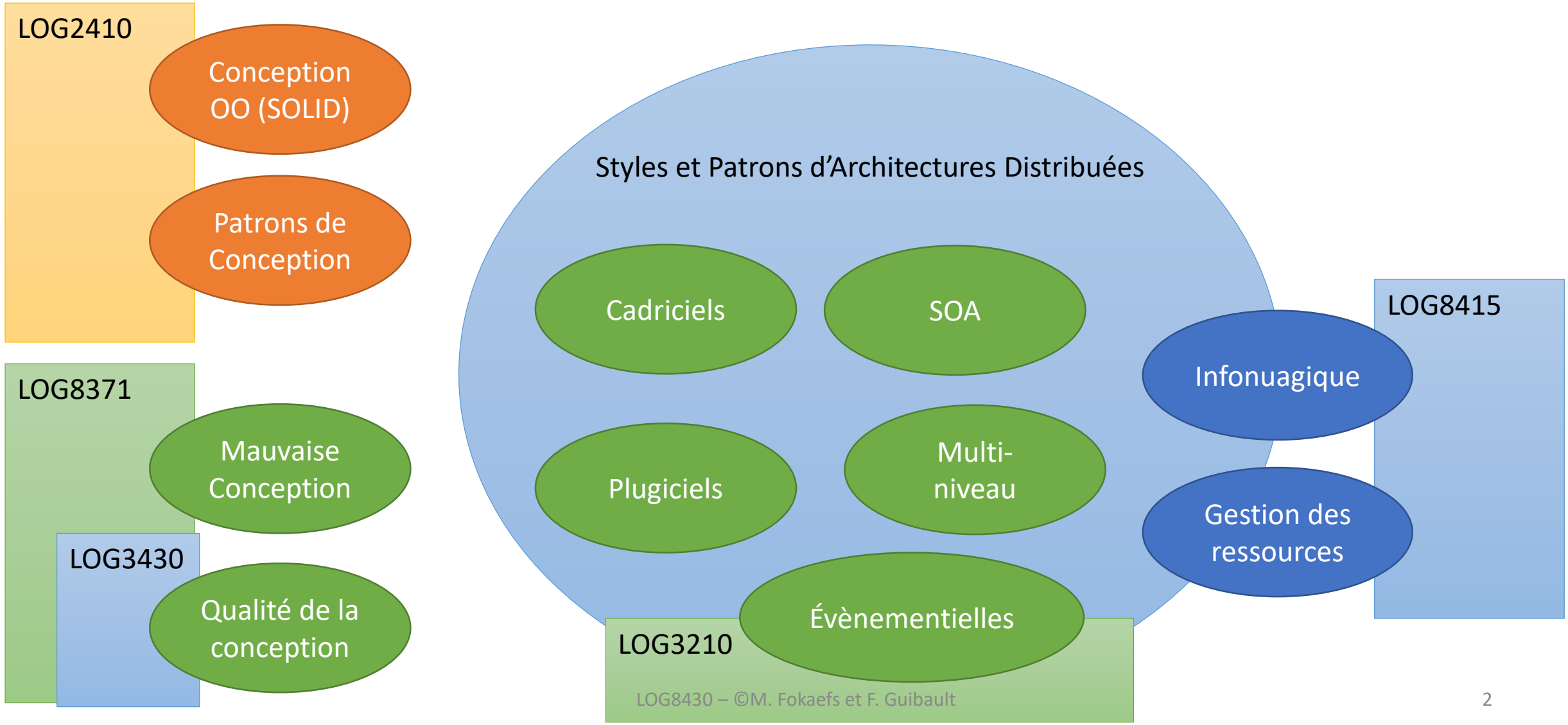
POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

LOG8430 : Attributs de qualité

Carte du cours



Précédemment

LOG2410

Conception OO (SOLID)

Patrons de Conception

Styles et Patrons d'Architectures Distribuées

Aujourd'hui

LOG2410

Conception OO (SOLID)

Patrons de Conception

LOG8371

LOG3430

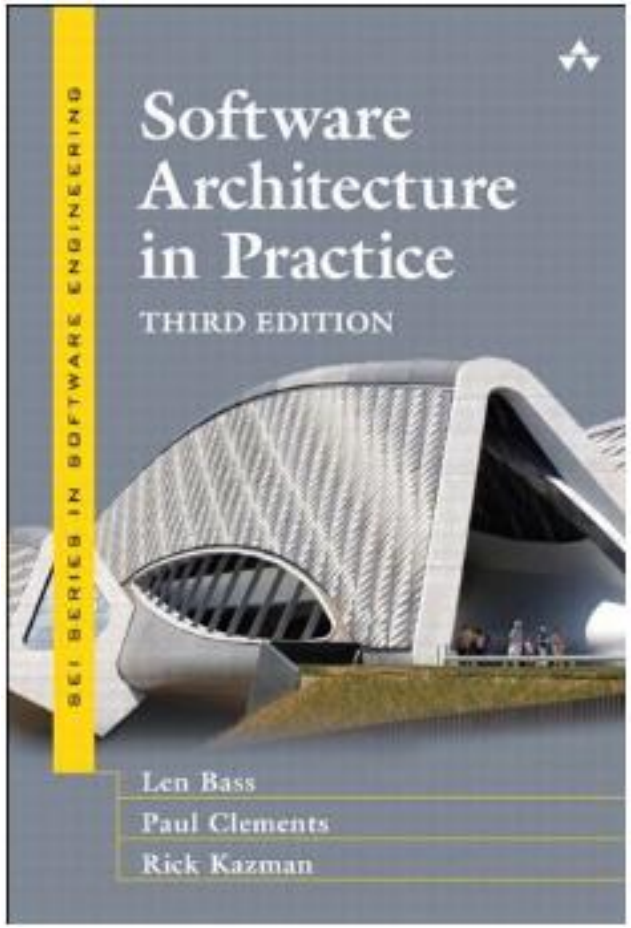
Qualité de la conception

Styles et Patrons d'Architectures Distribuées

Attributs de qualité

Le contenu de ce chapitre est tiré du livre:

Bass, L., Clements, P. et Kazman, R. *Software Architecture in Practice*. 3rd Edition. Addison-Wesley. 2013.



Attributs de qualité

Porter notre attention comme concepteurs sur les attributs non-fonctionnels de qualité.

Les systèmes logiciels sont fréquemment réimplémentés pour des raisons qui ne sont pas liées à la fonctionnalité:

- Difficultés de maintenance
- Problèmes de portabilité ou de mise à l'échelle
- Failles de sécurité
- Migration latérale de la fonctionnalité dans le processus de réimplémentation

Attributs de qualité

Un attribut de qualité est une propriété mesurable ou vérifiable (par un test) d'un système, que l'on utilise pour indiquer jusqu'à quel point un système satisfait les besoins des parties prenantes.

C'est une mesure du mérite d'un produit selon une dimension d'intérêt pour une partie prenante.

Deux catégories d'attributs de qualité: 1) des propriétés du système en cours d'exécution, 2) des propriétés du processus de développement du système.

Les attributs de qualité sont intimement liés aux requis du système.

Attributs de qualité – Un exemple

Dans le cas d'un système non-disponible en raison d'une attaque de déni de service, quel qualité du système est en cause ? S'agit-il:

- D'un problème de disponibilité ?
- D'un problème de performance ?
- D'un problème de sécurité ?
- D'un problème d'utilisabilité ?

Chaque communauté liée à ces différents attributs de qualités pourrait revendiquer en partie la propriété ou la responsabilité de trouver une solution au problème de gérer une attaque de déni de service.

Attributs de qualité – Un exemple

3 problèmes fondamentaux liés aux définitions habituelles des attributs de qualité:

1. Les définitions sont généralement difficiles à tester
2. Les discussions focalisent plus sur l'identification de la qualité qui est affectée par une situation que sur la façon d'adapter ou de modifier l'architecture pour y répondre
3. Chaque communauté a son propre vocabulaire pour parler d'un attribut de qualité

Une solution en 2 parties:

1. Utiliser des scénarios d'attributs de qualité pour caractériser chaque attribut,
2. Se concentrer sur les préoccupations sous-jacentes aux attributs de qualité pour illustrer les concepts qui sont fondamentaux à chaque attribut.

Spécification d'un requis d'attribut de qualité

Les attributs de qualité sont spécifiés en quatre parties:

1. Stimulus: description d'un évènement affectant le système.
2. Source du stimulus: origine du stimulus. L'origine du stimulus peut modifier la façon d'y répondre.
3. Réponse: description de la réaction attendue du système (exécution) ou des développeurs (développement).
4. Mesure de la réponse: déterminer si la réponse est satisfaisante.

Deux autres caractéristiques importantes:

1. Environnement: l'ensemble de circonstances dans lesquelles le scénario se déroule. Agit comme qualificatif du stimulus.
2. Artéfact: portion du système auquel s'applique le requis.

Exemple: un scénario général de disponibilité

Source du stimulus

Interne/externe: personne, matériel, logiciel, infrastructure physique, environnement physique

Stimulus

Erreur: omission, plantage, timing incorrect, réponse incorrecte

Artéfact

processeurs, canaux de communication, stockage persistant, processus

Environnement

Opération normale, démarrage, séquence d'arrêt, mode de réparation, mode d'opération restreint, surcharge

Réponse

Empêcher l'erreur d'entraîner une panne
Détection d'erreur: enregistrement, avertissement
Récupération d'erreur: désactivation de la source d'erreur, rendre non-disponible, réparer/masquer, entrer en mode d'opération restreint

Mesure de la réponse

Temps ou intervalle de disponibilité du système
Pourcentage de disponibilité
Temps en mode restreint
Temps de détection d'erreur
Temps de réparation
Proportion des erreurs traitées par le système

Atteindre des attributs de qualité: les tactiques

En concevant un système, comment atteindre des attributs de qualité ?

En appliquant un ensemble de techniques spécifiques: des tactiques architecturales.

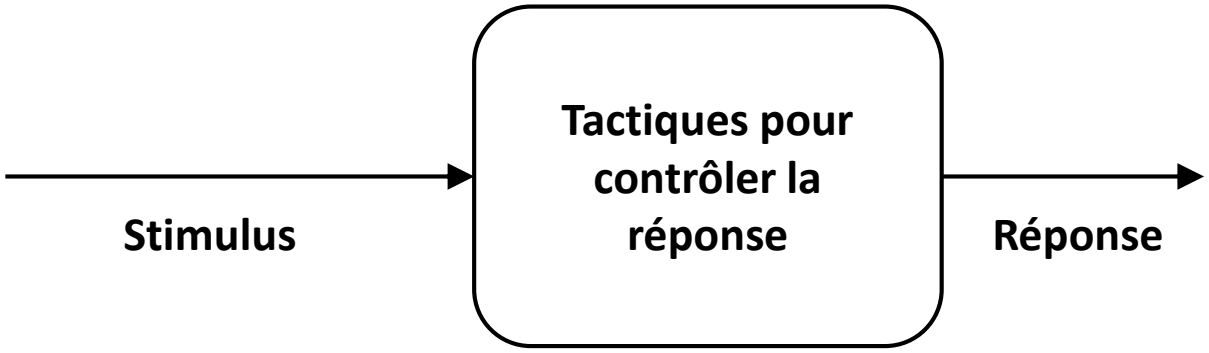
Une tactique architecturale est une décision de conception qui influence l'atteinte d'une réponse d'un attribut de qualité.

Une tactique affecte directement la réponse d'un système à un stimulus

Atteindre des attributs de qualité: les tactiques

Une tactique focalise sur un seul attribut de qualité.

- Pas de compromis pour d'autres attributs
- C'est au concepteur de considérer/contrôler les compromis
- Différent d'un patron architectural, qui incorporent par construction la notion de compromis
- Les tactiques sont des techniques éprouvées qui capturent les pratiques développées par des architectes logiciels aux travers des ans



Guider les décisions de conception pour la qualité

Une classification des décisions de conception architecturale

1. Allocation des responsabilités
2. Modèle de coordination
3. Modèle de données
4. Gestion des ressources
5. Correspondance entre les éléments architecturaux
6. Choix du moment où un aspect variable est fixé (*binding time*)
7. Choix de technologie

Décisions de conception: Allocation des responsabilités

Les décisions impliquant l'allocation de responsabilités incluent:

- Identification des responsabilités importantes: fonctions du système, infrastructure architecturale, satisfaction des attributs de qualité.
- Détermination de la façon dont ces responsabilités sont allouées aux éléments non-exécutables et d'exécutables (modules, composantes, connecteurs).

Des stratégies pour prendre ce type de décisions incluent:

- La décomposition fonctionnelle
- La modélisation d'objets du monde réel
- Les regroupements basés sur les modes d'opération importants
- Les regroupements basés sur des requis de qualité similaires

Décisions de conception: Modèle de coordination

Un système logiciel fonctionne par l'interaction de ses éléments au travers de mécanismes conçus à cette fin: l'ensemble de ces mécanismes forment le modèle de coordination.

Les décisions permettant de décider d'un modèle de coordination incluent:

- Identification des éléments du système qui doivent ou ne doivent pas assumer un rôle de coordination.
- Détermination des propriétés de coordination: précision, concurrence, complétude, exactitude et consistance.
- Choix des mécanismes de communication: avec ou sans état, synchrone ou asynchrone, garantis ou non, propriétés de performance (débit, latence,…).

Décisions de conception: Modèle de données

Tout système logiciel doit représenter des données de façon interne. Le modèle de données est l'ensemble des représentations des données et la façon de les interpréter.

Les décisions permettant de décider d'un modèle de données incluent:

- Choix des principales abstractions, de leurs opérations, de leurs propriétés: mécanismes de création, d'initialisation, d'accès, de persistance, de manipulation, de conversion et de destruction.
- Compilation des métadonnées nécessaires à l'interprétation correcte de données.
- Organisation des données: base de données, collection d'objets ou les deux. Correspondance entre les représentation dans la base de données et les objets

Décisions de conception: Gestion des ressources

C’est au niveau architectural que se décide l’arbitrage dans l’utilisation de ressources partagées par un système.

Les décisions liées à la gestion des ressources incluent:

- Identification des ressources qui doivent être gérées et détermination de leurs limites/capacité.
- Détermination des éléments responsables de gérer chaque ressource.
- Détermination des règles de partage et des stratégies d’arbitrage en cas de pénurie.
- Détermination de l’impact de la saturation de différentes ressources. Par exemple saturation du CPU vs. saturation de la mémoire.

Décisions de conception: Correspondance entre les éléments architecturaux

Une architecture doit établir deux types de correspondances:

1. Correspondance entre des éléments de types différents au sein de l'architecture: p. ex. correspondance entre une unité de développement (un module) et une unité d'exécution (un processus).
2. Correspondance entre des éléments logiciels et des éléments de l'environnement: p. ex. correspondance entre des processus et des CPU spécifiques.

Des correspondances utiles incluent:

- La correspondance entre des modules et des éléments d'exécution
- La correspondance entre des éléments d'exécution et des processeurs
- La correspondance entre des items du modèle de données et les entrepôts de données
- La correspondance entre les modules et les éléments d'exécution et les unités de livraison

Décisions de conception:
Choix du moment où un aspect variable est fixé

Chaque aspect variable d'un système doit être fixé afin de rendre le système exécutable. Le moment où un aspect variable est fixé est une décision architecturale.

Les décisions liées au choix du moment où un aspect variable est fixé incluent :

- L'enveloppe des choix,
- Le point dans le cycle de vie
- Le mécanisme pour réaliser l'aspect variable.

Décisions de conception:
Choix du moment où un aspect variable est fixé

Les décisions dans les six autres catégories ont toutes un moment où les choix possibles sont fixés qui leur est associé.

- Pour l'allocation des responsabilités, on peut utiliser une sélection des modules au moment de la construction via un fichier makefile.
- Pour le choix d'un modèle de coordination, on peut implanter un mécanisme de négociation de protocole à l'exécution.
- Pour la gestion des ressources, on peut concevoir un mécanisme de connexion de nouveaux périphériques à l'exécution, incluant le téléchargement et l'installation de pilotes.
- Pour le choix de technologie, on peut construire un magasin d'application qui sélectionne automatiquement la version appropriée d'une application en fonction du type du dispositif du client.

Décisions de conception: Choix de technologie

Toutes les décisions architecturales doivent éventuellement être réalisées en utilisant une technologie spécifique.

Si les choix technologiques ne sont pas fixés a priori, les décisions liées au choix de technologie incluent:

- Décider quelles technologies sont disponibles pour réaliser les décisions prises dans les autres catégories.
- Déterminer quels outils sont disponibles pour supporter les choix de technologie et si ces outils sont adéquats.
- Déterminer le niveau de familiarité interne et de support externe disponible pour la technologie et si ce niveau est suffisant.
- Déterminer les effets de bord liés à un choix de technologie
- Déterminer la compatibilité d'une nouvelle technologie avec l'environnement technologique existant.

Attributs de qualité

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

Attribut de qualité 1: Disponibilité

La disponibilité réfère à la propriété d'un système logiciel d'être là et prêt à réaliser des tâches au moment où l'on en a besoin.

On réfère souvent à cette propriété par le terme de fiabilité, mais la disponibilité inclut des considérations supplémentaires liées entre autres aux arrêts périodiques pour entretient.

La disponibilité ajoute à la notion de fiabilité celle de récupération.

La disponibilité réfère à la capacité d'un système de masquer ou de corriger ses erreurs de façon à ce que la durée cumulative d'absence de service ne dépasse pas une valeur spécifié au cours d'un intervalle de temps donné.

Attribut de qualité 1: Disponibilité

La disponibilité est étroitement liée à la sécurité:

- Une attaque de déni de service vise directement à rendre un système non disponible.

La disponibilité est étroitement liée à la performance:

- Il est très difficile de faire la différence entre un système extrêmement lent et un système qui n'est pas disponible.

La disponibilité est étroitement liée à la sureté:

- Un système hautement disponible est capable de détecter lorsqu'il entre dans un état instable ou dangereux et est en mesure de limiter les dégâts si cela arrive.

Attribut de qualité 1: Disponibilité

Une défaillance est une déviation du système de ses spécifications qui est observable par un humain.

La cause d'une défaillance est appelée une faute.

Les états intermédiaires entre l'apparition d'une faute et l'apparition d'une défaillance sont appelés des erreurs.

La disponibilité vise à minimiser la durée des arrêts de services par la mitigation des fautes.

Attribut de qualité 1: Disponibilité

Les principales considérations à examiner à propos des fautes incluent:

- Comment détecter les fautes ?
- À quelle fréquence les fautes se produisent-elles?
- Qu'est-ce qui arrive lorsqu'une faute se produit?
- Combien de temps est-il permis que le système reste hors service ?
- Quand des fautes ou des défaillances peuvent-elles se produire de façon sécuritaire?
- Comment les fautes peuvent-elles être évitées?
- Quel sorte d'avertissement est utile lorsqu'une faute se produit?

Il faut aussi se préoccuper du niveau de capacité du système suite à une faute: en mode d'opération dégradé.

Attribut de qualité 1: Disponibilité

La disponibilité d'un système matériel peut être calculée comme la probabilité que le système fournisse les services spécifiés à l'intérieur d'une marge prescrite pour une durée de temps spécifiée.

$$\frac{MTBF}{(MTBF + MTTR)}$$

Avec *MTBF=Mean Time Between Failures,* Temps moyen entre les défaillances, et *MTTR=Mean Time To Repair,* Temps moyen de réparation.

Pour les systèmes logiciels, il faut interpréter cette formule en évaluant ce qui peut amener une défaillance du système, quelle est la probabilité que cela arrive et qu'il faudra du temps pour la réparer.

Attribut de qualité 1: Disponibilité

Le niveau de disponibilité d'un système logiciel est souvent exprimé au travers d'un accord de niveau de service (service-level agreement ou SLA).

Un SLA précise le niveau de disponibilité garanti et les pénalités associées à un non-respect du niveau de service par le fournisseur.

Disponibilité	Durée d'arrêt/90 jours	Durée d'arrêt/an
99.0%	21 heures, 36 minutes	3 jours, 15.6 heures
99.9%	2 heures, 10 minutes	8 heures, 0 minutes, 46 secondes
99.99%	12 minutes, 58 secondes	52 minutes, 34 secondes
99.999%	1 minute, 18 secondes	5 minutes, 15 secondes
99.9999%	8 secondes	32 secondes

Attribut de qualité 1: Disponibilité

Planifier pour les défaillances

La meilleur façon de concevoir un système hautement disponible est d'analyser les sources de défaillance et de les gérer correctement.

Il faut comprendre les types de défaillances auxquels le système est principalement sujet et évaluer les conséquences de chacune.

Attribut de qualité 1: Disponibilité

Analyse de danger. On peut catégoriser les dangers qui peuvent se produire en cours d'opération. Le standard DO-178B en aéronautique propose les catégories suivantes, selon leur effet sur l'appareil, l'équipage et les passagers:

- Catastrophique: peut causer un écrasement, la perte de fonctions critiques.
- Dangereux: grand impact négatif sur la sécurité ou la performance. Peut entraîner une charge de travail inacceptable pour l'équipage et des blessures sérieuses ou fatales aux passagers.
- Majeur: impact négatif important. Peut entraîner une charge de travail importante pour l'équipage qui affecte la sécurité ou un inconfort important des passagers.
- Mineur: impact mineur qui peut être perçu mais qui n'entraîne que des inconvénients pour l'équipage ou les passagers.
- Pas d'effet: pas d'impact sur la sécurité de l'appareil, de l'équipage ou des passagers.

Attribut de qualité 1: Disponibilité

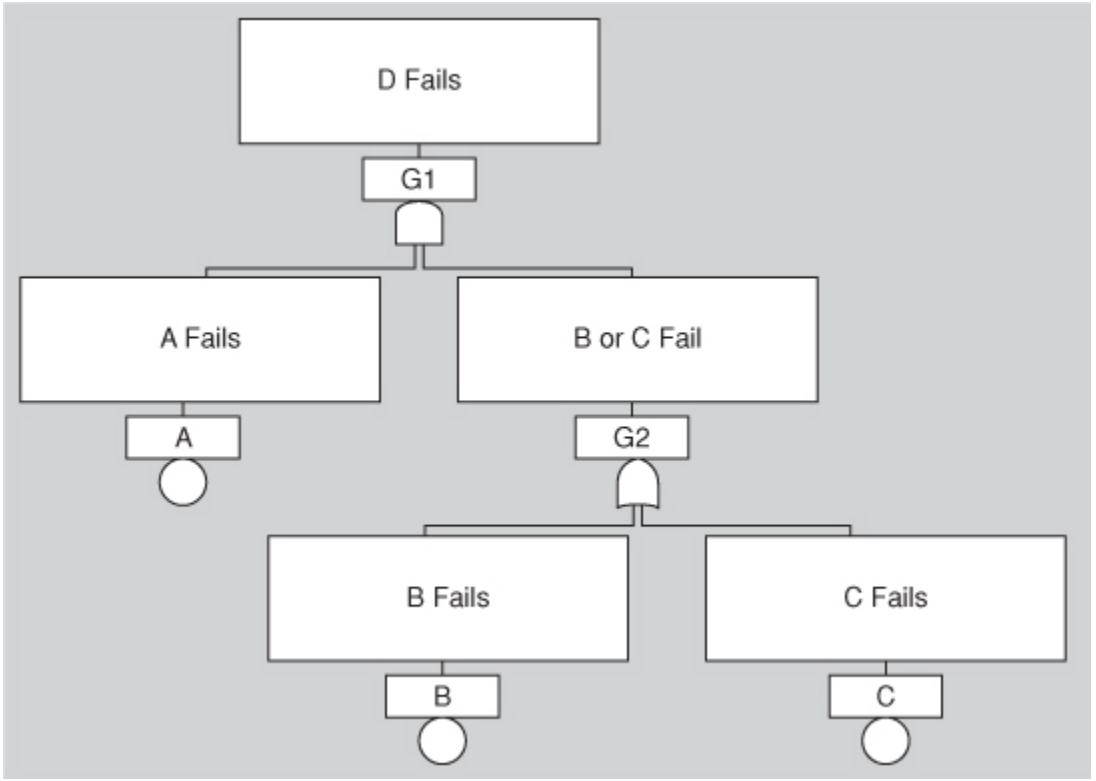
Arbre d'analyse des fautes.

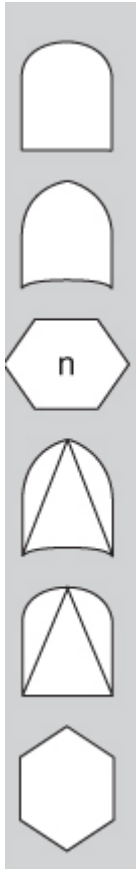
Technique analytique qui spécifie un état du système qui a un impact sur la sureté ou la fiabilité et qui analyse le contexte et les opérations du système pour déterminer toutes les façons par lesquelles cet état peut être atteint.

Méthode basée sur la construction graphique d'un arbre qui sert à identifier toutes les séquences parallèles et séquentielles de fautes qui peuvent mener à l'état non-désiré du système.

Attribut de qualité 1: Disponibilité

Arbre d'analyse des fautes.





ET

OU

n

COMBINAISON de n entrées

XOR

ET AVEC PRIORITÉ

INHIBE

Attribut de qualité 1: Disponibilité

L'arbre de fautes permet d'effectuer plusieurs types d'analyses:

- Les *ensembles de coupes minimales* permettent de trouver le sous-ensemble minimal de fautes qui vont causer une défaillance.
- Tous les ensembles de coupes minimales indiquent toutes les façon dont des fautes peuvent se combiner pour entraîner une défaillance.
- Un ensemble de coupes minimales qui a un seul élément permet d'identifier un point unique de défaillance.
- Des probabilités peuvent être assignées aux différentes fautes pour obtenir une probabilité d'apparition de la défaillance étudiée.

Scénario général de disponibilité

Source du stimulus

Interne/externe: personne, matériel, logiciel, infrastructure physique, environnement physique

Stimulus

Erreur: omission, plantage, timing incorrect, réponse incorrecte

Artéfact

processeurs, canaux de communication, stockage persistant, processus

Environnement

Opération normale, démarrage, séquence d'arrêt, mode de réparation, mode d'opération restreint, surcharge

Réponse

Empêcher l'erreur d'entraîner une panne
Détection d'erreur: enregistrement, avertissement
Récupération d'erreur: désactivation de la source d'erreur, rendre non-disponible, réparer/masquer, entrer en mode d'opération restreint

Mesure de la réponse

Temps ou intervalle de disponibilité du système
Pourcentage de disponibilité
Temps en mode restreint
Temps de détection d'erreur
Temps de réparation
Proportion des erreurs traitées par le système

Exemple de scénario concret de disponibilité

Source du stimulus

Moniteur de type battement de cœur

Stimulus
Serveur qui ne répond pas

Artéfact
processus

Environnement
Opération normale

Réponse
Informer un opérateur
Continuer les opérations normales

Mesure de la réponse
Pas d'arrêt de service

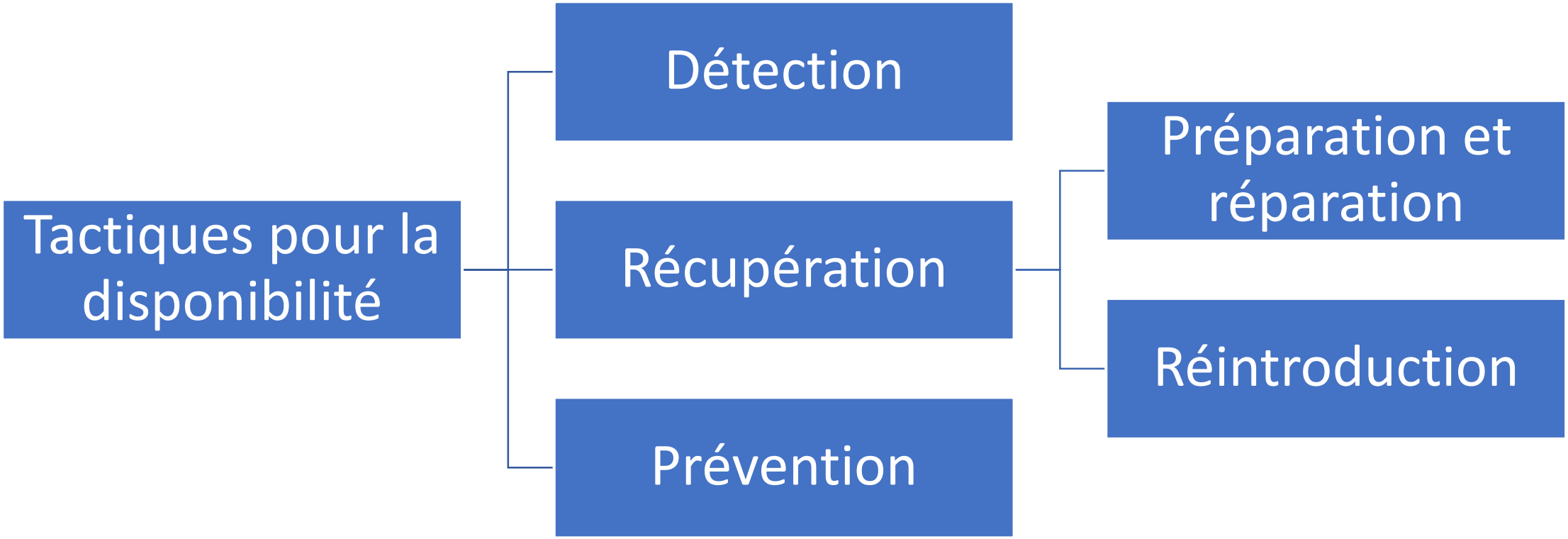
Attribut de qualité 1: Tactiques pour la disponibilité

Une défaillance se produit lorsque le système ne fournit plus un service qui est consistant avec ses spécifications. Cette défaillance est observable par les acteurs du système.

Les tactiques pour la disponibilité sont conçues pour permettre à un système de gérer les fautes et d'éviter les défaillances.

Les tactiques visent à éviter que les fautes entraînent une défaillance ou à limiter les effets de la faute et permettre une réparation.

Attribut de qualité 1: Tactiques pour la disponibilité



Attribut de qualité 1: Tactiques pour la disponibilité: **Détection**

Tactiques pour détecter ou anticiper la présence de fautes

- Ping/Écho
- Moniteur
- Battement de cœur
- Horodatage (*timestamp*)
- Vérification de cohérence
- Monitorage d'état
- Votation: réplication, redondance fonctionnelle, redondance analytique
- Détection d'exception: exceptions système, barrière de paramètre, typage de paramètre, temps mort
- Auto-vérification

Attribut de qualité 1: Tactiques pour la disponibilité: **Récupération par préparation et réparation**

Tactiques pour récupérer d'une faute par préparation et réparation

- Redondance active
- Redondance passive
- Rechange
- Traitement d'exception
- Retour arrière
- Mise à jour logicielle
- Nouvel essai
- Comportement ignoré
- Dégradation
- Reconfiguration

Attribut de qualité 1: Tactiques pour la disponibilité: **Récupération par réintroduction**

Tactiques pour récupérer d'une faute par réintroduction

- Opération en mode d'ombre
- Resynchronisation d'état
- Redémarrage en paliers
- Retransmission sans interruption

Attribut de qualité 1: Tactiques pour la disponibilité: **Prévention des fautes**

Tactiques pour prévenir les fautes

- Retrait de service
- Transactions
- Modèle prédictif
- Prévention d'exception
- Accroissement de l'ensemble de compétences

Liste de vérification pour la disponibilité Allocation des responsabilités

- Déterminer quelles responsabilités systèmes doivent être hautement disponibles.
- Parmi ces responsabilités, s'assurer que des responsabilités additionnelles ont été ajoutées pour détecter les omissions, plantages, minutages incorrects ou réponses incorrectes.
- S'assurer que des responsabilités sont liées à:
  - Conserver une trace des erreurs
  - Avertir les entités appropriées
  - Désactiver les sources d'évènements causant des fautes
  - Être temporairement indisponible
  - Réparer ou masquer les fautes/défaillances
  - Opérer en mode dégradé

Liste de vérification pour la disponibilité Modèle de coordination

- Pour les responsabilités systèmes qui doivent être hautement disponibles, s'assurer que:
  - Les mécanismes de coordination peuvent détecter les omissions, plantages, minutages incorrects ou réponses incorrectes. Par ex. examiner si la livraison garantie de message est nécessaire. La coordination peut-elle fonctionner en cas de condition d'opération dégradée?
  - Les mécanismes de coordination permettent de conserver une trace des erreurs, d'avertir les entités appropriées, de désactiver les sources d'évènements causant des fautes, de réparer ou masquer les fautes, ou d'opérer en mode dégradé.
  - Le modèle de coordination supporte le remplacement des artéfacts utilisés (processeurs, canaux de communication, stockage persistant et processus). Par ex. est ce que le remplacement d'un serveur permet au système de continuer à s'exécuter?
  - La coordination va fonctionner dans des conditions de communication dégradées, pendant le démarrage ou l'arrêt, en mode de réparation ou en surcharge. Combien de perte d'information le modèle de coordination peut-il tolérer?

Liste de vérification pour la disponibilité Modèle de données

- Pour les portions du système qui doivent être hautement disponibles, déterminer quelles abstractions de données, avec leurs opérations et leurs propriétés, pourrait causer une faute par omission, un plantage, un minutage incorrect ou une réponse incorrecte.
- Pour ces abstractions, s'assurer qu'elles peuvent être désactivées, être temporairement non disponibles ou être réparées ou masquées en cas de faute.
- Par exemple, s'assurer que les requêtes en écriture sont mise en cache si un serveur est temporairement indisponible et effectuées lorsque le serveur est remis en service.

Liste de vérification pour la disponibilité Correspondance entre les éléments architecturaux

- Déterminer quels artéfacts (processeurs, canaux de communication, stockage persistant ou processus) peuvent causer une faute: par omission, plantage, minutage incorrect ou réponse incorrecte.
- S’assurer que la correspondance entre les éléments architecturaux est suffisamment flexible pour permettre la récupération en cas de faute.

Liste de vérification pour la disponibilité Correspondance entre les éléments architecturaux

- Considérer par exemple:
  - Quels processus s'exécutant sur un processeur défectueux doivent être réassignés en cours d'exécution.
  - Quels processeurs, entrepôts de données ou canaux de communication peuvent être activés ou réassignés en cours d'exécution.
  - Comment les données sur des processeurs ou entrepôts de données défectueux peuvent être rendus disponibles par des unités de remplacement.
  - À quelle vitesse un système peut-il être réinstallé en se basant sur les unités de livraison fournies.
  - Comment réassigner en cours d'exécution des éléments à des processeurs, des canaux de communication et des entrepôts de données.
  - Lorsque des tactiques utilisant de la redondance sont employées, la correspondance des modules avec les composantes redondantes est importante. Par ex. il est possible d'écrire un module qui contienne du code approprié à la fois pour les composantes actives et de secours.

Liste de vérification pour la disponibilité Gestion des ressources

- Déterminer quelles ressources critiques sont nécessaires pour poursuivre les opérations en présence d'une faute: omission, plantage, minutage incorrect ou réponse incorrecte.
- S'assurer qu'il y a suffisamment de ressources disponibles en cas de faute pour:
  - garder une trace de la faute,
  - avertir les entités appropriées,
  - désactiver les sources d'évènements causant la faute,
  - être temporairement indisponible,
  - réparer ou masquer la faute/défaillance,
  - opérer normalement durant le démarrage, l'arrêt, en mode de réparation, en mode d'opération dégradé ou en surcharge.

Liste de vérification pour la disponibilité Gestion des ressources

- Déterminer le temps de disponibilité des ressources critiques. Quelles ressources critiques doivent être disponibles durant quel intervalle de temps, pendant lequel les ressources critiques peuvent opérer en mode dégradé, et le temps de réparation des ressources critiques. S'assurer que les ressources critiques sont disponibles pendant ces intervalles de temps.
- Par exemple s'assurer qu'une queue d'entrée est suffisamment grande pour accumuler les messages prévus si un serveur tombe en panne, de façon à ne pas perdre définitivement les messages.

Liste de vérification pour la disponibilité
Choix du moment où un aspect variable est fixé

- Déterminer quand et comment les éléments architecturaux sont fixés.
- Si les éléments sont fixés au moment de l'exécution afin de permettre de choisir parmi différentes composantes qui peuvent elles-mêmes être source de fautes (p.ex. processus, processeurs, canaux de communication), s'assurer que la stratégie de disponibilité choisie est suffisante pour couvrir les fautes introduites par toutes les sources.

Liste de vérification pour la disponibilité
Choix du moment où un aspect variable est fixé

- Par exemple:
  - Si le choix entre différents artéfacts, comme des processeurs, est effectué à l'exécution et que ces artéfacts devront traiter ou peuvent générer des fautes, est-ce que les mécanismes de détection et de récupération des fautes vont fonctionner pour tous les choix possibles ?
  - Si le choix de la définition d'un seuil ou d'une tolérance de ce qui constitue une faute est fixé à l'exécution (par exemple combien de temps un processus peut s'exécuter sans répondre et sans que l'on suppose qu'il y a une faute), est-ce que la stratégie de récupération choisie est en mesure de traiter tous les cas ? P. ex. si une faute est signalée après 0.1 ms mais que le temps de récupération est de 1.5 s, il peut s'agir d'un décalage inacceptable.
  - Quelles sont les caractéristiques de disponibilité du mécanisme de sélection lui-même ? Peut-il échouer ?

Liste de vérification pour la disponibilité Choix de technologie

- Identifier les technologies disponibles qui peuvent (aider à) détecter les fautes, récupérer des fautes ou réintroduire les composantes ayant échoué.
- Identifier les technologies disponibles qui peuvent aider à répondre à une faute (par ex. un enregistreur de traces).
- Identifier les caractéristiques de disponibilité des technologies elles-mêmes: de quelles fautes peuvent-elles récupérer? Quelles fautes peuvent-elles introduire dans le système?