Institut catholique des arts et métiers
Université catholique d'Afrique centrale - Université Loyola du Congo

Les modèles et bases de données des système d'information

Introduction

MCED_SIBD_00

Christina KHNAISSER *(christina.khnaisser@usherbrooke.ca)*

Luc LAVOIE *(luc.lavoie@usherbrooke.ca)*

(les auteurs sont cités en ordre alphabétique nominal)

CoFELI/Scriptorum/SIBD_00-Introduction (v100), version 1.0.0.a, en date du 2024-04-28

— document de travail —

Sommaire

- Introduction à la problématique de la mise à l'échelle organisationnelle (ou plus grande) des modèles de données.
- Examen des solutions qui n'en furent pas.
- Identification de la cause première de la majorité des problèmes.
- Étude des solutions qui en découlent.

Préalables stricts

- IFT187 ou INFO221 (Éléments de base de données)

Préalables recommandés

- SI_00-Introduction aux systèmes d'information

Plan

Introduction

Le présent documnet présente un recensement des problèmes rencontrés lors du développement et de l'exploitation de systèmes d'information.

Après une courte présentation de ceux-i, ils renvoient à des modules complémentaires où ils sont traités de façon plus complète.

1. Problèmes connus

Nous présentons ci-après quelques problèmes connus susceptibles de compromettre l'adéquation des systèmes d'information Tous ont un impact potentiel sur chacun des huit critères, directement (XX) ou indirectement (—).

Tableau 1. Matrice d’impact des problèmes présentés en regard de l’adéquation

Problème	Cohérence	Validité	Efficacité	Évolutivité	Efficience	Complétude	Réfutabilité	Éthique
Couplage	XX	XX	XX	XX	—	—	—	—
Duplication	—	—	—	XX	—	XX	XX	XX
Testabilité	XX	XX	XX	XX	—	XX	XX	XX
Injection	XX	XX	XX	—	XX	XX	XX	XX
Authentification	—	—	—	—	—	—	—	XX
Contrôle d’accès	—	XX	XX	—	—	XX	XX	XX
Gestion du cycle de vie	XX	XX	XX	XX	XX	XX	XX	XX

1.1. Couplage

Définition

- Le couplage consiste à lier deux entités entre elles.

Discussion

- Un couplage n'est pas une erreur en soi.
- En fait, sans couplage, aucun raisonnement n'est possible.
- Par contre, des couplages incorrects ou contradictoires mèneront à des raisonnements faux, ils doivent donc être éliminés.
- Par ailleurs, les couplages inutiles ou redondants rendront tout modèle plus difficile à modifier, puisqu'il faudra s'assurer d'en maintenir la cohérence en cas de modification ;
- À défaut de mettre en place un automatisme pour ce faire, il est très fortement recommandé de les éliminer.

Exemple de couplage entre LC et BDS

- Dans un LC, concevoir une transaction contenant une instruction SELECT tributaire de la BDS (la structure des tables, les différentes clés et autres contraintes et assertions).
- Le LC devient dès lors dépendant de chacun des éléments structurels référés et la BDS ne peut être librement modifiée sans considérer l'impact sur le LC.

Exemple de couplage interne à la BDS

- Les dépendances fonctionnelles présentent une autre opportunité de couplage traité au sein même de la BDS grâce à la normalisation et aux actions compensatoires.

Conséquences potentielles (partielles)

- À travers les couplages, tout changement à la BDS impacte potentiellement tous les LC :
  - explosion de l'envergure de travail requis lors d'un changement ou d'une évolution ;
  - nécessité de devoir synchroniser la modification des LC impactés.
- La situation est d'autant plus préoccupante que de nombreuses organisations peinent à maintenir l'inventaire des LC tributaires de chacune de leurs BD.
- En pratique, certaines modifications pourraient ne pas pouvoir être reportées jusqu'à ce que tous les LC soient adaptés (faute de ressources humaines, en cas d'urgence, etc.).

Justifications parfois avancées

- « On a pas le temps (les ressources, l'expertise, les procédés…) pour mettre en oeuvre une solution découplée. » (ÇA SE PALLIE)
- « SQL n'a pas de modules, de fonction ni de procédures. » (C'EST FAUX)

Défis

- Maintenir la cohérence avec un couplage minimal.
  - Sans le cas d'une BD, la cohérence adéquate est le maintien de l'intégrité des données, même dans un contexte transactionnel et en cas de panne (ce qui inclut le maintien des propriétés ACID).
- Définir la minimalité dans le cas du couplage.
  - Quels sont les couplages acceptables, superflus, inacceptables ?
  - Comment procéder au choix entre deux ensembles de couplages acceptables ?
- Mise en place simple, efficace et expressive.

1.1.1. Duplication (des données, des modèles…)

Définition

- Faire un duplicata (un double) d'une source (d'un jeu de données, d'une BD, d'un SGBD, d'un SI…) dans le but d'en permettre l'utilisation indépendante de l'original par des tiers.

Exemples

- DataMart avec ou sans Entrepôt
- DataLake avec ou sans DataLake House

Conséquences potentielles

- Les données sont souvent mises à disposition
  - sans les modèles,
  - sans les guides d'interprétation contextuelle,
  - sans vérification, validation, ni certification,
  - sans possibilité de maintenance,
  - sans règles d'accès, etc.
- Plusieurs sources peuvent partager des parties d'un même modèle sans qu'il soit possible de le déceler, encore moins de le démontrer.
- Une énorme duplication de l'effort d'arrimage parmi les intervenants, les hypothèses avancées par chacun introduiront des biais ; des divergences, voire des contradictions dans les données elles-mêmes.
- Un très grand degré d'incertitude quant à l'information contenue et aux

résultats inférés

Justifications avancées

- Accélérer la mise à disposition des données.
- Offrir une pluralité de modèles aux analystes.

Défis

- Transmettre l'intégralité des connaissances, des connaissances, des modèles, des procédés, des pratiques et des usages requis au bon usage du duplicata
- Maintenir l'original et le duplicata en phase tout au long de leur évolution represcrive.
- Qui a dit qu'un duplicata ne pouvait être dupliqué à nouveau ?

1.2. Testabilité

Définition

La testabilité d'un système est le taux de couverture réalisé par un ensemble de tests et de jeux d'essais. Il existe par ailleurs plusieurs types de mesures de taux de couverture. Par exemple, couverture des fonctions (routines), des instructions, des points de tests et des chemins d'exécution. En particulier, la méthode MC/DC (voir les normes ED-12c et DO-178c) est prescrite par la plupart des protocoles applicables aux logiciels critiques.

Exemples

Exposé présenté en classe

Conséquences potentielles

Exposé présenté en classe

Justifications avancées

Exposé présenté en classe

Défis

Exposé présenté en classe

1.3. Injection

Définition

En informatique, l'injection consiste à introduire lors de l'exécution d'un logiciel des instructions dans celui-ci par l'entremise de mécanismes de communication prévu et contrôlé par le logiciel, puis de faire exécuter ces instructions.

Discussion

Ceci est généralement rendu possible par l'exploitation d'une faille de sécurité et l'existence d'un mécanisme d'interprétation d'instructions au sein du langage dans lequel est écrit le logiciel (ou de la présence d'un tel mécanisme développé au sein du logiciel lui-même).

Les langages «interprétés», sans possibilité de contrôle permanent de l'origine et de l'intégrité du code exécutable (CPOIC), (BASIC, PHP, Javascript, TypeScript, Python, etc.) permettent la création de logiciels particulièrement exposés à ce genre d'attaques. Pour cette raison, ils sont exclus des outils autorisés dans le développement de logiciels critiques.

Dans le cas logiciels produits grâce à des langages «compilables» dans le jeu d'instructions natives du processeur d'exécution (comme C, Cobol, FORTRAN, C++, Ada), en plus de la présence d'un mécanisme de CPOIC, il sera exigé que toute instruction d'interprétation dynamique (de type EXEC) soient totalement absente du code.

Pour les logiciels utilisant un langage «mixte» pouvant faire appel à des instructions exécutées par un processeur virtuel (Java, SQL, etc.), mais pour lequel un mécanisme CPOIC est disponible, des dispositions *ad hoc* doivent être prévues.

Exemples

Exposé présenté en classe

Conséquences potentielles

Exposé présenté en classe

Justifications avancées

Exposé présenté en classe

Défis

Exposé présenté en classe

1.4. Contrôle d’accès

- Voir CoFELI:MCED/SGBD_12-Controle-d-acces

1.4.1. Authentification

- Voir CoFELI:MCED/SGBD_13-Authentification

1.5. Gestion du cycle de vie

- Voir CoFELI:MCED/SGBD_11-Gestion-du-cycle-de-vie

1.5.1. Gestion des versions

- Voir CoFELI:MCED/SGBD_11-Gestion-du-cycle-de-vie

1.5.2. Gestion des configurations

- Voir CoFELI:MCED/SGBD_11-Gestion-du-cycle-de-vie

1.5.3. Gestion de l’alimentation

- Voir CoFELI:MCED/SGBD_11-Gestion-du-cycle-de-vie

1.5.4. Gestion des migrations

- Voir CoFELI:MCED/SGBD_11-Gestion-du-cycle-de-vie

1.6. Multiplicité des modèles, interfaces et services

- Voir CoFELI:DESI/SILC/SILC_03-Architectures

Conclusion

Les SI sont parmi les systèmes lesplus complexes développés par les humains. Certains contrôlent de larges secteurs d'activité à l'échelle planétaire (gestion de l'environnement, de la santé publique, des communications, des transports, des secteurs agricoles, énergétiques, financiers, militaires, industriels, manufacturiers, etc.). Plusieurs SI sont devenus « irremplaçables », dans le sens où leur dysfonctionnement prolongé entrainerait des pertes incommensurables en vies humaines et une régression globale des conditions de vie sur la planète.

Bien que plusieurs de ces systèmes soient en fin de vie utile, sévèrement déficients, notoirement incomplets, ce sujet ne suscite plus guère d'intérêt dans les sphères gouvernementales, scientifiques, sociales et encore moins politiques et financières.

En effet, qui a besoin d'informations et de raisonner quand il suffit de juger selon « moi, maintenant ».

Produit le 2025-05-04 20:50:25 UTC

Institut catholique des arts et métiers
Université catholique d'Afrique centrale - Université Loyola du Congo