IFT2255 - Génie logiciel

Introduction aux changements, analyse des besoins

Bruno Dufour
dufour@iro.umontreal.ca

Changements et logiciels

- Le changement d'un logiciel consiste à modifier son code pour ajouter des fonctionnalités
- Plusieurs caractéristiques :
  - Type (but)
  - Impact
  - Portée
  - Stratégie
  - Type de code

Types de changements

- **Perfectifs**: visent à accroître la valeur d'un logiciel en ajoutant des fonctionnalités
- **Adaptatifs**: visent à préserver la valeur du logiciel en l'adaptant à des changements d'environnement
  - ex: adapter à un nouveau système d'exploitation
- **Correctifs**: visent à corriger les erreurs existantes (bogues)
- **Préventifs**: visent à faciliter les changements futurs maintenance à venir
  - Invisibles pour l'utilisateur
  - ex: amélioration de la structure

Impact sur la fonctionnalité

- **Incrémental** : ajoute une nouvelle fonctionnalité
- **Contraction** : retire une fonctionnalité obsolète d'un logiciel
  - Permet de réduire la complexité du logiciel
- **Remplacement** : remplace une fonctionnalité existante par une autre
  - ex: correction de bogue, amélioration de la performance
- **Réusinage** : substitue une structure par une autre sans modifier le comportement observable
  - permet de limiter la dégradation d'un logiciel due aux changements

Portée de l'impact

- **Impact localisé** : affecte au plus quelques modules étroitement liés (le plus fréquent)
  - facile à réaliser, impact facile à prévoir
  - souvent une conséquence de l'organisation du code (ex: classes faiblement couplées)
- **Impact significatif** : affecte un nombre élevé de modules (commun)
  - impact plus difficile à prévoir, requiert des outils et techniques plus avancés
- **Impact massif** : affecte la majorité des modules (rare)
  - Coûteux et à haut risque
- Un changement affecte aussi d'autres artéfacts (ex: documentation, cahier des charges, etc.)

Stratégie adoptée

- **"Quick fix"** : changement d'urgence
  - Par exemple, lorsque la vie d'être humains en dépend
  - Aussi utilisé lorsque le logiciel n'est plus activement maintenu
- **Changement durable** : toutes les autres situations
  - Le coût d'un changement rapide est plus élevé à long terme que le bénéfice qu'on en retire
  - Il est préférable bien exécuter un changement, de façon professionnelle et réfléchie

Type de code visé

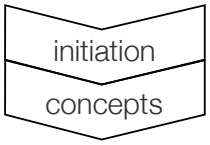
- **Code source** : le plus courant.
  - Le logiciel est souvent compilé à nouveau et déployé pour compléter les changements
- **Code compilé** : rarement utilisé, mais utile lorsque
  - le code source n'est pas disponible
  - la situation est urgente et la compilation et/ou le déploiement sont trop longs
  - le compilateur produit du code incorrect ou inutilisable

Phases du processus de changement

- Initiation
  - Débute habituellement par une requête de changement
  - Les requêtes sont souvent priorisées

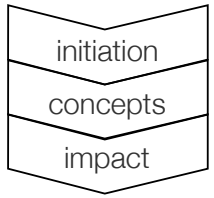
Phases du processus de changement

- Localisation des concepts
  - Les concepts sont extraits de la requête de changement
  - Ces concepts sont repérés dans le code et deviennent le point de départ pour les changements à effectuer



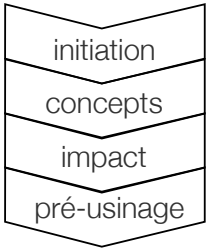
Phases du processus de changement

- Analyse de l'impact
  - Détermine la stratégie à utiliser et l'impact des changements à apporter
  - Les classes identifiées lors de la localisation des concepts forment l'**ensemble d'impact initial**
  - Les dépendances sont analysées et d'autres classes sont ajoutées à l'ensemble d'impact



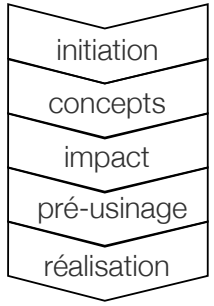
Phases du processus de changement

- Pré-usinage
  - Certains changements sont apportés afin de minimiser l'impact des changements
    - ex: extraire une classe parente
  - Permet souvent de contenir le changement à effectuer ou d'en réduire la portée



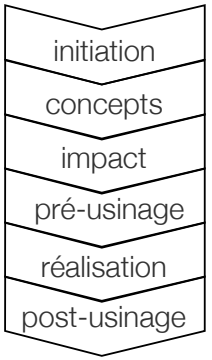
Phases du processus de changement

- Réalisation
  - Ajout de nouveau code qui interagit avec le code existant
  - L'impact de l'ajout ou de modifications se répercute sur les classes voisines
    - Propagation des changements
    - Ondulation (*ripple effect*)



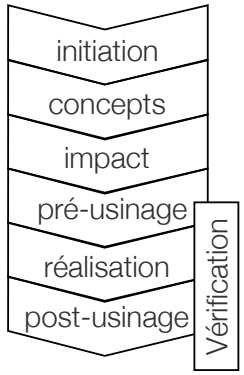
Phases du processus de changement

- Post-usinage
  - Permet d'éliminer ou de retarder la dégradation introduite par les changements
  - ex: trop de responsabilités pour une méthode ou une classe suite à des ajouts



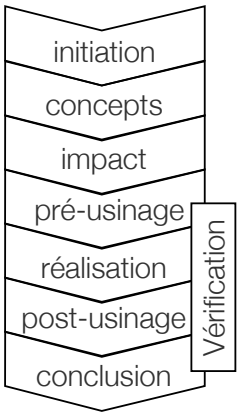
Phases du processus de changement

- Vérification
  - S'assure que le changement est correct et bien effectué
  - Tests automatisés
    - unitaires
    - fonctionnels
    - structurels
    - de régression
    - etc.
  - Revues de code



Phases du processus de changement

- Nouvelle révision dans le SCR
- Déploiement d'une nouvelle version?
- Préparation au prochain changement



Initiation des changements & besoins

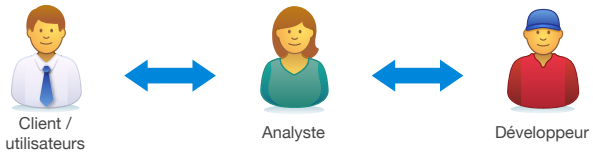
Initiation des changements

- Nouveaux besoins
  - besoins = exigences = requirements
- Par exemple
  - Un bogue est rapporté par un utilisateur
  - Un utilisateur demande une nouvelle fonctionnalité
  - Un programmeur propose une amélioration
  - Un gestionnaire désire rendre un produit concurrentiel

Identification des besoins

- Identifier les besoins correctement est difficile!
- Causes les plus répandues de l'échec de projets:
  - Contribution insuffisante du client
  - Spécification incomplète des besoins
  - Modification des besoins
- Difficultés:
  - Problèmes complexes, connaissance limitée du domaine, clients sans connaissances techniques, communication intensive

Le rôle de l'analyste



Client / utilisateurs

- Expertise, jargon du domaine
- Indécis, opinion changeant selon l'offre
- Besoins ambigus, éléments manquants

Analyste

- Doit devenir aussi informé du fonctionnement de l'entreprise que les utilisateurs
- Doit devenir l'expert

Développeur

- Schémas
- Langages formels
- Spécifications souvent incompréhensibles pour les non initiés.

Élicitation des besoins

- Collecte des informations
  - Identification des opérations et procédés administratifs
    - "Que faites-vous ?"
  - Réalisation des opérations
    - "Comment le faites-vous ?"
  - Identification des informations requises pour réaliser les opérations
    - "Quelles informations utilisez-vous ?"

Collecte des informations

- Méthodes traditionelles
  - Entrevue avec clients, utilisateurs et experts du domaine
    - Entrevue formelle: préparée, agenda, questions ouvertes/fermées
    - Entrevue informelles: encourager le client à parler
  - Questionnaires
    - accompagnent ou préparent l'entrevue
    - Type de questions (fermées de préférence)
      - Question à choix multiple
      - Question avec "cote" de la réponse (ex. d'accord, pas d'accord, etc.)
      - Question avec "ordonnancement" des réponses

Collecte des informations

- Observation
  - passive (on regarde seulement) ou active (on participe aux activités)
  - Documenter l'observation (ex: diagramme de flux des travaux)
- Étude des documents et des systèmes logiciels existants
  - Documents d'entreprise: formulaires, procédures de travail, descriptions d'emploi, plans, correspondance, comptes, minutes des réunion, etc.
  - Formulaires et rapports: rapports techniques, saisie d'écran, manuels utilisateurs, modèles d'analyse et de conception, etc.
  - journaux d'affaires, livres de gestion
- Étude des solutions (déjà existantes) des fournisseurs

Collecte des informations

- Méthodes actuelles
  - Prototypage
    - Maquette démonstrative, première étude de faisabilité
    - Identification des besoins conflictuels, omis ou mal saisis
    - Prototype jetable:
      - Attention portée sur les besoins les moins bien compris
    - Prototype évolutif
      - Attention portée sur les besoins les mieux compris

Collecte des informations

- Développement conjoint d'applications (*joint application development - JAD*)
  - Série d'ateliers (*workshops*) ou réunions de travail auxquelles participent clients et développeurs
  - Accélère les communications / feedback rapide
  - Réduit les risques qu'un des participants se manifeste avec de nouvelles exigences

Types de besoins

- Les besoins peuvent traduire des exigences concernant:
  - Les fonctionnalités
  - L'environnement physique
  - Les interfaces
  - Les humains (utilisateurs)
  - La documentation
  - Les données
  - Les ressources
  - La sécurité
  - L'assurance de la qualité

Classification des besoins

- **Besoins fonctionnels**: comportement, caractéristiques, capacités
  - ex: « Le système lit les fiches des employés et imprime des chèques de paie »
  - Les autres besoins sont non-fonctionnels (*contraintes*)
- **Besoins d'utilisation**: facteurs humains, aide, documentation
  - ex: « La police utilisée pour le texte doit être lisible d'une distance de 2 mètres »
  - ex: « Ne pas utiliser les couleurs associées au formes communes de daltonisme »

Classification des besoins

- **Besoins de fiabilité**: fréquence des défaillances, récupération
  - ex: « Si une défaillance se produit dans le système, … »
- **Besoins de performance**:
  - ex: « Le temps de réponse du système est inférieur à 1 seconde pour 90% des accès. »
  - ex: « Le système doit être en mesure de traiter 1Mo de données transactionnelles par seconde. »

Classification des besoins

- **Besoins de support**: adaptation, internationalisation, maintenance
  - ex: « Le système devra permettre des changements fréquents dans la topologie du réseau. »
  - ex: « Le système devra pouvoir incorporer plusieurs composants externes (inconnus) pour le calcul des impôts. »
- **Besoins de d'implémentation**:
  - ex: « Le système doit utiliser Linux et Java »
  - Souvent dû à des contraintes budgétaires ou à la disponibilité du personnel qualifié

Caractéristiques des besoins

- **Corrects**: ne contiennent pas d'erreurs
- **Clairs, sans ambiguïtés, intelligibles**: compréhensibles par les programmeurs sans avoir à supposer ou deviner
- **Cohérents**: non-conflictuels, possibles à satisfaire en même temps
- **Réalistes**: respectent ce qui peut être accompli avec la technologie courante
- **Pertinents**: nécessaires et non-redondants
  - les besoins superflus devraient être éliminés

Caractéristiques des besoins

- **Vérifiables**: facilement mesurables / quantifiables de façon à déterminer si l'implémentation atteint l'objectif visé
  - ex: immédiatement, rapidement → en moins de 1s
  - facilite l'écriture de tests automatisés

Validation et négociation

- Les besoins répondent-ils aux exigences du client ?
- Réviser la liste des besoins
  - Sont-ils cohérents? Clairs? Intelligibles? ...
  - Élimination des besoins non pertinents ou irréalistes
    - Bien définir les frontières du système
    - Identifier les besoins qui ne répondent pas aux objectifs du système, qui sont hors plan, etc.
  - Faire la liste des besoins exclus pour cause de
    - trop grande difficulté de réalisation
    - mise en oeuvre par matériel (*hardware*)
    - technologie existante inadéquate
    - etc.
- Tout compromis doit être négocié avec le client

Priorisation des besoins

- Les besoins sont habituellement enregistrés sous forme de liste priorisée (*backlog*)
  - Les développeurs choisissent le prochain changement à effectuer en fonction de cette liste
  - La liste peut être le résultat d'une phase d'analyse dédiée (*modèle en cascade*) ou élaborée progressivement (*modèles agiles*)
- Plusieurs critères de priorisation
  - Sévérité
  - Valeur
  - Risque
  - Besoins

Priorisation par sévérité

1. Erreur fatale
2. L'application est sévèrement compromise
   - Impossible de contourner l'erreur
3. Certaines fonctionnalités sont compromises
   - Possible de contourner l'erreur
4. Problème mineur
   - N'implique pas de fonctionnalité critique

Priorisation par valeur d'affaires

1. Fonctionnalité essentielle
   - L'application est inutile
2. Fonctionnalité nécessaire
   - Les utilisateurs comptent sur cette fonctionnalité
3. Fonctionnalité importante
   - L'application est utile même sans cette fonctionnalité
4. Amélioration (mineure)

Priorisation par risque

1. Risque sévère
   - Le projet risque d'échouer si non-résolu
2. Risque important
   - Ne peut être ignoré
3. Risque éloigné
   - Moins sévère, mais requiert l'attention de l'équipe
4. Obstacle mineur

Priorisation par besoins

1. Besoin clé
   - Si implémenté trop tard, la majorité du code devra être réécrit
2. Besoin important
   - Si repoussé, peut forcer à réécrire beaucoup de code
3. Réécriture non négligeable
4. Réécriture mineure / limitée

Processus d'initiation de changement

- Sélectionner un ensemble de besoins avec une priorité élevée
- Analyser ces besoins
- Le besoin avec la priorité la plus élevée après l'analyse constitue la prochaine requête de changement

Cas d'utilisation

Cas d'utilisation

- Technique permettant d'identifier et de décrire les fonctionnalités d'un logiciel qui sont significatives pour ses utilisateurs
  - Permet de décrire les interactions du logiciel avec son environnement
  - Expression du comportement du logiciel (actions et réactions) selon le point de vue des utilisateurs
  - Détermination des besoins fonctionnels des utilisateurs cibles

Acteur

- Rôle joué par une personne ou un logiciel
  - Représentation idéalisée d'une personne, d'un logiciel, d'un processus, d'une organisation qui interagit (depuis l'extérieur) avec le logiciel
  - Une même personne peut correspondre à plusieurs acteurs
    - Un acteur est différent d'un autre si ses besoins, ses attentes de service envers le logiciel sont différents
  - Un même acteur peut être joué par plusieurs entités
- L'acteur est toujours un élément **externe** au logiciel

Acteur (suite)

- L'acteur peut consulter ou modifier l'état du logiciel
  - interaction avec le cas d'utilisation par envoi de message
- En réponse à l'action d'un acteur, le logiciel fournit un service
- Catégories d'acteurs
  - **Acteurs principaux** : participant externe interagissant avec le logiciel dans le cadre du cas d'utilisation et bénéficiaire direct du service offert
  - **Acteurs de support** : participant offrant un service qui contribue à la réalisation du cas d'utilisation
- Le temps (horloge, alarme) peut parfois être considéré comme un acteur

Scénario

- Une séquence spécifique d'interactions entre les acteurs et le système décrit
  - Exécution particulière d'un cas d'utilisation
  - Peut correspondre à un succès ou à un échec
- Description fonctionnelle (*black-box*) et non pas structurelle (*white-box*)
  - Spécifier ce que le système doit accomplir sans décider comment il le fera
  - On se concentre sur les préoccupations réelles des utilisateurs: pas de solutions d'implémentation, pas d'inventaire fonctionnel

Cas d'utilisation

- Un ensemble de scénarios de succès ou d'échecs connexes
- Comporte
  - Un scénario principal
  - Des scénarios alternatifs (0+, optionnel)

Documentation des cas d'utilisation

- Description textuelle compréhensible par du personnel non-technique (ex: client)

1. Nom du cas d'utilisation
2. Acteurs
3. Préconditions (satisfaites au départ)
4. Postconditions (satisfaites si le scénario se termine par un succès)
5. Scénario principal
   - Succès
   - Description du cours normal des choses, sans branchements
6. Scénarios alternatifs
   - branchements du déroulement principal

Exemple - Guichet automatique

- UC1. Retirer de l'argent au distributeur
- Acteur principal : client
- Pré-conditions
  - Le distributeur contient des billets
  - Il est en attente d'une opération
  - Il n'est ni en panne, ni en maintenance
  - Il est prêt à recevoir une carte bancaire
- Post-conditions
  - La carte bancaire est sortie du distributeur
  - La somme d'argent sur le compte est égale à la somme d'argent qu'il y avait avant, moins le montant du retrait

Exemple - Guichet automatique

- Scénario principal
  - Le client introduit sa carte bancaire
  - Le logiciel lit la carte et vérifie que la carte est valide
  - Le logiciel demande au client de taper son code
  - Le client tape son code confidentiel
  - Le logiciel vérifie que le code correspond à la carte
  - Le client choisi une opération de retrait
  - Le logiciel demande le montant à retirer
  - etc.

Exemple - Guichet automatique

- Scénarios alternatifs
  - 2. (A) Carte invalide : au cours de l'étape (2) si la carte est jugée invalide, le logiciel affiche le message d 'erreur « carte invalide », rejette la carte et le cas d'utilisation est terminé
  - 6. (B) Code erroné : au cours de l'étape (5), si le code entré n'est pas valide, le logiciel affiche le message d'erreur « code erroné », rejette la carte et le cas d 'utilisation est terminé

Exemple - Magnétoscope numérique

UC1 : Enregistrement différé d'une émission en utilisant le guide de programmation
Acteurs : Utilisateur
Préconditions : Le magnétoscope est sous tension et fonctionne correctement.
Postconditions : La minuterie est réglée pour le programme sélectionné.
Scénario principal :
1. L'utilisateur appuie sur la touche 'guide'.
2. Le système vérifie que l'information du guide de programmation est à jour.
3. Le système affiche le guide de programmation.
4. L'utilisateur navigue et sélectionne un programme dans le menu à l'aide des touches '→', '←', '↑' et '↓', puis appuie sur la touche 'enregistrement'.
5. Le système vérifie que le disque dur n'est pas plein.
6. Le système vérifie que le programme n'a pas encore débuté.
7. Le système vérifie que le programme sélectionné n'est pas en conflit avec les autres enregistrements différés.
8. Le système règle la minuterie pour le programme sélectionné et affiche une boite de dialogue qui confirme le réglage de l'enregistrement différé.
9. L'utilisateur appuie sur la touche 'OK'.
10. Le système retourne à l'affiche du guide de programmation.

Scénarios alternatifs :
5. (A) Disque dur plein : le système affiche un message indiquant à l'utilisateur que le disque est plein. L'utilisateur appuie sur la touche 'OK'. Le système retourne à l'affiche du guide de programmation et le cas d'utilisation est terminé.

7. (B) Conflit détecté : le système affiche un message d'erreur indiquant le conflit. L'utilisateur appuie sur la touche 'OK'. Le système retourne à l'affiche du guide de programmation et le cas d'utilisation est terminé.

Définition des événements

- Événement: occurrence qui survient à un moment et un endroit précis
  - que l'on peut décrire
  - dont le système doit se rappeler (pertinente)
- Quels sont les événements qui engendrent
  - une réaction
  - une activité, ou
  - un traitement?

Types d'événements

- Événements externes
  - Survient à l'extérieur du système et est habituellement initié par un agent ou un acteur externe.
  - ex: un client passe une commande, un client actualise les informations de son compte.
- Événements temporels
  - Résultent de l'atteinte d'un point dans le temps
  - ex: moment de produire les chèques de paie (à chaque deux semaines), moment de produire les factures mensuelles (le 28 de chaque mois).
- Événements d'état
  - Se produit lorsque quelque chose survenant dans le système déclenche un besoin de traitement.
  - ex: rupture de stock, déclenchement de l'alarme de feu.