LOG210

PLAN PROPOSITION S2021/2

Site web actuel Lidéal serait de générer un site web directement à partir du contenu du cours. Moodle ne contiendrait que les quiz.

- Notes de cours version html#dss

version pdf de ce document: remote

version pdf de ce document:local

LOG210

PLAN DES SÉANCES

Séance 1 | Séance 2 | Séance 3 | Séance 4 | Séance 5 | Séance 6 | Séance 7 | Séance 8 | Séance 9 | Séance 10 | Séance 11 | Séance 12 | Séance 13 |

SÉANCE 1

SÉANCE 2

SÉANCE 3

SÉANCE 4

SÉANCE 5

SÉANCE 6

SÉANCE 7

SÉANCE 8

SÉANCE 9

SÉANCE 10

SÉANCE 11

SÉANCE 12

SÉANCE 13

QUIZ

Quiz 02 | Quiz 03 | Quiz 04 | Quiz 05 | Quiz 06 | Quiz 08 | Quiz 09 | Quiz 10 | Quiz 11 | Quiz 12

LABORATOIRES

- Planification des laboratoires
- Git lab0 introduction
- Git labo1 SGA

MODULE INTRODUCTION LOG210

ANALYSE ET CONCEPTION DE LOGICIELS

1. Présentation de l'équipe
2. Présentation personnelle 7.48m
3. Présentations des étudiants
4. Faire un logiciel pour un domaine que vous ne connaissez pas

Il faut trouver un moyen pour faire un lien entre ces titres de section et la section correspondante dans les notes de cours

ÉQUIPE

- Responsable du cour
  - Christopher Fuhrman
  - christopher.fuhrman@etsmtl.ca

- Chargé de cours
  - Yvan Ross, M.Ing
  - cc-yvan.ross@etsmtl.ca

Utiliser Discord pour communiquer avec moi

COMMUNIQUER AVEC MOI

- Par discord: **objet précis**, ex. `GR01: question sur le modèle du domaine`
- Rencontres organisées de façon proactive **en dehors** de ces heures. Proposez moi au moins deux périodes de disponibilité.
  - Lun 8h30-12:00
  - Lun 13:30 - 17:00
  - Mar 18:00 - 21:30
  - Jeu 08:30 - 12:00
  - Ven 13:30 - 16:30

CHARGÉS DE LABORATOIRE

- 01 Jeu 08:30 - 11:30 Labo A-3322
  - hind.errahmouni.1@ens.etsmtl.ca
- 02 Lun 13:30 - 16:30 Labo A-3322
  - edouard.laforge.1@ens.etsmtl.ca
- 03 Mer 18:00 - 21:00 Labo A-3322
  - islem.saidani.1@ens.etsmtl.ca

Utiliser Discord pour communiquer avec eux à propos du laboratoire.

PRÉSENTATION PERSONNELLE

CHRISTOPHER FUHRMAN

PRÉSENTATION PERSONNELLE

- Professeur depuis 2001 (Conception de logiciels)
- Expérience industrielle
  - 2019 : Développement Pharo.org Inria Lille
  - 2014-20 : PlantUML Gizmo
  - 2008 : Développement convertisseur XML FBI-WVU
  - 2000 : Directeur du programme NASA « ASSET »
  - 1998-1999 : Développeur chez ProLogic
  - 1989-1991 : Développeur chez Apple

PRÉSENTATION PERSONNELLE

- Diplômes universitaires
  - 1988 Bachelor of Science (informatique) WVU
  - 1996 Docteur ès sciences (informatique technique) École Polytechnique Fédérale de Lausanne

YVAN ROSS

PRÉSENTATION PERSONNELLE

- Expérience industrielle
  - 2013-: Chargé de cours ETS
  - 2010-2015: Consultant
  - 2011-2013: Web analytique cloud (ROR), gestion de projet
  - 2009-2011: Reconnaissance vocale. C++, java, ROR, Perl

PRÉSENTATION PERSONNELLE (...)

- 2008-2009: Capteur d'analyse réseau électrique. C++, simulink , Matlab

PRÉSENTATION PERSONNELLE (...)

- 2001-2008: R&D en imagerie médicale, C++, Matlab

PRÉSENTATION PERSONNELLE (...)

- 1998-2001: Développeur imagerie par microscopie, C++

PRÉSENTATION PERSONNELLE (...)

- 1994-1998: Directeur R&D, conception de système d'inspection automatisée par vision, C++, conception d'équipement de convoyage

PRÉSENTATION PERSONNELLE (...)

DIPLÔMES UNIVERSITAIRES

- 1994 Maîtrise en technologie des systèmes

DIPLÔMES UNIVERSITAIRES

- 1990 Baccalauréat en Génie de la production automatisée

PRÉSENTATIONS DES ÉTUDIANTS

Questionnaire à tiny.cc/quizdesign

Nom de la salle de class: **ETSDESIGN**

PRÉSENTATIONS DES ÉTUDIANTS

- Comment développer un logiciel?
  - Quelle(s) méthode(s) de développement avez-vous suivie(s)?
  - Pour quel genre de logiciel?
- Quelles sont vos attentes pour LOG210?

PRÉSENTATIONS DES ÉTUDIANTS

- Veuillez vous présenter sur Discord dans General.
  - Votre nom, votre programme d'étude collégiale, votre programme d'étude universitaire, vos stages (entreprise, domaine, projet), pourquoi vous êtes en génie logiciel.

MEILLEURES PRATIQUES POUR APPRENDRE

Quelles sont les meilleures pratiques pour apprendre?

MES CONSEILS POUR RÉUSSIR

- L'étudiant qui réussit dans mes cours...
  - ~~s'assied vers l'avant de la salle;~~
  - est motivé pour apprendre et pour participer,
  - est bien discipliné,
  - est organisé,
  - cherche de l'aide d'une façon proactive,
  - sais que le **multitâche** nuit à la qualité.

PROACTIF

1. **Domaine:** psychologie
2. **Auteur**: Office québécois de la langue française, 2002
3. **Définition:** Qui exerce un effet sur des faits ou des processus **à venir**

Exemple: Pompiers qui visitent les maisons pour vérifier les détecteurs de fumée

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?

Revers de la médaille: certains reprochent à leur métier la nécessité de se mettre régulièrement au niveau. Ils trouvent difficile de suivre les évolutions avec les langages et les nouvelles pratiques qui ne cessent d'apparaître ou de se modifier 😫🤯😫

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?

https://www.linkedin.com/pulse/2019-logiciel-le-meilleur-metier-au-monde-florian-verdet/?trackingId=Lx%2BFqqqTQ4eCuCzmVJ4Ow%3D%3D

INGÉNIEUR LOGICIEL, LE MEILLEUR MÉTIER AU MONDE ?

Autre inconvénient: la solitude, notamment pour les développeurs qui préfèrent travailler depuis chez eux. Un choix qui leur offre un emploi du temps flexible, mais qui les mène à passer de longues heures sans interagir avec le moindre collègue.

https://www.linkedin.com/pulse/2019-logiciel-le-meilleur-metier-au-monde-florian-verdet/?trackingId=Lx%2BFqqqTQ4eCuCzmVJ4Ow%3D%3D

FAIRE UN LOGICIEL POUR UN DOMAINE QUE VOUS NE CONNAISSEZ PAS

Exemple:
Application Web permettant de planifier les formations (faites par des spécialistes en TI) des enseignants dans les écoles québécoises.

Comment procéder?

- Appliquer un processus pour la réalisation
  - Appliquer une méthodologie d'analyse
  - Appliquer une méthodologie de conception

MODULE OUTILS

1. Notes de cours
2. Outils de travail
3. Visual studio code
4. vscode: Markdown all in one
5. vscode: Plantuml
6. vscode: livrShare
7. plantuml website
8. Mastering markdown
9. Git: Fork and pull request
10. Outils UML - 14.44m

NOTES DE COURS

Il existe des notes de cours depuis janvier 2020.

- https://drive.google.com/file/d/1__uJnJ7v-aBnf3DSuLGJ6KOvGb7uMVOo4
- Les notes de cours ne remplacent pas le livre obligatoire

OUTILS DE TRAVAIL

- Plan de cours
- Moodle - classe inversé
- Examen - enaquiz
- Google Drive pour le contenu
- Socrative
- Zoom / obs / youtube / discord
- one note
- laboratoires

OUTILS DE TRAVAIL...

- Visual studio code
  - extension
    - https://visualstudio.microsoft.com/services/share/
    - plantuml
    - Markdown all in One
    - Markdown PDF
- https://www.npmjs.com/package/gitinspector
- https://www.npmjs.com/package/tplant
- https://plantuml.com/
- https://wakatime.com/dashboard

LIVRE OBLIGATOIRE

- Version anglaise ou française?

DÉFINITIONS

- Pro-ingénierie
  - Générer du code source à partir des diagrammes (UML)

DÉFINITIONS

- Rétro-ingénierie
  - Générer des diagrammes (UML) à partir du code

DÉFINITIONS

- Ingénierie cycle
  - Fermer cette boucle

QUE DOIT-ON RECHERCHER DANS UN OUTIL?

- Essayez un outil gratuitement avant de l'acheter.
- Testez l'outil sur un projet réel.
- Choisissez un outil qui s'intègre à votre IDE préféré (VS, Eclipse, etc.)
- Rétro-ingénierie est important
- L'impression sur grande feuilles est importante (Larman)

LISTE D'OUTILS UML

1. Wikipedia
   - Suggestions:
     - Papyrus
     - Modelisoft
     - PlantUML

UML EN MODE ESQUISSE

- Esquisses murales UML au début d'une itération
- Suivies par 3 semaines de codage et de tests
  - Les modèles évolueront avec le code
- Entamer de nouveau la modélisation de nouvelles esquisses
- Générer les diagrammes à partir du code existant le jours précédant cette modélisation

EXEMPLE D'UNE ITÉRATION

MODULE ANALYSE ET CONCEPTION

ORIENTÉ OJET

1. Try to fix bug in production
2. Spectre de la conception
3. Analyse et conception orienté objet 21.26m
4. AOO – Approche orienté objet 7.26m
5. Première étude de la conception

TRY TO FIX BUG IN PRODUCTION

SPECTRE DE LA CONCEPTION

Figure 2.1 des notes de cours.

SURVOL DES MODÈLES

Modèle de données transactionnelles
Modèle de données analytiques
Modèle de données de référence
Modèle de données de référentiel
Modèle de données de référentiel
Modèle de données de référentiel

- Utilisées pour faire le design

1. LOG121 (Cay Horstmann, chap. 2.12):

Mail box
message passed **MessageQueue**
message grouping
message retry and actual messages

EXEMPLE CRC

BoyerdeCouperpetCouteur			
Responsabilités	**Collaborateurs**	**Invoices**	
Accepte plusieurs bons Calcule le prix Applique des rabais (voir Rabais) Sauvegarde des bons (persistance) Calcule des taxes	ExtraitDeCouperpetCouteur ExtraitDeRabais SalesDeMagasinDeDetail	**Responsabilités**	**Collaborateurs**
		Trouve de nouveaux clients à ranger Calcule	Revenue

Figure 9.12 Exemples d'CRC cards (de http://www.bigrigsoftware.com/) et des Unicliers classes.

ref: Object-Oriented Software Engineering, Bruegge & Dutoit, Prentice-Hall, 2000.

RÉSUMÉ

- Modélisation agile
  - Caméra pour numériser les esquisses
  - Habileté avec UML
- Modélisation dynamique est importante
- Pas beaucoup de temps pour modéliser
  - Une itération de 3 semaines (120 h)
  - 2 à 8 heures de modélisation, pas plus

MODULE TRAVAIL D'ÉQUIPE

1. Travail d'équipe (HRC) - 25.53m
2. Confiance - 4.33m
3. Humilité - 10.05m
4. Culture d'équipe - 20.146m
5. Rencontres efficace - 7.17m
6. Équipe en laboratoire

Séances | Quiz | Laboratoires | Intro | Équipe | PV | DOCUS | SVN | HRD | Dev | Central | RDD | UXD, | Cloche | Exercice | Tuyaux qui | Complexité | FURPS | TDD | DDP | Diagrammes | Divers

DÉVELOPPEMENT DE LOGICIEL EN ÉQUIPE

DÉVELOPPEMENT DE LOGICIEL EN ÉQUIPE

Pratiquement tout conflit social est dû à un manque d'humilité, de respect ou de confiance.

référence: Ocean Geeks-Brian W. Fitzpatrick (Google), Ben Collins-Sussman (Subversion, Google)

HUMILITÉ

Trouvez-moi un exemple ou:

- un étudiant ou un professeur fait preuve d'humilité

HUMILITÉ

Trouvez-moi un exemple ou:

- un étudiant ou un professeur fait preuve d'humilité
- un étudiant ou un professeur ne fait pas preuve d'humilité

RESPECT

Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier

RESPECT

Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe

RESPECT

Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe
- d'un étudiant envers un professeur ou un chargé de laboratoire

RESPECT

Trouvez-moi un exemple d'un manque de respect:

- envers un coéquipier
- envers les élèves de la classe
- d'un professeur envers un étudiant
- d'un professeur/chargé de laboratoire envers un étudiant

CONFIANCE

Trouvez-moi un exemple d'une:

- action qui donne confiance à mes coéquipiers

CONFIANCE

Trouvez-moi un exemple d'une:

- action qui donne confiance à mes coéquipiers
- action qui diminue la confiance envers mes coéquipiers

POURQUOI L'ENSEIGNANT DÉCIDE LES MEMBRES DE L'ÉQUIPE? (1/2)

- Pour apprendre les compétences non-techniques
  - Demandées par les employeurs
  - N'évoluant pas autant que les compétences techniques (p.ex. Flash)
- On ne choisit pas (facilement) son équipe en entreprise
- Travailler avec des collègues différents tout au long de son baccalauréat élargit son réseau de contacts
- Impact positif pour la carrière

Mark Granovetter, «The Strength of Weak Ties», Academic Press, 1977.

POURQUOI L'ENSEIGNANT DÉCIDE LES MEMBRES DE L'ÉQUIPE? (2/2)

- Favorise le « coaching » car il y a de la diversité
- Un coéquipier moins fort dans un aspect du projet peut bénéficier du coaching d'un membre plus fort
- Le « coach » peut bénéficier également de ce qu'on appelle «deep learning»
- L'équipe est finalement plus cohésive et plus performante

ÉVITER LES COMPORTEMENTS DYSFONCTIONNELS

- Chacun fait "sa part" sans se soucier de la dynamique de l'équipe et de la qualité globale du projet
  - L'équipe = moi (appliquer l'humilité)
  - Rester flexible pour les tâches, s'adapter
  - Proactivité envers problèmes dans l'équipe:
    - Exiger la présence des coéquipiers
    - Demander de l'aide (technologies)
    - S'impliquer!

ÉVITER LES COMPORTEMENTS DYSFONCTIONNELS

- Un ou deux membres prennent tout le contrôle (manque de confiance)
  - Reconnaître les niveaux différents et aider (faire du coaching)
  - Être patient avec les membres ayant moins d'expérience
  - Membres ayant besoin d'aide doivent la chercher tôt (être proactif, s'impliquer)
  - Facteur de bus > 1

CONFIANCE

Séances | Quiz | Laboratoires | Intro | Équipe | PV | DOCUS | SVN | HRD | Dev | Central | RDD | UXD, | Cloche | Exercice | Tuyaux qui | Complexité | FURPS | TDD | DDP | Diagrammes | Divers

CONFIANCE

- Je crois que les autres coéquipiers sont compétents et qu'ils feront la bonne chose.
- Je suis à l'aise lorsqu'ils prennent le volant, le cas échéant.

TENDANCES COMPORTEMENTALES CHEZ LES DÉVELOPPEURS...

- Je ne veux pas montrer mon code source à mes coéquipiers...
  - Mon code n'est pas fini...
  - J'ai trop peur d'être jugé(e)...
  - J'ai peur que quelqu'un voie mon idée... Il s'agit de l'insécurité. C'est normal!

CACHER SES IDÉES AUGMENTE LE RISQUE...

- de faire des erreurs dans la conception initiale
- de « réinventer la roue ».
- de terminer le travail plus tard que son compétiteur, qui, lui, a collaboré avec son équipe.

HUMILITÉ

HUMILITÉ

- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible

HUMILITÉ

- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible
- Je suis ouvert à m'améliorer

HUMILITÉ

- Je ne suis pas le centre de l'univers.
- Je ne suis ni omniscient ni infaillible
- Je suis ouvert à m'améliorer

Moi < Équipe

HUMILITÉ EN PRATIQUE...

- Si quelqu'un travaille sans assez d'humilité
  - «Yeux-tu descendre de ton piédestal?»
- Même si tu es le plus fort en JavaScript dans l'équipe, ne le mets pas en évidence constamment. C'est agaçant.
- Dans certaines cultures, l'humilité est très importante (p. ex. le confucianisme).

HUMILITÉ EN PRATIQUE...

- Apprendre à donner et à accepter les critiques **CONSTRUCTIVES**
- Les donner avec respect, p.ex. « J'ai de la misère à comprendre le flot de contrôle ici dans ton code » plutôt que « Ton code est mal écrit. »
- Les accepter avec humilité, p.ex. « Je comprends ton point de vue...Je vais refactoriser ce code en fin de semaine. »
- Savoir que son estime de soi n'équivaut pas à sa qualité de code.

HUMILITÉ EN PRATIQUE...

- Apprendre à être patient
- Chacun a sa personnalité et donc sa façon de déboguer, de concevoir, d'écrire du code
- Rester susceptible à l'influence des autres...
- Plus on est ouvert à être influencé, plus on peut influencer
- Plus on est vulnérable, plus on a l'air fort
- Si on veut être entendu, on doit d'abord écouter

CULTURE D'ÉQUIPE

CULTURE D'ÉQUIPE

- Qu'est-ce qu'une culture d'équipe?
  - Ensemble de valeurs, d'objectifs, d'expériences qui est unique à chaque équipe.
  - Éléments « codage » : revues de code, développement piloté par les tests, documentation de la conception, etc.
  - Éléments sociaux: sushi à midi, ou un 5 à 7 le vendredi, etc.
- Qu'elle soit bonne ou mauvaise, la culture existera
- Le leader ne décide pas la culture; il en occupe.

CULTURE FORTE D'ÉQUIPE

- Culture ouverte au changement qui l'améliore, mais résistant à un changement radical qui lui fait mal.
- Celle qui concentre l'effort sur **la livraison de logiciel génial** est la meuve réussite.
- Efforts pour souder une équipe ne mènent pas toujours à la productivité:
  - faire la fête, surenchère de programmation, faire des rencontres

CULTURE FORTE D'ÉQUIPE

- Une culture forte sera une culture d'auto-sélection
- Culture de code propre et facile à maintenir attirera des développeurs appréciant ces valeurs...
- Culture d'agressivité, d'initiation, de dérapages verbaux, etc. attirera des développeurs appréciant ces valeurs...

ÉLÉMENTS POUR LES CULTURES D'ÉQUIPE RÉUSSIES

- Énoncé de mission d'équipe
- Culture de HRC
- favorise une participation des coéquipiers extroverts et introverts
- team.ego > coéquipier[].ego
- La critique constructive
  - facile à recevoir
  - difficile à donner : ne pas oublier « respect »
- Bonne communication est essentielle
- Moyens de communication diversifiés
- Listes de diffusion
- Documentation de design
- Communication synchrone (rencontres) vs asynchrone (courriel)

COMPORTEMENTS MENAÇANT UNE BONNE CULTURE

- Ne pas respecter le temps des autres
- Ne pas respecter une décision prise par l'équipe
- Ne pas écouter ou respecter les autres
- Ne pas faire de compromis
- Être perfectionniste
- Être provocateur (troll) / Répondre aux provocateurs (trolls)
- Devenir trop affectif

L'attention et la concentration sont primordiales.

COMMENT AGIR FACE À CES COMPORTEMENTS

- Chercher des faits dans le drame
- Si quelqu'un se plaint, même avec trop d'émotions, doutez le bénéfice du doute et chercher les causes (malgré le manque de respect, etc.)
- Amener la discussion sur un plan technique si possible.
- Souvent il y a des choses à améliorer dans la situation.
- La gentillesse peut chasser les trolls en fin de compte...

COMMENT AGIR FACE À CES COMPORTEMENTS

- Se concentrer sur l'objectif à long terme
- Un témoin de comportement délétère se demande:
  - Malgré la perte de concentration de l'équipe à court terme, une résolution du drame sera-t-elle bénéfique à l'équipe à long terme?
  - Est-ce que la situation se résoudra d'elle-même?
- Si la réponse est « non » à une de ces questions, mettre fin au comportement immédiatement (sans résolution).

COMMENT AGIR FACE À CES COMPORTEMENTS

- Savoir quand abandonner
- Parfois le comportement d'un coéquipier ne s'améliore pas malgré beaucoup d'efforts. Il faut dans ce cas l'isoler de l'équipe.

Avant de demander un changement d'équipe, il faudra avoir essayé une approche HRC et être en mesure de l'expliquer à l'enseigant.

CULTURE D'ÉQUIPE

- La plupart des gens ne sont pas des imbéciles.
- Cependant, il est nuif de penser aux gens comme « **bons**» ou « **mauvais** ».
- La malice qui menace une bonne culture d'équipe est souvent expliquée par l'**ignorance, le besoin d'être reconnu**, ou **un manque d'empathie**…
- Il faut toujours être tolérant envers les gens, mais *ne pas tolérer les comportements* qui nuisent à une bonne culture d'équipe (selon la norme HRC).

RENCONTRES EFFICACES

Séances | Quiz | Laboratoires | Intro | Équipe | PV | DOD | CCU | HSD | Dis | Control | RDC | ODO | Cache | Exercice | Tapescript | Complexité | FURPS | TDD | GOF | Diagrammes | Divers

RENCONTRES EFFICACES

1. N'inviter que ceux qui doivent absolument **être** présents.
2. Préparer et distribuer l'ordre du jour bien avant que ça commence.
3. Terminer tôt si les objectifs ont été atteints.
4. Respecter l'ordre du jour.
5. Planifier autour des pauses habituelles (p.ex., à midi, à la fin de la journée).

[TeamGeek]

ÉQUIPE EN LABORATOIRE

Séances | Quiz | Laboratoires | Intro | Équipe | PV | DOD | CCU | HSD | Dis | Control | RDC | ODO | Cache | Exercice | Tapescript | Complexité | FURPS | TDD | GOF | Diagrammes | Divers

TRAVAIL D'ÉQUIPE AU NIVEAU DU LABORATOIRE

- Quels sont les problèmes HRC rencontré par votre équipe?
- Quels solutions avez vous apportées?

MODULE PROCESSUS UNIFIÉ

1. Phases du processus unifié + inception - 5.21m
2. Durée typique - 15.20m
3. Développement itératif et incrémental - 13.23m

Séances | Quiz | Laboratoires | Intro | Équipe | PV | DOD | CCU | HSD | Dis | Control | RDC | ODO | Cache | Exercice | Tapescript | Complexité | FURPS | TDD | GOF | Diagrammes | Divers

PHASES DU PROCESSUS UNIFIÉ

PHASE INCEPTION

- Étude de faisabilité
  - Vision approximative
  - Estimations globales
  - Cas d'utilisation
  - Continuer ou non (p.ex. chercher le financement)

- Ne pas confondre avec
  - analyse des exigences

PROCESSUS UNIFIÉ?

- Quelle est une durée typique d'une itération dans le processus unifié?

APPROCHE ÉVOLUTIVE VS APPROCHE EN CASCADE

PROCESSUS UNIFIÉ

- Souple et ouvert – peut accommoder l'Extreme Programming (XP), Scrum, etc.
- Développement piloté par les tests
  - intégration continue « war room »
  - réunions quotidiennes
  - etc.
- Processus *itératif* et *évolutif*

ITÉRATIF ET ÉVOLUTIF

RÉSUMÉ

- Avantages du processus unifié
  - courtes itérations produisant un système partiel
  - rétroaction du client permet d'ajuster la prochaine itération
  - meilleure gestion du risque
- Inconvénients
  - Clients veulent les « promesses » de coûts au début
  - Contrat traditionnel (!) n'est pas adapté pour un processus itératif (orienté modèle en cascade)

PROCESSUS UNIFIÉ

- Souple et ouvert – peut accommoder l'Extreme Programming (XP), Scrum, etc.
- Développement piloté par les tests
- Processus *itératif* et *évolutif*

ITÉRATIF ET ÉVOLUTIF

STABILISATION

Les premières itérations sont fort de la nature du système, mais les dernières itérations du système converge vers les spécifications et le concepton les plus appropriées

Les des itérations ultérieures, un changement significatif des spécifications se sera mis jour en cours de route. Le système converge vers une version stable et avantage concurrentiel.

une libration : conception,
implémentation, intégration et tests

ITÉRATIF ET ÉVOLUTIF

GESTION DES EXIGENCES (MIMINUM)

Ecrire des spécifications correctes et complètes n'est pas facile

MODULE DIAGRAMME DE CAS D'UTILISATION

1. DCU - Diagramme de cas d'utilisation - 3.24m

Séances | Quiz | Laboratoires | Intro | Equipe | PV | DCU | CU | MDD | Dis | Control | RDCU | OCL | Cache | Exercice | Tuyaux up | Complexité | FURPS | TDD | GDP | Diagrammes | Divers

DCU (LOG410)

DIAGRAMME DE CAS D'UTILISATION

https://plantuml.com/use-case-diagram

CONCEPTS DE BASE DU DIAGRAMME DE CAS D'UTILISATION

1. **Sujet (subject)**
   - Le sujet est utilisé pour délimiter le système de son environnement
2. **Acteur**
   - Entité externe qui interagit avec le système
   - Peut représenter des personnes / utilisateurs et d'autres systèmes externes avec lesquels le système interagit

CONCEPTS DE BASE DU DIAGRAMME DE CAS D'UTILISATION...

1. **Cas d'utilisation**
   - Fonctionnalité fournie par le système
   - Décrit comme une séquence d'étapes / actions
2. **Relations**
   - L'interaction entre les acteurs et un cas d'utilisation est spécifiée à l'aide d'une association.
   - Différents types de relations entre les cas d'utilisation: include, extend, generalization/inheritance

TROIS TYPES D'ACTEURS

1. **Acteur principal**
   - Il a un objectif et utilise le système à l'étude (SAE)
2. **Acteur de support**
   - fournit un service, une information, au SAE
   - p.ex. système d'autorisation de paiement. Peut être humain, système externe ou équipement.
3. **Intervenant**
   - a un intérêt, mais ni acteur principal ou acteur de support (ex: service des taxes du gouvernement, comptable, vérificateur, propriétaire, client, ...)

DIAGRAMME DE CAS D'UTILISATION (AGILE)

MODULE CAS D'UTILISATION

1. CU - Cas d'utilisation informel - 1.53m
2. CU - Cas d'utilisation - types

Séances | Quiz | Laboratoires | Intro | Equipe | PV | DCU | CU | MDD | Dis | Control | RDCU | OCL | Cache | Exercice | Tuyaux up | Complexité | FURPS | TDD | GDP | Diagrammes | Divers

CAS D'UTILISATION - INFORMEL

1. **Traiter une vente**. Un client arrive à la caisse les articles qu'il souhaite acheter. Pour enregistrer chaque article, le caissier utilise le système POS, lequel présente le détail des articles et le montant total des achats. Le client fournit les informations nécessaires pour le règlement. Le système valide et enregistre ces informations, puis met à jour les quantités en stock et imprime le ticket de caisse destiné au client. La vente est terminée et le client peut quitter le magasin.

DE LA DESCRIPTION DU CAS D'UTILISATION AU DIAGRAMME DE CAS D'UTILISATION

CAS D'UTILISATION

- Que fait le système et non comment
- Trois types:
  - Abrégé
    - Un paragraphe décrivant le scénario principal
  - Informel (casual)
    - Plusieurs paragraphes décrivant plusieurs scénarios
  - Détaillé
    - Toutes les étapes et les variantes sont indiquées en détail, de même que les préconditions et les garanties en cas de succès

DÉFINITIONS

- Acteur
  - Entité qui a un comportement, comme une **personne** (identifiée par un rôle)
- Scénario
  - Suite spécifique d'actions et d'interactions entre un ou plusieurs acteurs et le système. C'est une histoire particulière de la façon dont on utilise un système
- Cas d'utilisation
  - Collection de scénarios de réussite ou d'échec.

ACTEUR

- Quelque chose ayant un comportement
  - un usager, identifié par un rôle
  - un système informatique
  - une organisation
  - par exemple : un caissier

MODULE D'INTERFACE USAGÉ

1. IU - Interface Usager vs DSS 6.22m

Séances | Quiz | Laboratoires | Intro | Equipe | PV | DCU | CU | MDD | Dis | Control | RDCU | OCL | Cache | Exercice | Tuyaux up | Complexité | FURPS | TDD | GDP | Diagrammes | Divers

INTERFACE USAGER

- Réaliser les wireframes des différentes interfaces usagées nécessaires pour la réalisation du cas d'utilisation Noter une réservation
- Utiliser plant uml
  - https://plantuml.com/salt

MODULE MDD

1. Modèle du domaine - 12.25m
2. Catégories - 30.27m
3. Classes, Attributs, Associations - 58.47m
4. Composition - 22.47
5. Révision - 6.05m
6. Décalage de représentataion - 24.12m
7. Affinement du MDD - 25.42m

Séances | Quiz | Laboratoires | Intro | Equipe | PV | DCU | CU | MDD | Dis | Control | RDCU | OCL | Cache | Exercice | Tuyaux up | Complexité | FURPS | TDD | GDP | Diagrammes | Divers

QU'EST-CE QU'UN MODÈLE DU DOMAINE?

Représentation visuelle des classes conceptuelles des objets du monde réel dans un domaine donné [MO95, Fowler96]

- modèle conceptuel
- modèle objet du domaine
- modèle objet d'analyse

EXEMPLE AVEC L'OUTIL PLANTUML

Plant Uml class diagram

COMMENT CRÉER UN MODÈLE DU DOMAINE?

- Dans les limites des besoins de l'itération en cours:
  1. Identifier les classes conceptuelles.
  2. Les représenter sous forme de classes dans un diagramme UML.
  3. Ajouter des attributs et des associations.

RÉALISER VOTRE PREMIER MDD

En équipe de 2

1. Trouver les classes du domaine
2. Ajouter des associations entre les classes lorsque vous désirez persister cette relation
3. Ajouter un verbe pour décrire cette association
4. Ajouter une multiplicité à l'association

RÉALISER VOTRE PREMIER MDD

Cas d'utilisation «Noter une réservation»

Scénario principal

1. Un client appelle à l'hôtel pour placer une réservation.
2. Le commis démarre une nouvelle réservation.
3. Le commis entre:
   - La date d'arrivée;
   - La date de départ;
   - Le nom de la catégorie de chambre;
4. Le système inscrit les informations à la réservation.

Interface usager | Réaliser votre premier DSS | Itération suivante

SURVOL VISUEL

Système
Système_Réservation
Système_Gestion_Ressources
Système_Gestion_Clients
Système_Gestion_Employés
Système_Gestion_Finances
Système_Gestion_Fournisseurs
Système_Gestion_Ressources_Humaines
Système_Gestion_Marketing
Système_Gestion_Ventes
Système_Gestion_Stocks
Système_Gestion_Transport
Système_Gestion_Sécurité
Système_Gestion_Maintenance
Système_Gestion_Qualité
Système_Gestion_Environnement
Système_Gestion_Sécurité_Informatique
Système_Gestion_Formation
Système_Gestion_Recrutement
Système_Gestion_Paie
Système_Gestion_Comptabilité
Système_Gestion_Achats

TYPES DE DONNÉES CH. 9.16

- Définir de nouveaux types quand les types primitifs ne sont pas adéquats
  - ex : Sigle de cours → LOG120

  - ex : un code SKU

UML : CLASSES TYPES DE DONNÉES

*GÉNÉRALISATION

Lire Ch. 32.2-32.7 (version française 26.2-26.7)

SOUS-CLASSES CONCEPTUELLES

Les énoncés sur les super-classes s'appliquent aux sous-classes

RÈGLE « À 100% »

- 100% de la définition d'une classe conceptuelle est applicable aux sous-classes
- Sous-classe
  - conforme à 100% à sa superclasse
    - en termes d'attributs
    - en termes d'associations

RÈGLE « EST UN »

- Sous-classe conceptuelle
  - est membre de l'ensemble de sa superclasse
  - objet d'une sous-classe **est un** objet de la super-classe

DÉFINIR UNE SOUS-CLASSE CONCEPTUELLE QUAND ELLE :

- possède des attributs additionnels intéressants
- possède des associations additionnelles intéressantes
- réagit, est manipulée, est traitée d'une façon intéressante et
  - différemment de sa superclasse
  - différemment d'autres sous-classes
- représente un objet animé (humain, animal, robot) ayant un comportement différent et intéressant

SUPER-CLASSE CONCEPTUELLE

- Quand définit-on une super-classe conceptuelle?
  - Lorsque les classes sont des variations de concepts similaires
  - respectent les règles à 100% et « est un »
  - ont des attributs communs
    - qui pourraient être placés dans la super-classe
  - ont les mêmes associations
    - qui pourraient s'appliquer à la super-classe

EXEMPLE DE PARTITION INUTILE

figure: F26.6, A32.6

CLASSES D'ASSOCIATION

- Services d'autorisation de paiement
  - assignent une identification à chaque magasin
  - demande d'autorisation de paiement ? codeVendeur
  - le même magasin peut utiliser plusieurs services, chacun lui donnant un codeVendeur différent
- Dans quelle classe conceptuelle mettre codeVendeur?

SOLUTION INTERMÉDIAIRE

CLASSE D'ASSOCIATION

AJOUT - CLASSE D'ASSOCIATION

- Règles pour l'ajout :
  - un **attribut** est lié à une association
  - la durée de vie des instances de la classe d'association dépend de l'association
    - «historic mapping» (patron d'analyse, M. Fowler)
- il y a une association N-N entre deux concepts et des **informations liées à l'association** elle-même.

https://martinfowler.com/apsupp/properties.pdf

CLASSES D'ASSOCIATION (EXEMPLE)

*SOLUTION EXERCICE MDD SUR GOOGLE CLASSROOM

voir: seance-03-exercice-reserverLivre

EXERCICE MDD (HÔTEL)

1. Un client appelle à l'hôtel pour placer une réservation.
2. Le commis démarre une nouvelle réservation.
3. Le commis saisit la date d'arrivée, la date de départ; le nom de la catégorie de chambre; la quantité de chambres.
4. Le système inscrit les informations à la réservation.

Les étapes 3 et 4 sont répétées tant que le client n'indique pas qu'il a terminé

5. Le commis termine la réservation.
6. Le système affiche toutes les informations entrées.
7. Le commis saisit les informations auprès du client et confirme la réservation à l'aide du nom et du numéro de téléphone du client.
8. Le système enregistre la réservation et affiche le numéro de confirmation.
9. Le commis communique le numéro de confirmation au client.

voir seance-01-exercice-CU02

MDD

Correction de l'exercice

MDD?

- Attributs dérivées?

MDD?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?
  - Règle « à 100% »?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?
  - Règle « à 100% »?
  - Règle « est un »?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?
  - Règle « à 100% »?
  - Règle « est un »?
  - Création d'une sous-classe conceptuelle?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?
  - Règle « à 100% »?
  - Règle « est un »?
  - Création d'une sous-classe conceptuelle?
  - Création d'une super-classe conceptuelle?

MDD?

- Attributs dérivées?
- Généralisation/spécialisation?
  - Règle « à 100% »?
  - Règle « est un »?
  - Création d'une sous-classe conceptuelle?
  - Création d'une super-classe conceptuelle?
- Classe d'association

AGRÉGATION ET COMPOSITION

- N'utilisez pas l'agrégation (c'est un placebo)
- Utilisez la composition lorsque c'est approprié.

COMPOSITION?

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties
- **Propriétés** du composite (ex. son emplacement s'étendent aux composants)

COMPOSITION?

Durée de vie du composant est limitée à celle du composite;

- **dépendance de création-suppression** de la partie avec le tout.
- **Assemblage logique ou physique évident** entre le tout et les parties
- **Propriétés** du composite (ex. son emplacement s'étendent aux composants)
- **Opérations** appliquées au composite (destruction, déplacement et enregistrement) se propagent aux composants

COMPOSITION

AVANTAGES

- Clarifie des contraintes du domaine
- Aide à identifier le **patron créateur**
- Opérations sont souvent propagées aux composants

INCONVÉNIENT

- «Chicanerie» - laisser tomber la composition en cas de doute ou de conflit dans l'équipe (voir Bikeshedding)

MODÉLISATION DES ASSOCIATIONS

- La création d'une instance ne signifie pas la création d'une association
- Association un à plusieurs (⚠ dans la solution)
  - Avec un attribut
  - Avec une map (tableau associatif)
  - Avec une liste

CLASSE DE « DESCRIPTION »

- Article représente un article physique
- Attributs prix, numéro de série et code
- Chaque fois qu'un article est vendu, l'objet Article correspondant est « supprimé »

CLASSE DE « DESCRIPTION »

- Il est nécessaire de disposer de la description d'un produit ou d'un service, indépendamment de l'existence actuelle de ces derniers
- La suppression d'instances d'une entité qu'elle décrit entraîne la perte d'information qui doit être mémorisée
- Elle réduit la redondance ou la duplication des informations

CLASSES DE DESCRIPTION

- Souvent, les objets d'une classe de description sont stockés dans un Catalogue
- Lire en détail la section 9.13 du livre pour d'autres exemples

CLASSES DE DESCRIPTION

- Attention : ne pas appliquer les classes de description n'importe comment:

MDD?

- Agrégation vs composition?
- Comment interpréter une multiplicité de plusieurs

SYMBOLE, INTENSION, EXTENSION

Individu
Mike
Mike
Fraise
Fraise
Fraise

CHP26 AFFINEMENT DU MDD

1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?

CHP26 AFFINEMENT DU MDD

1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?
4. Est-il préférable d'utiliser une agrégation ou une composition? Pourquoi?

CHP26 AFFINEMENT DU MDD

1. Qu'est-ce que la généralisation?
2. Quelle est la relation entre la sous classe et la super classe?
3. Qu'est-ce qu'une classe d'association?
4. Est-il préférable d'utiliser une agrégation ou une composition? Pourquoi?
5. Qu'est-ce qu'un attribut dérivé?

AFFINEMENT DU MODÈLE DU DOMAINE

Catégorie	Exemples
Objets physiques ou tangibles	CarteDeCrédit, Chèque
Transactions	PaiementEnEspèces, PaiementACrédit, PaiementParChèque
Systèmes externes	ServiceDAutorisationDesCrédits, ServiceDAutorisationDesCheques
Organisations	ServiceDAutorisationDesCrédit, ServiceDAutorisationDesCheques, Documents de travail comptables, contractuels, juridiques

Tableau F26.1/A32.1

EXTENSIONS AU UC1: TRAITER VENTE

- 7b. Payer par carte de crédit
  - ... information de crédit ... demande d'autorisation ... autorisation de paiement ...
- 7c. Payer par chèque
  - ...
  - chèque ... pièce d'identité ...
  - autorisation de paiement par chèque ... paiement par chèque ...

SYSTÈMES D'AUTORISATION DE PAIEMENT

- RequêtePaiementACrédit et AccordCrédit
  - Abstractions des éléments dans l'activité d'autorisation de paiement
  - Pas nécessairement des informations (en termes de données transactionnelles) transmises sur le fil du réseau
  - **Classes conceptuelles toujours**

EXEMPLE DANS POS

Généralisation/spécialisation

EXEMPLE DANS POS

Hiérarchie des services d'autorisation

TRANSACTIONS D'AUTORISATION

- Est-ce utile?

Figure F26.1/A32.9

TRANSACTIONS D'AUTORISATION

Le modèle du domaine n'est pas jugé en fonction de sa précision (justesse, correcte ou non) mais en terme de son utilité. Il n'est pas un but en soi

MODÉLISATION DE CHANGEMENT D'ETAT

- Ne modélisez pas les différents états possibles d'un concept sous classes
- Deux solutions
  - définir une hiérarchie d'états et les associés à X ;
  - ignorer la représentation des états dans le MDD et faire un diagramme d'états

Payment
Payment
Payment

Système
Système
Système

Voir: Seance-03-solution-reserverLivrebibliotheque

POINTS IMPORTANTS

Voir le PDF des notes de cours

RETOUR D'UNE OPÉRATION SYSTÈME (DSS)

- Du cas d'utilisation...
  - Le système présente le total incluant les taxes calculées.
- L'affichage du total => couche présentation
- Mais le calcul du total?
  - Aucune classe n'en a la responsabilité
  - Affectez-la avec « Expert »
  - (il y a un attribut dérivé `/total` dans le MDD)

EXPERT POUR CALCULER LE TOTAL

1. Énoncer la responsabilité
   - Qui doit connaître le montant total de la vente?
2. Récapituler les informations nécessaires
   - total est la somme de tous les sous-totaux des lignes
   - sous-total de ligne :- quantité * prix
3. Dresser la liste des informations nécessaires et des classes qui les possèdent.

Informations nécessaires au total de la vente	Expert
Produit.prix	Produit
LigneArticles.quantité	LigneArticles
Toutes les LigneArticles de la vente en cours	Vente

Analyse détaillée: p.F337

CONCEPTION

`Vente.getTotal()`

(diagramme de communication)

CHP10 DSS

1. Quels sont les artéfacts qui influencent ou influencés par le DSS?

CHP10 DSS

1. Quels sont les artéfacts qui influencent ou influencés par le DSS?
2. Qu'est-ce qu'un DSS?

CHP10 DSS

1. Quels sont les artéfacts qui influencent ou influencés par le DSS?
2. Qu'est-ce qu'un DSS?
3. Pourquoi tracer un DSS?

INFLUENCE ENTRE ARTEFACTS DU PU

Artefacts Entrés et sortis du processus PU
Modèle du domaine
Modèle de conception
Modèle de données persistantes
Modèle de déploiement
Modèle de cas d'utilisation
Modèle de test
Modèle de processus
Modèle de gestion de projet
Modèle de configuration et de changement
Modèle d'environnement
Artefacts Entrés et sortis du processus PU
Modèle de domaine
Modèle de cas d'utilisation
Modèle de conception

Quel sont les trois type de postcondition?

Created by Benjamin Fy

CONTRATS

- Postcondition:
  - « ... sur la base de correspondance avec clé » :
    - implique une multiplicité de 1 à plusieurs
    - implique l'utilisation d'une Map (tableau associatif)

CONTRATS - MAPS

- sur la base de correspondance avec "Clé":
  - implique l'utilisation d'un tableau associatif (Map<>)

CHP11 CONTRATS

1. Quels sont les éléments d'un contrat?

Created by Benjamin Fy

CHP11 CONTRATS

1. Quels sont les éléments d'un contrat?
2. Quelle est la relation entre le DSS et les contrats?

CHP11 CONTRATS

1. Quels sont les éléments d'un contrat?
2. Quelle est la relation entre le DSS et les contrats?
3. Que décrivent les postconditions?

INFLUENCE D'ARTEFACTS DU PROCESSUS UMPIE

Modèle conceptuel
Modèle de conception
Modèle d'implémentation
Modèle de déploiement
Modèle de cas d'utilisation
Modèle de test
Modèle de processus métier
Modèle d'exigences
Modèle d'analyse
Modèle de données
Modèle de mise en oeuvre
Modèle de validation
Modèle de processus
Modèle de déploiement
Modèle de gestion des changements et des configurations
Modèle d'environnement
Modèle de gestion de projet
Modèle de mesures
Modèle de cas d'utilisation métier
Modèle d'architecture
Modèle d'interface
Modèle de composants
Modèle de base de données
Modèle de sécurité
Modèle de performance
Modèle de fiabilité
Modèle de test d'intégration
Modèle de test système
Modèle de test d'acceptation
Modèle de test d'unité
Modèle de test de régression
Modèle de test de performance
Modèle de test de sécurité
Modèle de test de fiabilité
Modèle de test de mise en oeuvre
Modèle de test de déploiement
Modèle de test de validation
Modèle de test de processus
Modèle de test de gestion des changements et des configurations
Modèle de test de gestion de projet
Modèle de test de mesures
Modèle de test de cas d'utilisation métier
Modèle de test d'architecture
Modèle de test d'interface
Modèle de test de composants
Modèle de test de base de données
Modèle de test de sécurité
Modèle de test de performance

RDCU (SOLUTION JAVA SWING)

Opération `démarrerJeu` - qui **envoie** cette opération ?

RDCU (SOLUTION JAVA SWING)

Opération `démarrerJeu` - qui **reçoit** cette opération ?

PRINCIPE CONTRÔLEUR GRASP

Problème: Quel est le premier objet en dehors de la couche présentation **qui reçoit** et coordonne (« contrôle ») les opérations système ?

PRINCIPE CONTRÔLEUR GRASP

Solution: Affectez une responsabilité à la classe qui correspond à l'une de ces définitions:

1. Elle représente le **système global**, un « **objet racine** », un **équipement** ou un **sous-système**.
2. ...

Modèle du domaine (adapté du livre de Bibi du Ch. 1 de Larman)

RDCU - CONTRÔLEUR

JeuDeDés est le contrôleur GRASP (inspiré du livre)

RDCU - DÉTAILLÉ

Faire un design modulaire et facile à comprendre.
Affecter les responsabilités aux bonnes classes.

RDCU

Prendre les bonnes décisions pour une solution facile à comprendre et modulaire...

DÉCALAGE DES REPRÉSENTATIONS

Facile? Les classes logicielles devraient ressembler des classes conceptuelles.

Qui fait quoi? Qui a quelle responsabilité?

RDCU

Approche: conception orientée-responsabilité

GRASP

General **R**esponsability **A**ssignment **S**oftware **P**atterns

Pour décider où mettre les méthodes...

GRASP

- Contrôleur (séparation des couches)
- **Créateur**
- **Expert en information**
- Faible couplage
- Forte cohésion
- Polymorphisme
- Indirection
- Protection de variation
- Fabrication pure

*RDCU (SURVOL)

RDCU: SCÉNARIO DÉMARRER

- C'est l'initialisation du système.
- C'est implicite mais essentiel!
- On doit instancier les objets faisant partie de l'application.

INSTANCIER LES OBJETS CASE

- Quelle classe s'en occupe? (sondage 2)

CRÉATEUR (GRASP)

- Les «conteneurs» créent les objets «contenus»

CRÉATEUR (GRASP)

- **Problème:** Qui crée? (postcondition d'un objet)
- **Solution:** Affecter à la classe B la responsabilité de créer les objets d'une classe A si...
  - B possède les données d'initialisation des objets A
  - B contient ou agrège des objets A
  - B utilise étroitement des objets A
  - B enregistre des objets A

On s'inspire du MDD. On réutilise les liens existants.

*CRÉATEUR

- Qui crée les cases (Square)?

*CRÉATEUR (ANNOTATION)

EXPERT EN INFORMATION

- **Problème:** Quel est le principe général d'affectation des responsabilités aux objets?
- **Solution:** Affecter la responsabilité à la classe qui possède les informations nécessaires pour s'en acquitter

En termes de paramètres, associations

EXPERT (GRASP)

- (Sondage) Où mettre la méthode `getCase(nom)`?

EXPERT

- Application du patron Expert

Exercice RDCU (Google Classroom)

Voir seance-03-exercice-reserveVue

Created by Mooncascade

GRASP

- Contrôleur
- Créateur
- Expert en information
- **Faible couplage**
- **Forte cohésion**
- Polymorphisme
- Indirection
- Protection de variation
- Fabrication pure

FAIBLE COUPLAGE

Question : Qui peut fournir la liste des cases ?

FAIBLE COUPLAGE

- Si la classe Pion (chien) avait la responsabilité

FAIBLE COUPLAGE

- Problème :
  - Comment réduire l'impact des modifications?
- Solution :
  - Assigner les responsabilités de sorte à **éviter tout couplage inutile**.
  - Appliquer ce principe pour **évaluer plusieurs solutions possibles**.

FAIBLE COUPLAGE

- Expert soutient Faible Couplage
- Expert évite de faire du couplage inutile, puisqu'un expert n'aura pas à demander à quelqu'un d'autre…

FAIBLE COUPLAGE

MOTIVATION

- Classes fortement couplées sont
  - difficiles à maintenir
  - difficiles à comprendre seules
  - difficiles à réutiliser

FORTE COHÉSION

- Poursuivons la conception

COHÉSION

- Mesure de combien une classe (ou module) supporte une seule responsabilité

COHÉSION

Faible	Forte
Beaucoup de fonctions	Peu de fonctions
Responsabilités écartées	Fonctions apparentées

FORTE COHÉSION

- Plusieurs choix opposés de conception

FORTE COHÉSION

- JeuDeMonopoly délègue les responsabilités

FORTE COHÉSION

1. **Problème** :
   - Comment s'assurer que les objets restent compréhensibles et faciles à gérer, et, bénéfice second, qu'ils contribuent au Faible Couplage?
2. **Solution** :

FORTE COHÉSION

1. **Problème** :
   - Comment s'assurer que les objets restent compréhensibles et faciles à gérer, et, bénéfice second, qu'ils contribuent au Faible Couplage?
2. **Solution** :
   - Assigner les responsabilités de sorte que la cohésion demeure élevée. Appliquer ce principe pour évaluer les solutions possibles.

FORTE COHÉSION

- Comparaison des choix

COHÉSION: EXERCICE

- Lesquelles des classes suivantes ont des bonnes responsabilités?

COHÉSION: EXERCICE

- Déterminez si ces classes ont une cohésion forte ou faible:

VALIDATION DE LA COMPRÉHENSION

Les patrons GRASP Forte Cohésion et Faible Couplage servent principalement à

1. concevoir des classes avec beaucoup de responsabilités
2. choisir les classes dans la couche applicative pour traiter une opération système
3. évaluer plusieurs possibilités lors de la réalisation d'une dynamique de conception
4. choisir une bonne classe pour créer des instances d'une autre classe

RDCU-GRASP

1. **Contrôleur**
2. **Créateur**
3. **Expert en information**
4. **Faible couplage**
5. **Forte cohésion**
   - Polymorphisme
   - Indirection
   - Protection de variation
   - Fabrication pure

VISIBILITÉ

- Avant qu'un objet puisse envoyer un message à un autre objet, il faut que l'objet
  - voit « ce dernier
- Avoir une visibilité = posséder une référence (en Java) P 18.1

TYPES DE VISIBILITÉ

- Attribut
- Paramètre
- Locale
- Globale

VISIBILITÉ D'ATTRIBUT

- Relativement permanente
- Très commune

VISIBILITÉ DE PARAMÈTRE

- Durée de vie relativement courte
- Parametre peut devenir attribut

VISIBILITÉ LOCALE

- Durée de vie relativement courte
- Peut être définie via
  - Instance créée localement
  - Instance retournée par une méthode

VISIBILITÉ GLOBALE

- Relativement permanente
- Moins commune
- Meilleure définition via un singleton (GoF)

```
class Singleton {
  private static Singleton singleton;
  private static int maxNumberOfObjects = 1;
  private static int numberOfObjects = 0;
  private Singleton() {
  }
  public static Singleton getSingleton() {
    if (numberOfObjects < maxNumberOfObjects) {
      singleton = new Singleton();
      numberOfObjects++;
    }
    return singleton;
  }
  public void finalize() {
    numberOfObjects--;
    super.finalize();
  }
}
```

Usage

```
Singleton s = Singleton.getSingleton();
Singleton t = Singleton.getSingleton();
System.out.println("s == t: " + (s == t));
s = null;
t = null;
System.gc();
System.out.println("Done.");
new Object();
new Object();
s = Singleton.getSingleton();
t = Singleton.getSingleton();
System.out.println("s == t: " + (s == t));
s = null;
t = null;
System.gc();
```

CRÉATIVITÉ ET CHANGEMENTS PENDANT LE CODAGE

- Les artefacts de conception affectent l'implémentation
- Pendant la programmation et les tests
  - multitudes de changements peuvent surgir
- S'attendre à des déviations et le prévoir
  - Favoriser le changement
- Le codage n'est pas une activité triviale
  - Pourquoi?

MÉTHODES À PARTIR DES DIAGRAMMES D'INTERACTION

SALESLINEITEM

- SalesLineItem en Java

RÉSUMÉ

- Nous avons vu des processus de traduction
  - les diagrammes de classes UML et les définitions de classes
  - les diagrammes d'interaction UML et les corps des méthodes
- Mais le travail de programmation laisse encore beaucoup de place à l'exploration

VALIDATION DE LA COMPRÉHENSION

- Quelle est la motivation principale du principe GRASP Créateur?
  1. Pour avoir une création rapide (performante) d'instances
  2. Pour obtenir une conception plus facile à comprendre
  3. Pour avoir une conception qui prend moins de mémoire
  4. Pour identifier des occasions d'appliquer le patron Fabrique

VALIDATION DE LA COMPRÉHENSION

- Le pattern Contrôleur GRASP sert à trouver une bonne classe…
  1. de la couche présentation pour réaliser une opération système
  2. de la couche domaine pour réaliser une opération système
  3. de l'IHM pour implémenter une logique applicative
  4. de l'IHM pour contrôler le système

VALIDATION DE LA COMPRÉHENSION

- Quel genre de contrôleur GRASP est JeuDeMonopoly dans l'exemple?
  1. Contrôleur de jeu
  2. Contrôleur MVC
  3. Contrôleur de session
  4. Contrôleur de façade

VALIDATION DE LA COMPRÉHENSION

- Selon la figure A17.3/F16.3, qui connaît un objet Joueur donné le nom du joueur?

Cours_Joueur
Cours
Joueur
Joueur
Joueur

3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?

CHP17 CONCEPTION AVEC LES GRASP

1. Qu'est-ce qu'une RDCU?
2. Quelle est l'utilité des contrats d'opérations dans une RDCU?
3. Est-ce que le retour d'information dans une opération du DSS est important pour la création d'une RDCU?
   - Pourquoi?

CHP18 VISIBILITÉ

1. Nommez les 4 types de visibilités.

CHP18 VISIBILITÉ

1. Nommez les 4 types de visibilités.
2. Comment déterminer l'ordre d'implémentation des classes?

CHP18 VISIBILITÉ

1. Nommez les 4 types de visibilités.
2. Comment déterminer l'ordre d'implémentation des classes?
   - Pourquoi?

EXERCICE RDCU

Correction

https://github.com/yvanross/LOG210-exercices

RDCU - AIDE MÉMOIRE

Voir la figure 8.1 des notes de cours.

AUTRES PATTERNS D'AFFECTATION DES RESPONSABILITÉS

LES NEUF PATRONS GRASP

- Expert
- Créateur
- Contrôleur
- Faible Couplage
- Forte Cohésion
- Polymorphisme
- Fabrication pure
- Indirection
- Protection des variations

POLYMORPHISME

- Principe de base pour la conception OO
- Cas qui varient mais qui sont similaires
- Dans une famille de classes
  - opération polymorphique
  - dans laquelle les cas sont différents
- Formes géométriques (carré, cercle, triangle, etc.)
  - se dessine avec une méthode draw()
  - chaque classe implémente sa méthode selon son cas…

POLYMORPHISME

Fig. 25.1 (3e édition)

POLYMORPHISME

Fig. 25.2 (3e édition)

POLYMORPHISME

Fig. 25.3 (3e édition)

POLYMORPHISME

Fig. 25.4, 25.5, 25.7 (3e édition)

FABRICATION PURE

- Parfois en suivant un patron
  - assigne une responsabilité
  - mais il y a des conséquences non désirables
    - augmentation du couplage
    - diminution de la cohésion
- Solution
  - fabrique une classe « comportementale »
  - qui n'a pas d'équivalent dans le modèle du domaine
  - invention de la part du concepteur

FABRICATION PURE

Fig. 25.8 (3e édition)

FABRICATION PURE

Fig. 25.9 (3e édition)

COMPRÉHENSION

Quel autre GRASP implique une fabrication pure? Autrement dit, on propose une classe « inventée » ne faisant pas partie du MDD.

1. Créateur
2. Contrôleur
3. Expert
4. Forte cohésion
5. Faible couplage

INDIRECTION

- Mécanisme souvent utilisé pour réduire le couplage
- Objet intermédiaire
  - découple deux composants

INDIRECTION

Date
TaxesSaisieAdapter
CGP service communication
getTaxes()
getTaxes(Taxes) : ...
...
...

- Composants (classes, interfaces, ...)
  - variations (ou instabilités) de ces composants en affectent d'autres
- Éliminer ces effets indésirables
- Identifier des zones de variabilité

PROTECTION DES VARIATIONS

- Créer une interface stable (API) autour des composants instables

Client
Client
Client

CLASSIFICATEURS UML

- Dans les DCC, il y en a deux classificateurs courants:
  - Les classes
  - Les interfaces

https://plantuml.com/class-diagram

NOTATION COURANTE (VUE EN LOGIC)

ATTRIBUTS EN UML

DÉCALAGE DE REPRÉSENTATION

RELATION DIAGRAMMES D'INTERACTION ET DE CLASSES

ORDRE D'IMPLÉMENTATION

RÉSUMÉ

- DCC servent comme plan pour le codage
  - Soyez à l'aise avec la transformation
- DCC se construisent au fur et à mesure que l'on réalise la dynamique (à travers les diagrammes d'interaction)

DEUX FAÇONS DE NOTER LES ATTRIBUTS

```
public class Registrar {
  private int id;
  private Sale currentSale;
  private Date timeline;
}
```

CLASSES COLLECTION

- Multiplicités des associations
- Besoin d'un groupe d'objets similaires
- Choix du type de collection
  - Influencer par le besoin

```
public class Sale{
  private ArrayList<LineItem> lineItems =
      new ArrayList<LineItem>();
}
```

AJOUTER UNE CLASSE DE COLLECTION

OPÉRATION VS MÉTHODE

- UML DCC : signature des opérations
  - Déclaration (décision de conception)
  - Opérations sont quasiment « vides » en UML
- Java : implémentation des méthodes
  - Implémentation (décision de codage)
- Déclaration d'**opération** (UML) est différente de l'implémentation de **méthode** (langage OO)
  - Pas un contrat d'opération dérivé les contraintes sur une opération et non une méthode
  - La différence est subtile mais importante

MOTS CLÉS, STÉRÉOTYPES, PROPRIÉTÉS ÉTIQUETTES

- Voir les sections A16.7/F15.7, A16.8/F15.8
  - Stéréotype <<create> , <destroy>, <metaclass>, <Actor>, <Interface>, ...
  - Le stéréotype décrire un ensemble d'étiquettes
- Étiquette: @login_required
  - Fonctionnalités et attributs que l'on veux donner à une classe ou une méthode.

NOTATION DE DÉPENDANCE

fig F15.9

```
class Sale
  public Date getDate();
  public void makeNewLineItem(
      ProductSpecification pSpec,
      int quantity)
  public void addLineItem(LineItem pLineItem);
```

- Interfaces et réalisation d'interface

CHP15 DIAGRAMME DE CLASSE

1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?

CHP15 DIAGRAMME DE CLASSE

1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?
2. Quelles sont les deux façons de noter les attributs?

CHP15 DIAGRAMME DE CLASSE

1. Quelle est la différence entre un diagramme de classes conceptuelles et logicielles?
2. Quelles sont les deux façons de noter les attributs?
3. Pourquoi fait-on un DCC?

VALIDATION DE LA COMPRÉHENSION

- Combien d'attributs pour Register et Sale ?
  - A. 1, 1
  - B. 2, 1
  - C. 1, 2
  - D. 2, 2

ARCHITECTURE EN COUCHE

1. Architecture logique **en couches** - 21.45m
2. Packaging - 34.58m
3. Valider l'architecture en couche du laboratoire - 10.19m

ARCHITECTURE LOGIQUE

Solution avec Java/Swing:

ARCHITECTURE LOGIQUE

Solution avec Navigateur/HTTP/ExpressJS/Typeorm/...

COMPRENDRE ET RESPECTER L'ARCHITECTURE EN COUCHES DU LABORATOIRE

https://log210-cuhrman.github.io/log210-valider-architecture-couches/

PACKAGING?

...

PACKAGING?

ATTRIBUT DE QUALITÉ

- Performance
- Disponibilité
- Testabilité
- Interopérabilité
- Sécurité
- Usabilité
- Modifiabilité

OBJECTIFS

- Organiser les packages pour réduire l'impact des **modifications**
- Apprendre d'autres notations des structures de packages UML

PRINCIPES

- Organiser les packages par **cohésion**
- Packager une **famille d'interface**
- Créer un package par **tâche** et par **goupe de classes instables** (Branches)
- Les plus responsables sont les plus stables
- Factoriser les type indépendants
- Utiliser des **fabrications** pour limiter la dépendance aux packages concrets
- Pas de cycles dans les packages DSM

GROUPEMENT PAR COHÉSION FONCTIONNELLE

- On groupe les types qui sont fortement apparentés, parce qu'ils participent à une finalité, un service, des collaborations, une politique et une fonction qu'ils ont en commun.
- Pex. le package "Tarification" contient des classes liées à la tarification des produits.
- C'est intuitif.

GROUPEMENT PAR COHÉSION RELATIONNELLE

- Mettre dans un package des classes qui se sont fortement couplées les unes aux autres.
- CB = (NombreDeRelationsInternes + 1) / NombreDeTypes

- GOF Facade pour améliorer la cohésion relationnelle avec les packages externes

COHÉSION RELATIONNELLE - EXEMPLE

LeBrasDes

Relational Cohesion (H) average number of internal relationships per type

$H = (R + 1) / N$, where

R = number of type relationships internal to the package

N = number of types in the package

réciproqué: métrique de cohésion

PACKAGER UNE FAMILLE D'INTERFACES

- Placez une famille d'interfaces fonctionnellement liées dans un package à part
- Cela vaut pour les familles de trois interfaces ou plus
- Exemple: java.ejb comprend au moins douze

```
javax.ejb.AccessLocalException
javax.ejb.AccessLocalException.AccessLocalException()
javax.ejb.AccessLocalException.AccessLocalException(java.lang.Exception)
javax.ejb.AccessLocalException.AccessLocalException(java.lang.Exception,boolean)
javax.ejb.AccessLocalException.AccessLocalException(java.lang.String)
javax.ejb.AccessLocalException.AccessLocalException(java.lang.String,java.lang.Exception)
javax.ejb.AccessLocalException.AccessLocalException(java.lang.String,java.lang.Exception,boolean)
javax.ejb.AccessLocalException.getCause()
javax.ejb.AccessLocalException.initCause(java.lang.Throwable)
javax.ejb.AccessLocalException.toString()
javax.ejb.AccessLocalException.printStackTrace()
javax.ejb.AccessLocalException.printStackTrace(java.io.PrintStream)
javax.ejb.AccessLocalException.printStackTrace(java.io.PrintWriter)
javax.ejb.AccessLocalException.getMessage()
javax.ejb.AccessLocalException.getLocalizedMessage()
javax.ejb.AccessLocalException.fillInStackTrace()
javax.ejb.AccessLocalException.getStackTrace()
javax.ejb.AccessLocalException.setStackTrace(java.lang.StackTraceElement[])
```

CRÉEZ UN PACKAGE PAR TÂCHE DE LIVRAISON

- Les packages sont l'unité de base de développement et de livraison (on ne développe rarement qu'une seule classe)
- Créer un package par tâche de livraison

... ET PAR GROUPE DE CLASSES INSTABLES (2/3)

- Si un ensemble de classes dans un package a une tendance à évoluer ensemble, p.ex. on remarque qu'à chaque fois que c'est toujours les mêmes 5 ou 10 classes dans un package qui ont été modifiées.
- Exemple concret – chaque fois que l'on change le schéma de la BD il y a des changements dans plusieurs classes, surtout les appels à la méthode "sqlquery()"
  - Créez un package dans le package (sous package) pour isoler les classes ayant une tendance à changer ensemble
  - Notez que la tendance de ce genre de changement n'est pas visible dans les premières itérations.

LES PLUS RESPONSABLES SONT LES PLUS STABLES

Les packages « les plus responsables » engendrent le **plus de dépendances** et devraient être plus stables.

Les classes stables
Les classes instables
Les classes responsables
Les classes irresponsables
Les classes responsables
Les classes irresponsables
Les classes stables
Les classes inst

Forme.Forme.mpForme = Forme.FabriqueForme.getNewForme("cercle");

PRINCIPE: PAS DE CYCLES DANS LES PACKAGES

- Si un groupe de packages a une dépendance cyclique, on peut essayer de traiter comme un seul grand package. Cela n'est pas souhaitable, mais il arrive de packages volumineux (ou d'agrégats de packages) apparaît le risque d'effets refaçon.
- Factorisez ou bypassez pour casser un cycle dans un package plus petit.
- Rompez le cycle à l'aide d'une interface.

Figure 8.9 – Rupture d'un dépendance cyclique

COMBINAISON DES PATRONS DE PACKAGING

nd: Commonly used architectural patterns in Java applications

QU'EST-CE QU'UNE COUCHE?

- Groupement à forte granularité de classes, packages et sous-systèmes
- Possède un thème « cohésif »
- Une couche plus « haute » fait appel aux couches plus « basses », mais pas l'inverse
- Couches OO typiques
  - Présentation (IHM)
  - Logique applicative (objets du domaine)
  - Services techniques (persistance, journalisation, etc.)

CONCEVOIR UNE ARCHITECTURE EN COUCHES

- Séparation nette et cohésive
- Collaboration et couplage vont du haut vers le bas

fig F.12.2, A13.2

- Séparation Modèle-Vue est un cas simple

ARCHITECTURE - MODÈLE / VUE / CONTRÔLEUR

- Vues/contrôleurs modifient le modèle
- Modèle informe les vues des changements
- Vues se mettent à jour de nouveau en conséquence

AVANTAGES DE LA MODÉLISATION EN COUCHES

- Séparation de fonctions/services -> couplage réduit, cohésion améliorée
- La complexité est encapsulée (décomposable)
- On peut remplacer certaines couches
- Les couches basses contiennent des fonctions réutilisables
- Certaines couches (Domaine) peuvent être distribuées
- Segmentation logique facilite développement en équipe

RELATION ENTRE COUCHE DOMAINE ET DOC

RELATION ENTRE DSS ET COUCHES

System
UI
Presentation
Business Logic
Persistance
Technical Services
Client
Client
Serveur
Serveur
Client
Serveur
Client
Serveur
Client

https://github.com/yvanross/LOG210-exercices

EXERCICES EN LIGNE

https://github.com/yvanross/LOG210-exercices

MODULE TYPESCRIPT

1. Exemmples TypeScript (5GB) - 24.55m

Science | Quiz | Laboratoire | Intro | Equipe | PKI | DS | GCLD | NoSQL | BD | Central | DDD | CQRS | Cloche | Exercice | Typescript | Complexité | FFUPPS | TOO | GOF | Diagrammes | Divers

EXEMPLES TYPESCRIPT

Dépôt github
https://github.com/profcuhrmanets/exemples-ts/

ARROW FUNCTION (JAVASCRIPT)

src/arrow_function.ts

ARRAY.PROTOTYPE.MAP() (JAVASCRIPT)

src/array_map_arrow.ts

TABLEAU ASSOCIATIF MAP() (JAVASCRIPT)

src/tableau_associatif.ts

ACCÈS SGB

src/get_students_fetch.ts

⚠️ DEUX "MAP"

Le sens est différent:

- Tableau associatif `Map<>()`
  - https://howtodoinjava.com/typescript/maps/
- `Array.prototype.map()`
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Ob

MODULE COMPLEXITÉ

1. Complexité - Inhérente et accidentelle - 23.04m

Science | Quiz | Laboratoire | Intro | Equipe | PKI | DS | GCLD | NoSQL | BD | Central | DDD | CQRS | Cloche | Exercice | Typescript | Complexité | FFUPPS | TOO | GOF | Diagrammes | Divers

COMPLEXITÉS INHÉRENTE ET ACCIDENTELLE

OBJECTIF

Vous sensibiliser aux conséquences de vos choix de conception sur le plan de la complexité

DÉFIS POSÉS PAR LA COMPLEXITÉ

- La complexité est l'adversaire de tout développeur logiciel
- On peut distinguer entre deux sources de complexité (F. Brooks)
- Le domaine du problème
  - « complexité inhérente (essentielle) »
- La solution (logicielle)
  - « complexité accidentelle »

COMPLEXITÉS

- Inhérente (essentielle)
  - « Essentielle » selon Aristote
  - Pour « être » une voiture, une voiture a un moteur, des roues, des portes, etc.
- Accidentelle (due à des choix)

COMPLEXITÉ INHÉRENTE

- Le domaine du problème concernant le logiciel peut être complexe
  - Exemple : système d'exploitation pour ordinateur
- La complexité inhérente est due au « problème » et non à la solution
  - Linux est complexe parce qu'il doit supporter plein de fonctionnalités...
- Synonyme : « complexité essentielle »

COMPLEXITÉ INHÉRENTE

Classification of Software Complexity
Complexity Factors
Complexity Measures
Complexity Management
Complexity Trends
Complexity Laws
Complexity Theorems
Complexity Models
Complexity Metrics
Complexity Analysis
Complexity Tools
Complexity Standards
Complexity Training
Complexity Education
Complexity Certification
Complexity Consulting
Complexity Outsourcing
Complexity Offshoring
Complexity Litigation
Complexity Expert Witnesses
Complexity Patents
Complexity Obsolescence
Complexity Maintenance
Complexity Enhancements
Complexity Documentation
Complexity Reuse
Complexity Refactoring
Complexity Reengineering
Complexity Redesign
Complexity Redevelopment
Complexity Replacement
Complexity Retirement
Complexity Recovery
Complexity Risks
Complexity Rewards
Complexity Research
Complexity Publications
Complexity Conferences
Complexity Journals
Complexity Blogs
Complexity Wikis
Complexity Forums
Complexity Groups
Complexity Societies
Complexity Associations
Complexity Institutes
Complexity Universities
Complexity Colleges
Complexity Schools
Complexity Degrees
Complexity Certificates
Complexity Jobs
Complexity Careers
Complexity Salaries
Complexity Bonuses
Complexity Stocks
Complexity Options
Complexity Futures
Complexity Derivatives
Complexity Insurance
Complexity Warranties
Complexity Guarantees
Complexity Liabilities
Complexity Assets
Complexity Capital
Complexity Investment
Complexity Venture
Complexity Angel
Complexity Seed
Complexity Startup
Complexity Incubator
Complexity Accelerator
Complexity Grants
Complexity Awards
Complexity Prizes
Complexity Contests
Complexity Challenges
Complexity Hackathons
Complexity Bootcamps
Complexity Boot Camp
Complexity Camp
Complexity Conference
Complexity Convention
Complexity Exposition
Complexity Show
Complexity Trade Show
Complexity Fair
Complexity Festival
Complexity Carnival
Complexity Circus
Complexity Zoo
Complexity Aquarium
Complexity Museum
Complexity Gallery
Complexity Exhibition
Complexity Tour
Complexity Vacation
Complexity Holiday
Complexity Leisure
Complexity Recreation
Complexity Sports
Complexity Games
Complexity Toys
Complexity Hobbies
Complexity Crafts
Complexity Art
Complexity Music
Complexity Dance
Complexity Theater
Complexity Film
Complexity Photography
Complexity Video
Complexity Podcasts
Complexity Radio
Complexity TV
Complexity Internet
Complexity Web
Complexity Mobile
Complexity Social Media
Complexity Networks
Complexity Communities

- Réduire l'effort cognitif en gérant la complexité
  - Abstraction (OO)
  - Encapsulation (OO)
  - Masquage de l'information (OO)

ABSTRACTION ET ENCAPSULATION

ABSTRACTION

Vue plus simple

ENCAPSULATION

Accès limité

PATTERNS POUR GÉRER LA COMPLEXITÉ

- Proposer des solutions aux problèmes récurrents dans un contexte particulier

DESIGN PATTERNS SONT UNE PARTIE DE LA SOLUTION

- Proposer des solutions aux problèmes récurrents dans un **contexte particulier**

SOCRATIVE OU ZOOM

Nom de la salle de class: **ETSDESIGN**

VALIDATION DE LA COMPRÉHENSION

- En développement logiciel, la complexité inhérente est due au domaine du problème et cause des complexités dans la solution.
  - Vrai?
  - Faux?

VALIDATION DE LA COMPRÉHENSION

- La complexité inhérente est de mieux en mieux gérée grâce aux innovations technologiques.
  - Vrai?
  - Faux?

VALIDATION DE LA COMPRÉHENSION

- Le déverminage d'un algorithme de parcours d'arbre représentant une expression mathématique est difficile à cause de la complexité accidentelle
  - Vrai?
  - Faux?

VALIDATION DE LA COMPRÉHENSION

- Les outils de développement comme Eclipse répondent principalement aux difficultés due à la complexité inhérente.
  - Vrai?
  - Faux?

VALIDATION DE LA COMPRÉHENSION

- Qu'est-ce qui n'est pas une technique pour gérer la complexité inhérente?
  1. Masquage de l'information
  2. Encapsulation
  3. Ramasse-miettes (récupérateur de mémoire)
  4. Abstraction
  5. Généralisation

RÉSUMÉ

- Soyez attentif aux sources de complexité dans les logiciels que vous développez
- La complexité inhérente ne peut pas être éliminée si votre problème est complexe
- Cependant, elle peut être gérée avec un bon choix d'abstractions.
- Patrons utilisent des abstractions et de l'encapsulation pour gérer la complexité
- Cependant, ils amènent aussi de la complexité accidentelle
- Vérifier:
  - Le problème récurrent que le patron est censé résoudre existe-t-il (Facile)
  - Le coût de la complexité due au patron est acceptable par rapport aux bénéfices (moins facile)

MODULE EXIGENCES FURPS

1. FURPS - 39.44m

Séance | Quiz | Laboratoire | Intro | Cycles | PO | OO | CU | Ex | MVC | Ass | Central | MDD | GO_| Cache
| Exercice | Typescript | Complexité | FURPS | TDD | GOF | Diagrammes | Divers

FURPS - EXIGENCES DU CLIENT

- FURPS+
  1. Fonctionnalité (Use case)
  2. Usability (Aptitude à l'utilisation, facteurs humains, aide et documentation)
  3. Reliability (Fiabilité)
  4. Performance
  5. Supportability (Possibilités de prise en charge, adaptabilité, maintenabilité, etc.)

FURPS ET PATRONS

L'application d'un patron doit être motivée par des exigences:

- *Façade* favorise la modifiabilité du code (**S**)
- *Memento* favorise l'implémentation de « Undo » (**F** et **U**)
- *Stratégie* favorise des « plug-in » (**S**)
- *Stratégie* peut faciliter la mise en place de la redondance de l'information (**R** et **P**)
- *Flyweight* favorise une meilleure performance (**P**)

DÉFINITION DES BESOINS

- Cas d'utilisation
  - Descriptions des besoins fonctionnels
  - Le « F » dans FURPS

ÉNONCÉ LABORATOIRE

Le modèle FURPS est utilisé.

Plus d'exemples.

COMMENT ORGANISER LES BESOINS DANS LE FURPS

- Plusieurs artefacts décrivent les besoins

Usage Model
Use Cases
Supplementary Requirements
Domain Model
Glossary
Business Rules
Quality Attributes
Usability
Reliability
Performance
Supportability
Design Rules
Implementation Rules
Interface Rules
Domain Rules

RÉSUMÉ

- Les besoins ne sont pas juste les fonctionnalités
  - Besoins non fonctionnels (qualités) souvent ignorés: convivialité, fiabilité, extensibilité, etc.
- FURPS+ facilite la classification des besoins
- Définition des besoins est itérative dans le processus unifié

MODULE TDD

1. TDD - Développement piloté par les test - 15.01m
2. TDD en pratique

KATA TDD

- Kata TDD avec TypeScript
- Expérimenté avec SGB
  - npm run test
  - npm run watch
  - npm run test -- -g "nom ou partie du nom"
    - https://github.com/ylvanross/hqc2110-systeme-gestion-bordereau-node-express-ts

TDD LABORATOIRE
TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.

TDD LABORATOIRE
TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.
2. You are not allowed to write more of a unit test than is sufficient to fail and not compiling is failing.

TDD LABORATOIRE
TDD 3 LAWS

1. You are not allowed to write any production code at all until you have written a failing unit test.
2. You are not allowed to write more of a unit test than is sufficient to fail and not compiling is failing.
3. You are not allowed to write more production code than is sufficient to pass the currently failing test. These three laws give us the perfect low-level documentation for a system code examples.

TDD

1. Kata TDD avec TypeScript

- Développer un fonction permettant d'extraire les mots uniques d'une phrase
- Développer une fonction permettant de faire l'intersection des mots de deux phrases

MODULE GOF

1. Qu'est ce qu'un design pattern? - 14.25m
2. Adaptateur, Fabrique concrète, Singleton - 9.21m
3. Logique de tarification élaborée avec patron Stratégie et Composite - 18.57m
4. Actualisation interface usager - 6.21m
5. Révision Observateur - 8.47m
6. Découverte des patrons GOF - 10.23m
7. Recouvrement avec Proxy - 16.45m
8. Attention à la patternite
9. GRASP dans les GoF 7.27m

QU'EST CE QU'UN DESIGN PATTERN?

- Solution à un problème récurrent dans un contexte particulier

TROIS DÉFIS POUR BIEN UTILISER LES PATTERNS

1. (Re)connaître le problème
2. Comprendre le patron (la solution)
3. Appliquer la solution

1.(RE)CONNAÎTRE LE PROBLÈME

- Comprendre le problème
- Exercice de « croyance » (on débute en OO)
- Certains problèmes sont plus « récurrents » que d'autres...
- Auteurs (GoF) avaient beaucoup d'expérience en OO

2.COMPRENDRE LE PATRON (LA SOLUTION)

- abstractions
- notation UML est importante
- aspects statique et dynamique

3. APPLIQUER LA SOLUTION

- Étape concrète
- Mise en contexte particulier

3. APPLIQUER LA SOLUTION

Nom dans le patron	Vrai nom (Bouton Swing)
Subject	JButton
Observer	ActionListener
ConcreteObserver	la classe implémentant l'interface ActionListener
`attach()`	`addActionListener()`
`notify()`	`actionPerformed()`

PATRONS VUS DANS LE LIVRE DE LOG121

- **Adaptateur**
- Commande
- Composite
- Décorateur
- Itérateur
- Méthode fabrique
- Observateur
- **Proxy**
- Singleton
- Stratégie
- Template
- Visiteur

PERSPECTIVE ZEN

DESIGN PATTERN RELATIONSHIPS

Ref: Design Patterns, Element of Reusable Object-oriented Software

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate

- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues
- Vérifier après l'application du pattern que les bénéfices sont vraiment là

RÉSUMÉ

- Reconnaître des problèmes récurrents est difficile faute d'expérience adéquate
- Expérimenter avec les patterns (dans une version du projet qui n'est pas critique)
- Un patron ne s'enlève pas facilement!
- Discuter de leurs avantages et leurs inconvénients avec des collègues
- Vérifier après l'application du pattern que les bénéfices sont vraiment là
  - p.ex. il est plus facile d'étendre le logiciel

PATRON GOF LOG121

- Décorateur
- Itérateur
- Commande
- Façade
- Singleton
- Template method
- Factory method
- Proxy
- Observateur
- Visiteur

PATRONS GOF

Quiz patron GOF

*PROBLÈME

- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

Patron GOF?

ADAPTATEUR (GOF)

*PROBLÈME

- POS NextGen doit supporter: (1) calcul des taxes, (2) autorisation de crédit, (3) comptabilité, (4) gestion des stocks
- Services trop compliqués pour le périmètre du projet → Services tiers « externes »
- Plusieurs fournisseurs de service (comme *TaxMaster* et *GoodAsGoldTaxPro*)
- L'API de chaque service est **distincte** et **immuable**

Patron GOF?

ADAPTATEUR (GOF)

- Problème (suite)
  - On veut éviter ce genre de solution:

```
if (TaxMaster) {
  taxes = tm.calculateTaxes(montant);
} else if (GoodAsGold) {
  taxes = gag.computeTaxes(montant);
} else if [...] {
  ...
}
```

UNE de chaque service est détectée et invoquée

UNE de chaque service est détectée et invoquée

GRASP Polymorphisme

ADAPTATEUR POUR LE CALCUL DES TAXES

fig. F14.21

ADAPTATEUR

fig. F22.1/A25.1

ADAPTATEUR (GOF)

Client
Client
Client

Documentation System_getPropertyAt()
Documentation System_getPropertyAt()
Documentation System_getPropertyAt()

- *Stratégie* → *Polymorphisme* (GRASP)
- Fabriques sont souvent utilisées pour créer les stratégies pour autoriser des algorithmes insérables

CONFLITS DUS AUX POLITIQUES TARIFAIRES

- Politiques de prix pour aujourd'hui (lundi)
  - 20% de rabais pour les personnes du troisième âge
  - 15% de rabais sur un achat supérieur à 400$
  - 50 dollars de rabais pour sur un achat supérieur à 500$ (ce lundi seulement)
  - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
  - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600$

Patron GOF?

COMPOSITE

CONFLITS DUS AUX POLITIQUES TARIFAIRES

- Politiques de prix pour aujourd'hui (lundi)
  - 20% de rabais pour les personnes du troisième âge
  - 15% de rabais sur un achat supérieur à 400$
  - 50 dollars de rabais pour sur un achat supérieur à 500$ (ce lundi seulement)
  - Si on achète un carton de thé Darjeeling, il y a un rabais de 15% sur tout
- Comment résoudre les cas conflictuels?
  - Une dame de 77 ans qui est aussi client privilégié achète une caisse de Darjeeling et dépense 600$

Patron GOF?

FACTEURS DES STRATÉGIES

- Période du temps (lundi)

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas
  - Prix le plus haut

FACTEURS DES STRATÉGIES

- Période du temps (lundi)
- Type de client (personne du 3e âge)
- Un produit particulier (thé Darjeeling)
- Approche pour résoudre les conflits
  - Prix le plus bas
  - Prix le plus haut
  - Etc.

- Contexte / Problème
  - partois on traite un seul objet atomique
  - partois on traite un groupe composition d'objets
  - les traiter de la même façon
- Solution
  - définir des classes pour les objets
    - composites
    - atomiques
  - implémentant la même interface

Patron GOF?

COMPOSITE

- Contexte / Problème
  - partois on traite un seul objet atomique
  - partois on traite un groupe composition d'objets
  - les traiter de la même façon
- Solution
  - définir des classes pour les objets
    - composites
    - atomiques
  - implémentant la même interface

Patron GOF?

COMPOSITE POS

COMPOSITE – COLLABORATION

RÉALISATION DE CAS D'UTILISATION

RÉALISATION DE CAS D'UTILISATION

a:Sale
:makeNewSale()
:enterItem(id, quantity)
:specification(id, storeID)
:description
:price
:totalWithTax()
:totalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:makeLineItem(p, n)
:LineItem
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:makeLineItem(p, n, this)
:ComplexLineItem
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:enterItem(id, quantity)
:makeLineItem(p, n)
:LineItem
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:getPrice()
:endSale()
:totalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:getTotalWithTax()
:makePayment(cash)
:makeNewPayment(cash)
:Payment
:Payment(cash)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:addPayment(p)
:getBalance()
:getBalance()
:getBalance()

- Contexte / Problème
  - objet observateur
    - réagit selon les changements du sujet
  - objet sujet
    - annonce un changement
- miniminser le couplage du sujet vers l'observeur

OBSERVATEUR

- Contexte / Problème
  - objet observateur
    - réagit selon les changements du sujet
  - objet sujet
    - annonce un changement
- miniminser le couplage du sujet vers l'observeur

OBSERVATEUR

- Solution
  - interface observateur
  - les observateurs s'inscrivent au sujet
  - le sujet ne connait pas les observateurs particuliers
    - seulement des objets implémentant l'interface
  - le sujet envoie un message aux observateurs
    - lorsqu'un certain changement (événement) survient

OBSERVATEUR

Plusieurs observateurs, un sujet

(cf Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995

OBSERVATEUR

Communication

ref: Design Patterns, Gamma, Helm, Johnson & Vlissides, 1995

SOLUTION : OBSERVATEUR

fig. F26.22

EXERCICE OBSERVATEUR

- Rappeler le patron Observateur
- Réviser le diagramme de structure
- Réaliser les diagrammes de comportement
  - Communication
  - Activité
  - Séquence

SOLUTION : OBSERVATEUR STRUCTURE

fig. F26.22

OBSERVATEUR COMMUNICATION

Voir la note pour la solution

OBSERVATEUR ACTIVITÉ

Voir la note pour la solution

OBSERVATEUR SÉQUENCE

Voir la note pour la solution

"DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION

"DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION

DÉCOUVERTES « D'ANALYSE » LORS DE LA CONCEPTION

- Calculateurs de taxes retournent LigneTaxe
  - pas envisagé lors de l'analyse initiale
- Concept dans le modèle du domaine
  - mettre à jour le modèle du domaine
  - mettre à jour le glossaire
- Avec l'expérience ceci devient un réflexe
  - travail d'un ingénieur (vs technicien ou codeur)
  - nécessaire dans un processus itératif

ATTRIBUTS DE QUALITÉ

- Appliquer les GoF et GRASP dans les RDCU pour traiter les attributs de qualité (exigences URPS)
  - NextGen
    - basculement sur les services locaux,
    - gestion du terminal POS et
    - autorisations de crédit
  - Monopoly
    - différentes cases
    - achat de propriétés
    - paiement de loyers

EXIGENCES NEXTGEN

- En cas d'échec d'un service distant, le système doit continuer à fonctionner (tolérance aux pannes, **fiabilité**)
- Mise en cache locale (**performance**)
- Prise en charge d'équipements POS tiers, notamment différents scanners (**adaptabilité**)
- Gestion des paiements par carte de crédit, carte de débit ou chèque (**adaptabilité**)

PROBLÈME : FIABILITÉ – REPRISE SUR DÉFAILLANCE

- Tolérance à la panne de services distants
- Plusieurs scénarios de panne

RECOUVREMENT – MOTIVATION

- Les propriétaires des magasin ne veulent pas manquer des ventes en cas de panne
- Le développeur NextGen POS sait que les concurrents n'offrent pas ce degré de fiabilité

REPRISE SUR DÉFAILLANCE

- Récupération robuste en cas de défaillance
  - d'un service distant (calculateur de taxes, gestion des stocks...)
  - de la BD des produits (descriptifs et prix)

PRINCIPE DE RECOUVREMENT

Exemple: Pneu de secours permet de continuer sans un délai, perte de performance (pneu plus petit)

CONSEILS DE L'ARCHITECTE : CACHE LOCAL

- Augmente performance (et permet récupération quand l'accès à BD échoue)
- Utilise un cache local (simple fichier sur disque) des objets DescriptionProduit
- Conséquence : toujours consulter le cache local pour chercher les informations avant d'accéder au service distant
- Principe informatique « Cache »
  - http://en.wikipedia.org/wiki/Cache_(computing)
  - http://fr.wikipedia.org/wiki/Mémoire_cache

RECOUVREMENT - PRODUCT SPECIFICATION

- Recherche d'une spécification d'article
  - informations locales
    - performance
    - recouvrement
  - puis requête au système distant

RECOUVREMENT - PRODUCT SPECIFICATION

- Plusieurs niveaux de caches
  - mémoire vive (~1000)
    - performance
  - disque dur (~100 MB)
    - pannes de courant
  - serveur local au commerce
    - indépendance au réseau
  - web
  - etc.

RAPPEL – ADAPTATEURS

fig. F23.1

RAPPEL – ADAPTATEURS

- FabriqueDeServices retourne un adaptateur pour accéder aux variantes de services BD Produits

RAPPEL – ADAPTATEURS

- FabriqueDeService retourne un adaptateur pour accéder à la BD Produits

Utilisation du cache local avant BD
Magasin
Pagelet
Proxy
getAdaptateurProduits()
getAdaptateurProduits()
Proxy
Proxy
Proxy

Bridge	✓	✓	✓	✓

TABLE 7: GRASP DANS LES GOF

Pattern GoF	Indir.	Poly.	Fab. Pure	Vs. V.
Composite	✓	✓		✓
Decorator	✓	✓		✓
Facade	✓		✓	✓
Flyweight	✓	✓	✓	✓
Proxy	✓	✓	✓	✓
Chain of responsibility	✓	✓	✓	✓

TABLE 7: GRASP DANS LES GOF

Pattern GoF	Indir.	Poly.	Fab. Pure
Command	✓	✓	✓
Interpreter	✓	✓	✓
Iterator	✓	✓	✓
Mediator	✓	✓	✓
Memento	✓		✓
Observer	✓	✓	✓
State	✓	✓	✓

TABLE 7: GRASP DANS LES GOF

Pattern GoF	Indir.	Poly.	Fab. Pure	Vs. V.
Strategy	✓	✓		✓
Template method	✓	✓		✓
Visitor	✓	✓	✓	✓

IDENTIFICATION DES GRASP DANS LES GOF

https://docs.google.com/document/d/1Mmi0VFzzSK4Uusp=sharing

SOLUTION

https://docs.google.com/document/d/1pe3RTLkBFiHErDrne9et8qoyTlnNfcA23HIT-o/edit?usp=sharing

MODULE AUTRE DIAGRAMME UML

1. Diagramme d'intéraction - 9.33m
2. Diagramme d'état - 26.34m
3. Diagrammes d'état - 2.54m
4. Diagramme d'activité - 16.01m
5. Diagrammes d'activité - 1.18m
6. Diagrammes de composants - 17.37m
7. Diagrammes de déploiement - 10.02m
8. Révision diagrammes UML - 16.57m

Séance | Quiz | Laboratoire | Intro | Espace | PV | DCD | CU | VOD | Dos | Caveat | BDC | GSO, | Cheche | Exercice | Tutoriel | Composants | PUPPY | TDD | GOF | Diagrammes | Divers

DIAGRAMMES D'INTERACTION MODÉLISENT L'ACTION

Diagrammes de séquence

Diagrammes de communication

DIAGRAMME D'INTERACTION

- Pour détailler les opérations
- Annoté avec les GRASP

COMPARAISON

Type	Forces (+)	Faiblesses (-)
Séquence	Indique clairement la séquence et l'ordonnancement des messages. Grande richesse de la notation	Ajout de nouveaux objets doit être vers la droite; consomme l'espace horizontal
Communication	Économique en termes d'espace, permet d'ajouter des objets dans les deux dimensions	Rend plus difficile la lecture des séquences de messages. Moins d'options de notation

RAPPEL DE LOG121

CLASSE VS. INSTANCE

NOTATION: DIAGRAMMES DE SÉQUENCE

RÉSULTAT D'UN MESSAGE

MESSAGE D'UN OBJET À LUI-MÊME

DESTRUCTION D'UN OBJET

ITÉRATION SUR UNE COLLECTION (NOTATION EXPLICITE)

fig F14.16, A15.16

ITÉRATION SUR UNE COLLECTION (NOTATION IMPLICITE)

IMBRICATION DE CADRES

APPELS ASYNCHRONES ET OBJETS ACTIFS

DIAGRAMME DE COMMUNICATION

- Liens et messages

DIAGRAMME DE COMMUNICATION

- Liens et messages

Note: Voir solution dans la section exercice plus bas.

MESSAGE POUR UNE CLASSE

MESSAGE D'UN OBJET À LUI-MÊME

CRÉATION D'INSTANCE

AFFECTATION DE NUMÉROS D'ORDRE

AFFECTATION DE NUMÉROS D'ORDRE

ITÉRATION SUR UNE COLLECTION

collection en java

MESSAGE VERS UNE CLASSE

MESSAGES POLYMORPHES

*EXERCICE

TRANSFORMER EN DIAGRAMME DE SÉQUENCE

larman/F14.24, A15.24

PARALLÉLISME: DIAGRAMME DE COMMUNICATION UML

- For example,
  - messages 2.3a and 2.3b are concurrent within activation 2.3,
  - Name represents a concurrent thread of control,
  - Instance of A sends draw() messages concurrently to instance of B and to instance of C.
- Ref: https://www.uml-diagrams.org/communication-diagrams.html

DIAGRAMME DE COLLABORATION UML

Alternative
CAP
Alternative Operational Scenarios
Alternative Operational Scenarios
Service 1 Operational Scenarios
Service 2 Operational Scenarios
Service 3 Operational Scenarios
Service 1 Operational Scenarios
Service 2 Operational Scenarios
Service 3 Operational Scenarios
Service 1 Operational Scenarios
Service 2 Operational Scenarios
Service 3 Operational Scenarios
Service 1
Service 2
Service 3
Service 1
Service 2
Service 3
Service 1
Service 2
Service 3

MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS

Objets qui changent de rôle, comme une personne...

- état de civil ou de militaire
- résident temporaire, résident permanent, citoyen

MODÉLISATION D'OBJET ÉTAT-DÉPENDANTS

- Protocoles de communication
  - p.ex. en TCP, il le gestionnaire du protocole est dans l'état « fermé », un message « close » sera ignoré
- Flot page/fenêtre IHM
  - séquence légale des pages Web ou des fenêtres, souvent complexe
- Contrôleurs de flot ou de sessions
  - objets « session » p.ex. application Web
- Ordre des opérations système d'un cas d'utilisation
  - créerNouveauWebsite, saisirArticle, terminerVente, etc.
- Gestionnaire de flot de contrôle d'une application ou une fenêtre d'une IHM
  - séquences légales d'une fenêtre ou d'un formulaire (coller n'est pas valide si presse-papiers est nul)

NOTATION ADDITIONNELLE

Transitions peuvent avoir des actions et des conditions de guard

NOTATION ADDITIONNELLE

États imbriqués

- Pouvez vous réaliser le diagramme de classe correspondant?

*AUTOMATE FINI ET CAS D'UTILISATION 1/2

UML Trader Home
Trader
Trader Home
Trader Home
Trader
Trader
Trader

reprend sa carte, le système ouvre la barrière, le client entre pour se stationner, le système ferme la barrière. Le client se présente à la sortie, présente sa carte de crédit, le système imprime le recu, le client reprend sa carte de crédit, prend le recu, le système ouvre la barrière, le client peut alors sortir, le système détecte la sortie et ferme la barrière.

EXERCICES DIAGRAMME D'ACTIVITÉ

- Retour de voiture louée
- Recette de cuisine
- Retrait au guichet automatique
- Ordinateur de plongée
- Demander un remplacement
- Processus d'achat sur le web
- Vendre au comptoir

https://github.com/yvanrosu/LOG210-exercices

DIAGRAMME D'ACTIVITÉ

Comment implémenter?

Business Process Modeling Notation (BPMN)

1. BPMN Demo
2. UML
3. Derek Banas, Activity Diagram

DIAGRAMME D'ACTIVITÉ
EXERCICE RECETTE DE CUISINE

- En équipe de 4, réaliser le diagramme d'activité en suivant les étapes identifiées dans l'exercice suivant:

https://github.com/yvanrosu/LOG210-exercices/blob/master/activite/activite-recette-de-cuisine.md

DIAGRAMMES DE COMPOSANTS

Un composant n'est pas clairement défini dans l'UML:

« Un composant est la **partie modulaire** d'un système qui encapsule son contenu et dont la manifestation est **remplaçable** dans son environnement. Il définit son **comportement** en termes d'**interfaces** fournies et requises. Ainsi, un composant peut être utilisé comme un type dont la conformité est définie par ces interfaces fournies et requises. »

EXEMPLE EN PLANTUML

Figure F31.2 Composants UML

(PlantUML)

EXISTE-T-IL RÉELLEMENT DES COMPOSANTS LOGICIELS?

They have to exist. Sales and marketing people are talking about Components. Components are not a technology. Components are about how customers want to relate to software.

- They want to be able to buy their software a piece at a time, and to be able to upgrade it just like they can upgrade their stereo.
- They want to be able to mix and match components from vendors, and to be able to upgrade on their own schedule, not the manufacturer's schedule.
- They want to be able to mix and match pieces from various manufacturers. This is a very reasonable requirement. It is just hard to satisfy.

Ralph Johnson do component exist

MÉTAPHORE CINÉMA MAISON

1. Heroku
2. IOT

- IFTTT

vers2.photoshop

DIAGRAMME DE COMPOSANTS

fig. 31.2 AIF.2

D'AUTRES EXEMPLES

fig. F14.2 UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition

EXERCICES DIAGRAMME DE DÉPLOIEMENT

1. Application web
2. Tutorial online
3. UMLdeployment diagram - Apple iTunes
4. UML Deployment Diagrams Examples

Prenez le temps regarder la documentation de planUml pour les diagrammes de déploiement et de composants. Nous ferons un exercice la semain prochaine.

- https://plantuml.com/deployment-diagram
- https://plantuml.com/component-diagram

DIAGRAMMES DE DÉPLOIEMENT

Documenter (1) comment les fichiers exécutables seront affectés sur les nœuds de traitement et (2) la communication entre composants physiques

Figure 10.1 Diagramme de déploiement montrant les nœuds et les connexions entre les nœuds
Figure 10.2 Diagramme de déploiement montrant les artefacts

TYPES DE NŒUDS

Nœud physique (équipement)

Ressource de traitement physique (p.ex. de l'électronique numérique), dotée de services de traitement et de mémoire destinés à exécuter un logiciel. Ordinateur classique, cellulaire, etc.

TYPES DE NŒUDS

Nœud d'environnement d'exécution (EEN *execution environment node*)

Ressource de traitement logiciel qui s'exécute au sein d'un nœud externe (comme un ordinateur) et offrant lui-même un service pour héberger et exécuter d'autres logiciels.

NŒUD D'ENVIRONNEMENT D'EXÉCUTION

- Système d'exploitation (OS) est un logiciel qui héberge et qui exécute des programmes
- Machine virtuelle (JVM ou .NET)
- Moteur de base de données (p.ex. PostgreSQL) exécute les requêtes SQL
- Navigateur Web héberge et exécute JavaScript, applets Flash/Java
- Moteur de workflow
- Conteneur de servlets ou conteneur d'EJB

DIAGRAMME DE DÉPLOIEMENT

Client
Client
Client

MÉTHODOLOGIE DE DÉVELOPPEMENT UML

- UML est un langage de modélisation général qui inclut un large éventail de concepts, notations et diagrammes.
- Avant d'intégrer UML dans un processus de développement , il est fondamental de définir correctement une méthodologie pour prendre en charge son utilisation.
  - La définition d'une méthodologie de développement facilitera l'intégration et l'adoption d'UML , augmentera la qualité des modèles et augmentera la productivité des développeurs.

MÉTHODOLOGIE DE DÉVELOPPEMENT UML

- Pour chaque phase de développement , vous devez définir
  - Ensemble de diagrammes UML utilisés dans la phase
  - Rôle de chaque diagramme
  - Sous-ensemble de la notation UML utilisée
  - Remarque: certains diagrammes peuvent jouer différents rôles dans différentes phases du processus de développement
    - Par exemple: Les diagrammes de séquence peuvent être utilisés pour décrire les interactions entre le système et les acteurs lors de la phase de modélisation de cas d'utilisation , pour décrire les interactions entre objets lors de la phase de conception et capturer des traces d'exécution lors de la phase de test.
- Définir un ensemble de règles et de directives pour développer des éléments de modèle aux différentes phases de développement

PERSONNALISER / CUSTOMISER UML

- UML fournit un ensemble de concepts, notations et diagrammes pouvant être utilisés dans un large éventail de contextes de développement.
- Cependant, seul un sous-ensemble de ces éléments est requis dans tout contexte de développement
- Le mécanisme de profil UML peut être utilisé pour personnaliser le langage UML pour des domaines de développement spécifiques.
- Mais la plupart des outils UML n'ont pas réussi à fournir un support de premier ordre pour la personnalisation et la modélisation spécifique à un domaine.

ÉTAPES DE MODÉLISATION

Data Modeling
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial Business Modeling
Initial Requirements
Initial Analysis & Design
Initial Implementation
Initial Test
Initial Deployment
Initial Configuration & Change Mgmt
Initial Project Management
Initial Environment
Elaboration
Construction
Transition
Inception
Iterations
Time
Initial Iteration
Iterations
Iterations
Iterations

EXCEPTIONS: GRANDES LIGNES

- Patrons
  - « Donner le nom du problème et non celui de l'objet qui lance l'exception »
  - « Convertir exceptions »
  - « Journal centralisé d'erreurs »
  - « Dialogue d'erreur »

CONVERTIR EXCEPTIONS

- Comment présenter l'exception?
- Niveau d'abstraction
  - équivalent : faute -> erreur -> défaillance (faute)
- Exception de haut niveau
  - encapsule exception de bas niveau

JOURNAL CENTRALISÉ D'ERREURS

- Objet singleton global
  - lui acheminer toutes les exceptions
  - flexibilité des définitions de flux de sortie
- `java.util.Logging` (depuis JDK 1.4)

DIALOGUE D'ERREUR

- Objet singleton
  - mécanisme centralisé de notification des erreurs
  - pour notifier l'utilisateur
  - protection contre les variations dans le mécanisme de sortie
  - pas un objet de l'IU
  - indépendant de l'application

PATRON: DIALOGUE D'ERREUR

Client
Client
Client

première réponse de moitié avec un ROI positif sur la productivité de développeurs

réf.: https://understandingacycode.com/blog/5-arguments-to-make-managers-care-about-technical-debt/

DETTES TECHNIQUE: TESTS

- Essentiellement, il est dans votre intérêt, en tant que développeur, en tant qu'équipe, d'encourager de nombreux tests plus bas dans les piles présentées ici. Cela commence par des tests unitaires complets et vérifiés. Cela continue avec une discipline de test solide et reproductible (pour laquelle je recommande l'automatisation des tests).
- Pourquoi? Parce que - lorsque vous vous déplacez plus haut dans la pile, ce fichu bug que quelqu'un a enregistré est caché derrière couche après couche de code. Plus le bogue est éloigné du niveau d'unité, plus les composants et les variables d'environnement sont impliqués. Plus ceux-ci sont impliqués, plus il est difficile d'identifier et de réparer, et plus le coût est élevé.

http://jessenoller.com/blog/2008/09/17/the-cost-of-not-testing-software

DETTES TECHNIQUE: TESTS

DETTES TECHNIQUE: TESTS

A 75$/HEURE

	Codeg/Unit Testing	Integration	Sys/e Testing	Stys/Release
Hours to fix	0.2	8.7	12.2	24.8
Cost to fix ($)	240	726	915	1,116

TEST MANUEL VS TEST AUTOMATISÉ

https://abstracta.us/blog/test-automation/true-roi-test-automation/

AVANTAGES

- Un testeur manuel exécute des tests 8 heures par jour et rentre chez lui. À ce moment, les tests s'arrêtent. Avec l'automatisation des tests, nous pouvons exécuter des tests 16 heures de plus par jour (dans le meilleur des cas... bien sûr) pour le même coût, ce qui réduit le coût moyen des heures de test.

Manual	Automated
Hours: 8	Hours: 24
Hourly Rate: 1,250 Euro	Hourly Rate: 1,250 Euro
Cost: 1 Manual	Cost: 0.5 Manual

- Coût total : 5012505/mois
- 3 développeurs x 21 jours/mois * 16h/jour de test * 5 fois plus rapide = 5040h equivalent
- 7 développeurs x 135h/mois = 945h.

AVANTAGES

- Dans ce scénario, nous avons considérablement réduit le coût de chaque heure de test de 78 à1 7, 54, ce qui est un avantage que le directeur financier comprendra clairement. Ou vous pouvez le voir de cette façon; nous avons augmenté les tests de 1 350 heures à 5 985 heures équivalentes et gagné 315 000 $ de tests par mois pour le même coût (5 040 fois le coût horaire moyen d'un testeur).

VALEUR COMMERCIALE

- Améliorez la qualité des logiciels
- Évitez les problèmes de fonctionnement
- Maintenir une bonne image client
- Évitez les problèmes juridiques
- Diminuer le coût de la correction des bogues de 5 fois

VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel

VALEUR INFORMATIQUE

- Simplifie les tâches de routine
- Exécutez plus de tests sans augmenter les coûts dans le même temps
- Augmentez l'étendue de la couverture
- Trouvez les défauts difficiles à détecter plus tôt, lorsqu'ils sont plus faciles à corriger
- Améliorez la qualité du logiciel

Êtes-vous d'accord pour dire que le retour sur investissement de l'automatisation des tests est important?

VISUAL STUDIO CODE

- voir plugin Tech Debt Metrics

ENQUÊTE SUR LA RÉMUNÉRATION D'INGÉNIEUR

- https://www.genium360.ca/fr-ca/services-aux-entreprises/enquete-sur-la-remuneration/
- https://drive.google.com/file/d/1UnlnAz3y1it-ISS_ED2W-dj2JVJ8cITC/view?usp=sharing

RÉSAUTAGE

https://drive.google.com/open?id=1Cm2WTaG5KN0sqonAfOnbxjcuNfbMYFBW

DEVELOPER SURVEY RESULTS

2019

https://insights.stackoverflow.com/survey/2019

LISTE DE RECRUTEURS

https://drive.google.com/open?id=11XYE6k8m6YIcbAJ6hGBfkmIIJLH_DRTp

https://drive.google.com/open?id=1jJT5YChfylxJNfSQQPH1IsBLab2CQYpG

97 THINGS

97-THINGS-EVERY-PROGRAMMER-SHOULD-KNOW

- https://github.com/97-things/97-things-every-programmer-should-know/blob/master/en/SUMMARY.md

97-THINGS-EVERY-AGILE-DEVELOPER-SHOULD-KNOW

- https://github.com/97-things/97-things-every-agile-developer-should-know