Ingénierie du Logiciel
Master 1 Informatique – 4I502

Cours 3 : Analyse
Fiches Détaillées de Cas d'Utilisation

Yann Thierry-Mieg
Yann.Thierry-Mieg@lip6.fr

Spécifier le comportement

- Objectifs :
  - Préciser finement les interactions
  - *Faire des choix* concernant les modèles d'interaction
  - Construire un document *précis* et *exhaustif* pour préparer la conception
- Moyens :
  - A gros grain, diagramme de use case
  - Fiches détaillées pour *spécifier* les échanges
  - Maquettes, workflows, peuvent venir compléter la descritption

La fiche détaillée

- Un cas d'utilisation = une fiche détaillée
- Langage naturel structuré :
  - Rubriques pouvant varier selon l'entreprise
  - Cadre contraint, rubriques précises, aident à être exhaustif/systématique
  - Présentation homogène quel soit le domaine métier
  - Fiches souvent utilisées en traçabilité des exigences
- Le sens de chaque rubrique et la façon de rédiger les textes a beaucoup d'importance
  - Certes, c'est verbeux.
- On souhaite capturer l'ensemble des scenarios (nominaux, alternatifs, d'exception) que représente un cas d'utilisation.

Rubriques de la FD

- Référence unique pour pouvoir facilement citer : UC01
- Date, auteurs, changelog…
  - Tout ce qui est nécessaire pour tracer l'évolution de cette fiche
- Titre
  - Cohérent avec le cas d'utilisation
- Description
  - En une à deux lignes, sens métier/finalité globale du cas d'utilisation pour l'acteur.
- Acteurs:
  - Les acteurs concernés, qui ont accès à cette fonctionnalité
- Hypothèses:
  - Contraintes sur l'environnement mais non-contrôlables par le système.
  - Rarement utile d'en formuler
- Exigences non fonctionnelle : volume, débit, disponibilité…

En examen : Titre, Acteurs suffisent.

FD : les pré et post-conditions

- Préconditions :
  - Une expression booléenne sur l'état actuel du système quand l'interaction démarre
    - Testable sur les classes métier
    - Bouton « grisé » si la condition est fausse
  - ✓ Le système contrôle les pré-conditions, on ne les reteste pas dans le scenario

- Post-conditions :
  - Expressions booléennes sur les données, après une exécution *réussie* du cas d'utilisation
  - Exprimable avec le diagramme de classes métier
  - Reflète l'effet des mise à jour de données ou enregistrements réalisés pendant le use case
  - Il peut être pertinent de bien préciser *quelles* occurrences de données sont modifiées (e.g. celles liés à *ce* client qui vient de faire l'action).

Scenario Nominal : « Tout va bien »

- Scenario *nominal*
  - En faisant abstraction de tout ce qui pourrait se passer mal, ou autrement, quelle est la grande ligne principale de cette interaction avec le système ?
  - Focaliser sur ce scenario = productif
    - Si déjà ça marche, on a quelque chose d'opérationnel
- Un scenario d'interaction :
  - Linéaire, pas de branchement, une série d'étapes consecutives
  - Chaque étape est une action soit d'un acteur, soit du système
  - Les étapes sont numérotées SN1, SN2 … A1.1, A1.2, … et cross-referencés assez intensivement

Les étapes du scenario

- Actions du système : « Le système …»
  - affiche … : préciser ce que le système affiche
    - À la fois les données visibles (cf classes métier)
    - Leur agencement éventuel (« screenshot »)
    - Les boutons ou contrôles
  - calcule … : expliquer précisément ce qui est calculé et si possible comment
    - S'appuyer sur le diagramme de classes métier pour expliquer les données utiles
    - S'appuyer sur le cahier des charges pour la definition en langage naturel
    - Aider le concepteur !
  - Vérifie que… [TEST BOOL]:
    - Bien expliquer ce qui est testé ! Appui sur classes métier.
    - Point de branchement naturel pour les alternatives/exceptions.
  - Enregistre … Met à jour …
    - Doit fonctionner sur classes métier

Etapes : actions des acteurs

- Trigger :
  - Première étape du scenario nominal SN1
  - A l'initiative de l'acteur : déclencheur de l'interaction
- Autres étapes : "L'acteur …"
  - Saisit des données, sélectionne des cases…
    - On ne détaille pas chaque clic ! Pensez granularité "bouton OK".
    - Les écrans d'IHM aident à se faire une idée
  - ✓ C'est tout ! Tout ce que fait l'acteur à part nous donner de l'information ou nous orienter ne nous intéresse pas !
- Parfois une reference à un cas d'utilisation comme étape
  - SN4 : executer UC02
  - Correspond à un <<include>> sur le diagramme

Scenario Alternatif : « Tout finit bien »

- Scenario Alternatif:
  - Spécifié comme le SN : étapes linéaires
  - Aboutit à l'objectif : les post-conditions seront validées
  - Alternatif = branchement
    - SN3 : Le système vérifie que XX
      - => ALT A1.1. En SN3, si XX n'est pas vrai …
    - SN5 : L'utilisateur valide le choix XX
      - => ALT A2.1 En SN5, si l'utilisateur choisit plutôt YY…
  - ✓S'achève le plus souvent sur un débranchement :
    - Reprendre l'interaction en SN2

Scenario Alternatif : « Tout finit bien »

Peut au besoin être modélisé par un use case :

- ✓ ALT A3.2 : executer les UC03
- ✓ Correspond à un <<extend>> sur le diagramme
- ✓ /!\ granularité, UC03 doit lui même avoir e.g. des alternatives pour justifier ce choix

Permet de modéliser une boucle

- ✓ SN 8 : Le système vérifie qu'il n'y a plus d'article
- ✓ ALT A4.1 : en SN8, s'il reste des articles, retour en SN4

Scenario Exception « Ca se passe mal »

- Scenario d'exception
  - Scenario, étapes idem SN et ALT
  - L'action du use case ne va pas au bout : post-conditions non-validées
  - Branchement
    - Contrôle du système qui échoue sans recuperation possible
      - Echec de validation des conditions métier du CdC
    - Annulation utilisateur
  - ✓ Un problème ou une erreur, mais *prévue !*

Fiches Détaillées : Factorisation

- Exemple : Editer Liste étudiants
  - Ajouter, Modifier, Supprimer des étudiants
  - Modélisation avec des ALT :
    - SN :
      - 1. Le système affiche la liste actuelle
      - 2. L'acteur choisit créer un étu,
      - 3. Le système affiche un formulaire étudiant :….
      - 4. L'acteur renseigne les champs, valide
      - 5. Le système enregistre la modification
    - ALT : 1. En SN2, L'acteur choisit un étudiant dans la liste. 2. retour en SN3 avec les données de l'étudiant chargées
    - ALT 2 : 1. En SN4, l'acteur choisit de détruire l'étudiant 2. valide 3. retour en SN5
    - +EX…

Fiche Détaillées bilan

- Cohérence avec les diagrammes
  - De use case : Explique et affine ce dernier
  - De CM : pré, post, contrôles, mise à jour *réalisables*
- Précision dans l'expression des étapes
  - Raffine le CdC, on pourra s'en débarrasser
  - Pose des choix, deux équipes auront rarement les même fiches en partant du même CdC
- Rubriques Essentielles :
  - ID, Titre, Acteurs, Préconditions, SN, postconditions, ALT, EX.
  - Linéarité des scenarios, contraintes sur les rubriques => effort de rédaction

Le besoin : autres formes

Maquettes d'IHM

- Maquette
  - Élément jetable mais précis vis-à-vis de l'objectif, aidant à le visualiser et le présenter en particulier au client
  - Peu élaboré, rapide à construire
  - Différent d'un *prototype* qui lui sera progressivement raffiné pour participer à la solution
- Ebauches d'IHM :
  - Ecran principal/d'accueil : accès aux use case
  - Chaque use case => un ou plusieurs ecrans en général, + les transitions entre ces ecrans
  - Les boutons sur ces ecrans => souvent des ALT ou EX
- Des outillages peuvent aider, mais simple esquisse papier OK aussi.

User Story

- Utilisé dans des méthodes plus agiles
  - Beaucoup plus succinct, mais joue le même rôle que nos FD
- « I as XXX want YYY [so that ZZZ] »
  - Quelques lignes MAX
- Permet le découpage des tâches à réaliser
  - Plus proche d'un scenario de FD qu'un ensemble de scenarios
- Moins rigoureux qu'une FD, privilégie les interactions clients à l'oral, le role play
- Reste une description informelle d'un feature plutôt qu'une spécification du besoin.

Développement tiré par le besoin

- Idée de priorisation :
  - Pricing des tâches en difficulté de développement
    - E.g. Fibonacci
      - 1, 2, 3, 5, 8, 13, 21.
    - Redécoupage si pricing/difficulté trop difficile à évaluer
    - Suivi de l'avancement avec Kanban, burndown charts…
  - Evaluation client de la criticité/importance
    - Il doit avoir des préférences
  - ✓ On commence par les tâches critiques les plus faciles
  - ✓ Le développement est transverse à la structure du code
    - Un feature nécessite : un bouton, un écran, un contrôleur, l'accès aux données…

Exemple

Exemple : StoneHearth

UC01 : Mémoriser un Deck

Acteur : Joueur

Précondition : être logé sur son compte, sur l'écran d'accueil (pas de partie en cours)

Scenario Nominal :

1. Le joueur choisit de créer un deck
2. Le système affiche les emplacements, en indiquant les emplacements libres
3. Le joueur sélectionne un emplacement
4. Le système affiche toutes les cartes du joueur, et la liste des cartes du deck (donc vide si on a choisi un emplacement vide).
5. Le joueur sélectionne une carte à ajouter au deck parmi la collection
6. Le système met à jour le contenu du deck et l'affichage : les cartes déjà dans le deck ne sont pas sélectionnables dans la collection.
7. L'utilisateur choisit d'enregistrer le deck
8. Le système invite à saisir un nom pour le deck
9. L'utilisateur saisit le nom du deck et valide
10. Le système mémorise le deck

Post-condition : le deck nouvellement créé est disponible pour jouer.

Exemple : StoneHearth

A1 : ajout autre cartes

A1.1. En SN7, l'utilisateur peut au contraire continuer tant que le deck comporte moins de 30 cartes, retour en SN5

A2 : enlever une carte

A2.1 En SN5, l'utilisateur peut sélectionner une carte du deck en cours de construction,

A2.2 Le système retire la carte sélectionnée du deck.

A2.3 Retour en SN6.

A3 : emplacement occupé

A3.1 En SN3, si l'utilisateur choisit un emplacement non vide, le système initialise l'affichage avec le contenu actuel du deck

A3.2 retour en SN4.

E1 : annulation

E1.1 En SN7, l'utilisateur choisit d'annuler, ses modifications en cours ne sont pas sauvegardées, retour à l'écran d'accueil.