont pas affectées par les changements anticipés.

Attribut de qualité 3: Tactiques pour la modifiabilité

Il faut aussi considérer le moment, dans le cycle de développement d'un logiciel, où un changement est effectué.

Si on ignore le coût de préparation de l'architecture pour permettre un changement, on préfère que le changement soit effectué le plus tard possible dans le cycle de vie du logiciel.

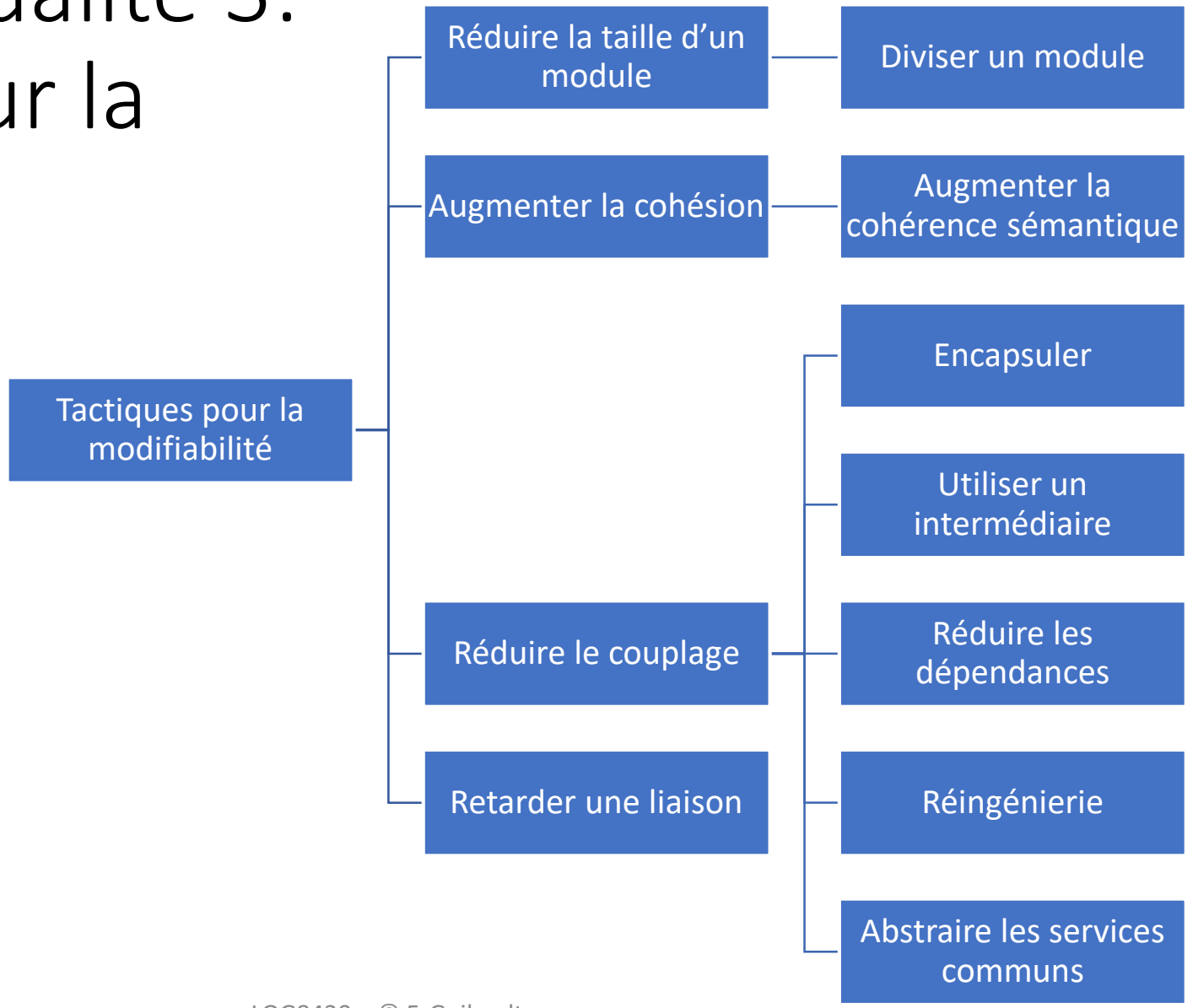
Des changements peuvent être faits facilement uniquement si l'architecture du système a été préparée pour permettre le changement.

Attribut de qualité 3: Tactiques pour la modifiabilité

Un autre paramètre justifiant les tactiques pour la modifiabilité est le moment où le changement est introduit:

- Une architecture correctement équipée pour permettre les changements tard dans le cycle de vie du logiciel coûte, en moyenne, moins qu'une architecture qui force les changements à être effectués plus tôt.
- La préparation du système fait en sorte que le coût de certaines modifications sera très bas ou nul pour des modifications faites tard.
- Cependant, ceci néglige le coût de préparation de l'architecture.

Attribut de qualité 3: Tactiques pour la modifiabilité



Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire la taille d'un module**

Si un module qui doit être modifié inclut beaucoup de fonctionnalités distinctes, le coût de modification risque d'être élevé.

Diviser un module en plusieurs modules plus petits devrait permettre de réduire le coût des changements futurs.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Augmenter la cohésion**

Plusieurs tactiques impliquent de déplacer des responsabilités d'un module vers un autre. L'objectif étant de réduire la probabilité d'introduire des effets indésirables en cas de modification.

Accroître la cohérence sémantique: si deux responsabilités dans un même module ne vise pas le même but/objectif, ces responsabilités devraient être placées dans des modules distincts.

- Déplacer des responsabilités peut exiger de créer un nouveau module, ou de déplacer une responsabilité vers un autre module existant,
- Pour identifier les responsabilités à déplacer, on peut se demander ce qui est le plus probable de changer. Si des responsabilités ne sont pas affectées par ces changements, elles devraient probablement être déplacées.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire le couplage**

Encapsuler: introduire une interface explicite à un module.

- L'interface inclut une API et les responsabilités associées (transformations de données, etc.)
- L'encapsulation réduit la probabilité qu'un changement dans un module se propage à d'autres modules.
- Le couplage avec le module est transféré vers l'interface, mais de façon souvent réduite puisque l'interface limite les façons d'interagir avec le module.
- Les responsabilités externes ne pourront interagir avec le module qu'au travers de son interface.
- Les interfaces conçues pour améliorer la modifiabilité devraient être abstraites en ce qui a trait aux détails du module qui sont appelés à changer. Ces détails devraient être cachés.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire le couplage**

Utiliser un intermédiaire: un intermédiaire casse les dépendances.

- Le type d'intermédiaire dépend du type de dépendance:
  - Un intermédiaire de type observateur (*publish-subscribe*) élimine la connaissance du producteur de données des consommateurs.
  - Un entrepôt de données partagé élimine la connaissance des lecteurs des données de ceux qui écrivent les données.
  - Dans une architecture orientée-service, où les services sont découverts à l'exécution par une recherche dynamique, le service de répertoire est un intermédiaire.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire le couplage**

Restreindre les dépendances: retreindre les modules avec lesquels un module interagit ou dont il dépend.

- Consiste à restreindre la visibilité d'un module et implémenter des autorisations d'accès.
- Cette tactique est courante dans les architectures en couches. Une couche ne peut accéder qu'aux couches inférieures, et parfois uniquement à la couche immédiatement inférieure.
- Utilisation d'un enrobage (wrapper), où les entités ne peuvent dépendre que de l'enrobage et non de ce qui est enrobé.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire le couplage**

Restructurer: étape de nettoyage du code afin d'en améliorer la qualité.

- Une pratique importante dans les approches de développement basés sur des méthodes agiles.
- S'assurer qu'il n'y a pas de duplication de code ou de code trop complexe.
- Le concept s'applique aussi à l'architecture du système:
  - → Des responsabilités communes (et le code associé) sont extraites des modules où elles se trouvent et assignées à un module qui leur est propre.
- En regroupant des responsabilités communes, l'architecte peut réduire le couplage.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Réduire le couplage**

Abstraire les services communs: Construire un module plus abstrait pour fusionner deux modules très semblables mais pas identiques.

- Lorsque deux modules fournissent des services similaires mais non identiques, il peut être plus efficace au niveau des coûts d'implémenter le service une seule fois de façon plus générale.
- Des modifications au service ne doivent être faites qu'à un seul endroit.
- Une approche courante pour introduire une abstraction consiste à paramétriser la description et l'implémentation des activités du module.
- Les paramètres peuvent être aussi simple que des valeurs pour certaines variables clés ou aussi complexe que des expressions dans un langage spécialisé qui sont interprétées.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Retarder une liaison**

Concevoir des artéfacts incorporant un certain niveau de flexibilité, puis utiliser cette flexibilité, est généralement moins coûteux que de coder manuellement un changement spécifique.

Ajouter des paramètres est le mécanisme le plus courant pour ajouter de la flexibilité.

Lorsque l'on choisit la valeur d'un paramètre durant une phase différente du cycle de vie du logiciel que celle où le paramètre a été introduit, on applique la tactique consistant à **retarder une liaison**.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Retarder une liaison**

Plus une valeur peut être liée tard dans le cycle de vie, plus flexible sera le système.

Introduire des mécanismes pour retarder les liaisons a un coût: on veut donc lier des valeurs le plus tard possible si l'introduction du mécanisme pour le faire est rentable.

On appel « externaliser » un changement le fait de permettre à une tierce partie (utilisateur ou installateur) de faire un changement sans modifier le code.

Attribut de qualité 3: Tactiques pour la modifiabilité: **Retarder une liaison**

Des tactiques pour effectuer une liaison

À la compilation:

- Remplacement de composantes (script de construction ou makefile)
- Paramétrisation à la compilation
- Aspects

Au déploiement:

- Fichiers de configuration/Wizards

Au démarrage ou à l'initialisation:

- Fichiers de ressource

À l'exécution

- Enregistrement à l'exécution
- Recherche dynamique (e.g. services)
- Paramètres interprétés
- Liaison au démarrage (e.g. dll)
- Serveurs de noms
- Plugiciels
- Publish-subscribe (observateur)
- Entrepôts partagés
- Polymorphisme

Liste de vérification pour la modifiabilité
Allocation des responsabilités

- Déterminer quelles changements ou catégories de changements sont susceptibles de se produire en considérant les forces techniques, légales, sociales, d'affaire ou de clientèle qui affecte le système:
  - Déterminer les responsabilités qui devraient être ajoutées, modifiées ou retirées pour effectuer le changement.
  - Déterminer quelles responsabilités sont affectées par le changement.
  - Déterminer une allocation de responsabilité aux modules qui place, autant que possible, les responsabilités appelées à changer ensemble (ou affectées par le changement) dans un même module, et celles qui vont changer à des moments différents dans des modules séparés.

Liste de vérification pour la modifiabilité
Modèle de coordination

- Déterminer quelle fonctionnalité ou attribut de qualité peuvent changer à l'exécution et de quelle façon cela affecte la coordination.
  → P.ex. l'information communiquée ou le protocole de communication changeront-ils en cours d'exécution? Si oui, s'assurer que ces changements n'affectent qu'un petit ensemble de modules.
- Déterminer quels dispositifs, protocoles et chemins de communication utilisés pour la coordination sont susceptibles de changer.
  → Pour ces dispositifs, protocoles et chemins de communication, s'assurer que ces changements n'affectent qu'un petit ensemble de modules.
- Pour les éléments pour lesquels la modifiabilité est un enjeu, utiliser un modèle de coordination qui réduit le couplage comme publish-subscribe, reporte la liaison comme un service de bus d'entreprise ou restreint les dépendances comme un mécanisme d'émission (broadcast).

Liste de vérification pour la modifiabilité
Modèle de données

- Déterminer quels changements aux abstractions de données, leurs opérations ou leurs propriétés sont susceptibles de changer.
- Déterminer quels changements à ces abstractions vont impliquer leur création, initialisation, sauvegarde, manipulation, traduction ou destruction.
- Déterminer, pour chaque type de changement, s'ils seront effectués par les utilisateurs, les administrateurs système ou les développeurs.
- Pour les changements effectués par les utilisateurs ou les administrateurs système, s'assurer que les attributs nécessaires sont visibles et que les utilisateurs ont les privilèges nécessaires pour modifier les données, les opérations ou leurs propriétés.

Liste de vérification pour la modifiabilité
Modèle de données

- Pour chaque type de changement identifié:
  - Déterminer quels abstractions de données devraient être changer,
  - Déterminer quels opérations sur les données seraient changées,
  - Déterminer quelles autres abstractions de données seraient affectées,
  - S'assurer que l'allocation des données minimise le nombre et la sévérité des modifications aux abstractions.
- Concevoir le modèle de données de façon à ce que les items alloués à chaque élément du modèle soient appelés à changer ensemble.

Liste de vérification pour la modifiabilité
Correspondance entre les éléments architecturaux

- Déterminer s'il est désirable de changer la façon dont la fonctionnalité est allouée aux éléments computationnels (processus, fils d'exécution, processeurs) à l'exécution, à la compilation, durant la conception ou durant la construction.
- Déterminer l'ampleur des modifications nécessaires pour permettre l'ajout, le retrait ou la modification d'une fonction ou d'un attribut de qualité. Ceci peut impliquer de déterminer, par exemple:
  - Les dépendances d'exécution,
  - L'assignation des bases de données,
  - L'assignation des éléments d'exécution aux processus, fils d'exécution ou processeurs.
- S'assurer que ces changements sont effectués à l'aide de mécanismes qui utilisent une liaison tardive des décisions de correspondance.

Liste de vérification pour la modifiabilité
Gestion des ressources

- Déterminer comment l'addition, le retrait ou la modification d'une responsabilité ou d'un attribut de qualité peut affecter l'utilisation d'une ressource. P. ex.:
  - Déterminer quels changements peuvent introduire des nouvelles ressources, en retirer des anciennes ou affecter l'utilisation de ressources existantes,
  - Déterminer quelles limites sur les ressources vont changer et comment.
- S'assurer que les ressources après les modifications sont suffisantes pour rencontrer les requis.
- Encapsuler tous les gestionnaires de ressource et s'assurer que les politiques implémenter par ces gestionnaires sont aussi encapsulées et que les liaisons sont retardées autant que possible.

Liste de vérification pour la modifiabilité
Choix du moment pour effectuer une liaison

- Pour chaque type de changement:
  - Déterminer le moment le plus tard où un changement devra être effectué,
  - Choisir un mécanisme pour retarder la liaison qui fournit la capacité appropriée au moment choisi,
  - Déterminer le coût d'introduire un mécanisme et le coût d'effectuer un changement en utilisant le mécanisme. Vérifier la rentabilité.
  - Ne pas introduire tellement de mécanismes de liaison que les changements soient empêchés à cause de dépendances complexes ou inconnues entre les choix.

Liste de vérification pour la modifiabilité
Choix de technologie

- Déterminer quels modifications sont rendues plus simples ou plus difficiles par vos choix de technologie:
  - ➢Vos choix de technologie vont-ils aider à effectuer, tester et déployer les changements ?
  - ➢Combien sera-t-il facile de changer vos choix de technologie (si certaines technologies changent ou deviennent désuètes) ?
- Choisir vos technologies pour supporter les changements les plus probables:
  - → P.ex. l'introduction d'un bus de service d'entreprise peut rendre facile de changer comment les éléments sont connectés, mais peut entraîner une dépendance envers un fournisseur commercial spécifique.