Fusion une méthode orientée objet

Fusion

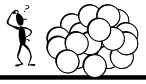
Une méthode de développement orientée objet

Fusion

- Propose
  - Une démarche de développement
    - » phases
    - » critères de fin de phase
  - Des outils de gestion
    - » les concepts (ontologies)
    - » les sorties de chaque phase sont définies
    - » vérifications croisée entre les phases
    - » des composants réutilisables
    - » un référentiel centralise la mémoire de la méthode
  - Des notations … devenues caduques à cause d'UML !
- Fusion peu s'adapter à la taille d'un projet

Démarche

- Analyse
  - décrit le comportement désiré du système par un modèle
    - » Classes, Objets
    - » Relations
    - » Ce qu'ils font
    - » Dans quel ordre
- Conception
- Réalisation



Démarche

- Analyse
- Conception
  - Choix de la mise en œuvre
    - » opérations par des objets
    - » relations entre classes (clientes, filles)
    - » attributs, opérations des classes
- Réalisation

Démarche

- Analyse
- Conception
- Réalisation
  - transforme la conception dans le langage cible
    - » héritage, classes
    - » interactions
    - » automates
  - optimisations, réglages

Objectifs de l'analyse

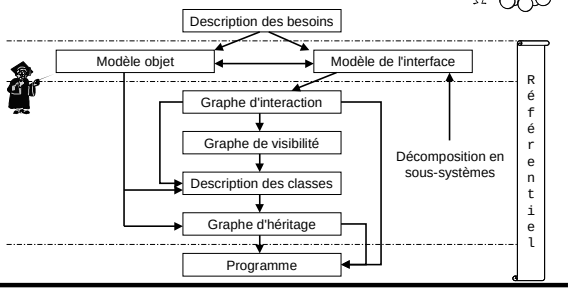
- Comprendre le problème
  - analyse du domaine (domain analysis)
- Délimiter le système
  - définir l'interface, la frontière
- Définir les fonctions du système
  - que font les objets du système

Fusion une méthode orientée objet

Analyse

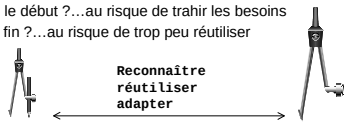
- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

La méthode



Un problème difficile

- On construit rarement un système à partir de rien.
- Comment intégrer dans la démarche la réutilisation ?
  - Dès le début ?…au risque de trahir les besoins
  - A la fin ?…au risque de trop peu réutiliser



- La réponse (?) : savoir-faire, expérience, connaissance des technologies
- Peut-être des éléments avec la notion de "composant"

Le référentiel

Nom	Type	Description
	classe opération invariant	

- Contient les noms et concepts
- Utile lors de la vérification croisée des modèles
- Peu avoir plus de colonnes pour des types spéciaux
- Attention au choix des noms (redondance)
- La description peut faire référence à d'autres noms...

Le modèle objet

Syndrome de la page blanche…il faut démarrer, on cherche :

- Des classes
  - à partir des noms (objets physiques, personnes, organisation, concepts, abstractions, …)
- Des relations
  - à partir des liens entre objets (communication, association physique, contenant, action, …)
- La modélisation est plus efficace en groupe "remue-méninge" qu'individuellement
- On cherche UN modèle, pas LE modèle !

Classe

- Une classe et,
- Deux attributs qui la caractérisent

Professeur
nom discipline

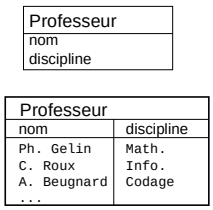
- Pourquoi choisir des attributs ? Plutôt qu'une relation vers une autre classe …
- Valeur - objet ; choix d'analyse

UML : http://www.essaim.univ-mulhouse.fr/uml/documentation/version1.0/notation/notation.html

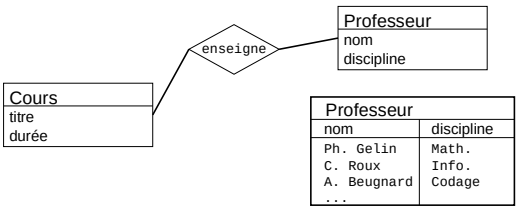
Fusion une méthode orientée objet

Instances

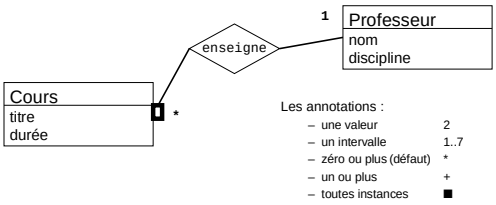
- Un exemplaire de la classe, avec des valeurs attribuées aux attributs
- Plusieurs instances différentes pourraient avoir les mêmes attributs



Classe - Relations



Cardinalité



Les annotations :

- une valeur 2
- un intervalle 1..7
- zéro ou plus (défaut) *
- un ou plus +
- toutes instances ■

Invariant

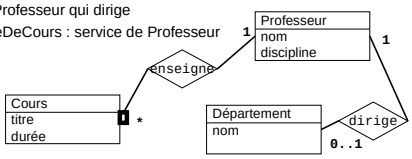
Assertion toujours valide (dans des états stables du système) - voir langage OCL de l'OMG

un chef de département fait moins de cours que les autres professeurs

chef.nombreDeCours < nonChef.nombreDeCours

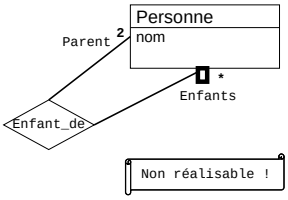
chef : Professeur qui dirige

nombreDeCours : service de Professeur

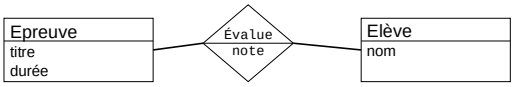


Rôle

- Les éléments d'une même classe peuvent participer à plusieurs relations
- On enlève les ambiguïtés en nommant les arc : le rôle



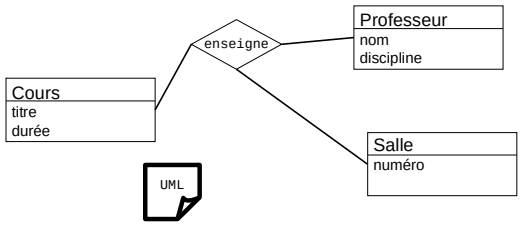
Attributs de relations



- Les relations peuvent être caractérisées par des attributs
- (Ce pourraient être en fait des classes un peu spéciales)
  - La note n'est ni un attribut de l'élève ni de l'épreuve

Fusion une méthode orientée objet

Relations n-aire



Elaboration du modèle objet

- La liste des objets et relation est critiquée :
  - présence de chaque entité analysée justifiée
  - validation par rapport aux besoins (étudiés hors Fusion)
- On place les entités dans dans le référentiel

Le référentiel

Nom	Type	Description
Professeur	classe	Les membres du corps enseignant
nom	attribut	Le nom de famille du Professeur
discipline	attribut	La discipline enseignée par le Professeur
enseigne	relation	Les Professeurs enseignent des Cours
limite_chef	invariant	Un chef de département enseigne moins
Parent	rôle	...

Guides de réflexions

- Utilisez les principes suivants :
  - Généralisation
  - Agrégation
  - Attributs
  - Cardinalité
  - Invariants
  - Relations dérivées
- La modélisation nécessite du savoir-faire, de l'expérience (capitalisation sous forme de Patterns)

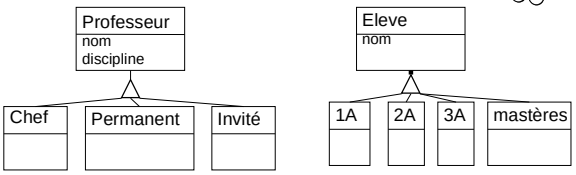
Généralisation

- Recherchez les relations
  - "est-un"
  - "est-une-sorte-de"
- On observe
  - les attributs communs à des classes
  - les relations communes

Généralisation, facteur de réutilisation, on *hérite* des propriétés (attributs, relations)

Attention, des généralisations trop précoces peuvent nuire au modèle (classes inutiles, liens forcés)

Généralisation : Partition

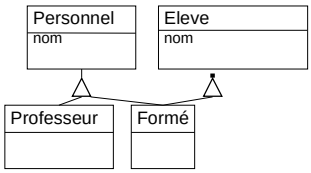


Pas de partitionnement
- Professeur peut posséder des instances
- Les sous-classes peuvent se recouvrir

Partitionnement
- Cours n'a pas d'instances
- Les sous-classes sont disjointes

Fusion une méthode orientée objet

Généralisation multiple



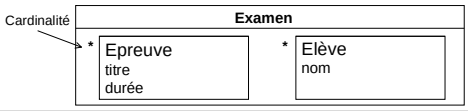
- Le nom de Formé est ambiguë, on la résout :
  - nom@Personnel
  - nom@Eleve

Agrégation

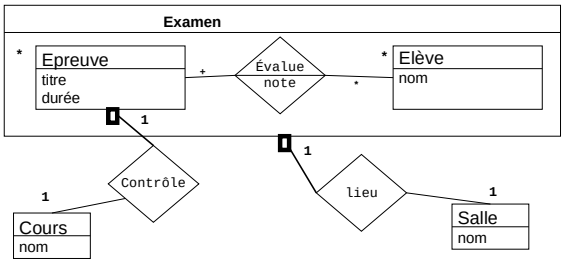
- Recherchez les relations
  - "possède-un"
  - "contient-un"
  - "participe-à"

Agrégation, structure le modèle

C'est une relation spéciale...



Agrégat : relation



Attributs

- Il est préférable de les ignorer dans une première phase
- Puis
  - Les classes sans attributs et reliées à une seule autre classe peuvent devenir des attributs
  - des attributs présents à de nombreux endroits peuvent devenir des classes

Cardinalité

- Mise en place lors de la définition des agrégations et des relations
- S'appuyer sur des tuples représentatifs :

(Cours,	*Module,*	*Enseignant)*
(Automate,	ISI101,	E. Cousin)
(Parallélisme,	ISI201,	R. Rannou)
(Système,	SST201,	R. Rannou)

Invariant

- Définis en cours d'analyse
- Décrivent des propriétés caractéristiques du système
  - gaussien : les notes sont réparties sur une gaussienne
  - moyenne : la moyenne est comprise entre 10 et 12
- Propriétés toujours vérifiés dans des états stables

Fusion une méthode orientée objet

Relations dérivées

- Ce sont des relations composées
  - *ancêtre* est une composition de la relation de *parenté*
- Elles n'apparaissent pas dans le modèle
- Elles sont placées dans le référentiel
  - pour documenter,
  - aider à la compréhension

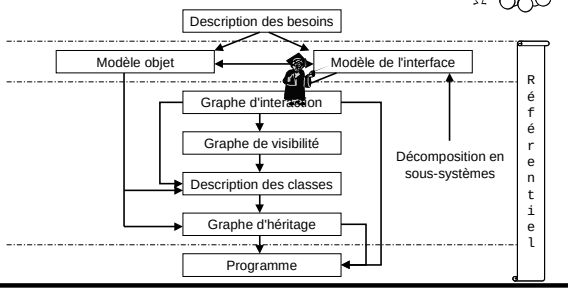
Modèle objet

- Composé du schémas et du référentiel
- Il représente le domaine du problème
  - des classes du système
  - des classes de l'environnement du système
- Sa complétude (vis-à-vis des besoins) et sa cohérence (interne) doivent être validées
- On cherche UN modèle, pas LE modèle

Analyse

- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

La méthode

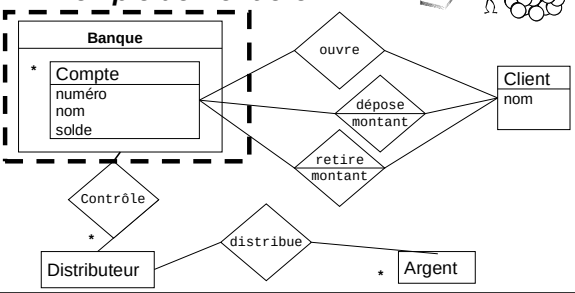


Délimiter la frontière

A partir du modèle objet

- Sélectionner les classes et relations à développer ou faisant partie du système à construire
- Les autres classes et relations
  - participent à la compréhension globale du système et,
  - Seraient utile en cas d'évolution, extension, du système ; on aurait une base commune.

Exemple de frontière



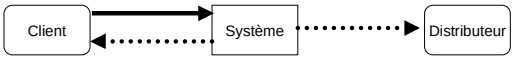
Fusion une méthode orientée objet

Analyse

- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

Environnement, agent

- Le système est actif
- Des événements provoquent des changements d 'état
- Un agent (humain, capteur, logiciel) est un objet externe au système qui communique avec lui par des événements
- L 'environnement est l 'ensemble des agents



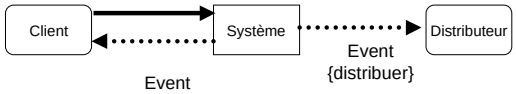
Evénement, opération

Lors de l 'analyse on imagine un système idéal

- Un événement est
  - instantané
  - atomique
- On distingue les événements d 'entrée et de sortie
- La communication est asynchrone (non bloquante)
- Un événement peut être caractérisé par des valeurs ou des objets
- Un événement d 'entrée déclenche des opérations

Modèle de l 'interface

Opération {ouvrir, déposer, retirer, vérifier_solde}

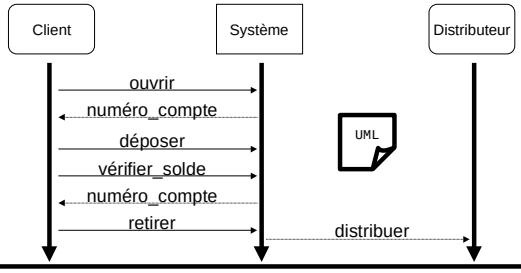


Event
{solde_courant, numéro_compte, fonds_insuffisants}

Elaboration de l 'interface

- Chercher les opérations et les événements
- Les opérations sont internes au système, déclenchées par des événements d 'entrée (on confond les deux)
- On s 'appuie sur des scénarios
  - diagrammes temporels (USE CASE) qui décrivent UNE interaction possible entre agents

Modèle de l 'interface



Fusion une méthode orientée objet

Le référentiel

Nom	Type	Description	Agent	Arguments
Client	agent	utilise le système pour...		
Distributeur	agent	contrôlé par le système pour…		
distribuer	event	une opération retirer est acceptée	Distributeur	
vérifier_solde	opération		Client	

Analyse

- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

Modèle du cycle de vie

- Généralise la notion de scénario
- Permet d 'exprimer des itérations, choix, succussions
- Apporte de l 'information pour l 'élaboration des opérations

Cycle de vie

- Alphabet : les opérations, les événements
  - Les sorties sont précédées de #
- Opérateurs
  - X.Y Y suit X
  - X | Y soit X soit Y
  - X* X se produit 0 ou plusieurs fois
  - X+ X se produit 1 ou plusieurs fois
  - [X] X se produit ou non
  - X||Y X et Y s 'entrelacent
- Substitution
  - on peut nommer les expressions
  - la récursivité est interdite

Exemple de cycle de vie

Vérification = vérifier_solde.#solde_courant

Transaction = retirer.(#fonds_insuffisants | #distribuer)
| déposer

Initialiser = ouvrir.#numéro_compte

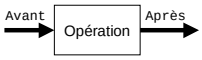
lifecycle Système_Bancaire
Initialiser.(Transaction|Vérification)*

Analyse

- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

Modèle des opérations

- Spécification déclarative
  - des changements d 'état (relation, attributs, instances)
  - des événements de sortie
- Une opération peut
  - créer une instance
  - changer la valeur d 'un attribut
  - ajouter ou supprimer un tuple dans une relation
  - envoyer un événement à un agent



- Pré, Postconditions et invariants

Syntaxe des opérations

Opération	nom
Description	texte
Reads	Items
Changes	Items
Sends	Agents and Events
Assume	Condition
Results	Condition

UML adopte OCL

Détail des opérations (1/2)

Opération	identificateur unique
Description	commentaire
Reads	Items nécessaires à la réalisation de l 'opération et qui restent inchangés - identificateur typé, objet, attribut ou relation - mot clé `supplied` indique un argument (qui ne peut pas être une relation
Changes	Items nécessaires à la réalisation de l 'opération et qui peuvent être modifiés - identificateur typé, objet, attribut ou relation - mot clé `new` indique la création d 'un nouvel objet

Détail des opérations (2/2)

Sends	liste les agents et les événements qui participent à l 'opération (source ou destination)
Assumes	précondition qui doit être satisfaite pour réaliser l 'opération - En cas de non satisfaction, le comportement est indéfini - n'a accès qu'aux objets visibles de Reads et Changes
Results	postcondition qui décrit le résultat de l 'opération si Assumes était valide - `Initial` référence la valeur d 'un objet avant l 'exécution de l 'opération et `Final`, sa valeur après

Exemple d'opération

Opération	ouvrir
Description	ouvre un nouveau compte pour un client
Reads	`supplied` nom: client.nom
Changes	Banque new Compte
Sends	client: {numéro_compte}
Assumes	client existe
Results	un compte est ajouté à la banque compte.solde = 0 et compte.nom = nom numero_compte est transmit au client

Nom local

Exemple d'opération

Opération	retirer
Description	demande d 'argent, refusée si le compte n 'est pas assez approvisionné
Reads	`supplied` numéro: compte.numéro `supplied` demande: argent
Changes	compte `with` compte.numéro = numéro
Sends	client {fonds_insuffisants} distributeur {distribue}
Assumes	le compte existe et le solde n'est pas négatif
Results	si (Initial compte.solde > demande) alors compte.solde est diminué de demande distribue est transmis au distributeur sinon fonds_insuffisants est transmis au client le solde n 'est pas négatif

Fusion une méthode orientée objet

Elaboration des schémas d 'opération

- S 'appuyer sur les cycles de vie pour Sends et Results
- La clause Results
  - l 'ordre des conditions ne doit pas être significative
  - doit être validée pour voir si elle est assez forte (contraignante)
  - les invariants qui risquent d 'être affectés sont présents

Elaboration des schémas d 'opération

- La spécification est satisfiable
  - pour toute valeur initiale satisfaisant Assumes, il existe des valeurs satisfaisant Results
  - sinon, étendre Results pour traiter des cas particuliers
  - renforcer Assumes pour reporter sur l 'environnement les cas problématiques
- Détailler Reads, Changes et Sends

Schéma d'opération : résumé

- Elaborer les clauses Assumes et Results
  - décrire chaque aspect du résultat par une condition
  - s 'appuyer sur le cycle de vie pour isoler les événements qui doivent être produits par Results
  - vérifier que Results n 'autorise pas des valeurs non désirées
  - Ajouter les invariants pertinents
  - vérifier la satisfiabilité
  - mettre à jour le référenciel
- Déduire Sends, Reads, Changes

Cycle de vie et précondition

Cycle \ Assumes	true	false
Accept	exécution	indéfini
Reject	ignoré	ignoré

Analyse

- Démarche
  - Construire un modèle objet du domaine
    - » objet, relation, opération, événement
  - Délimiter l'interface du système
    - » placer les frontières
  - Construire un modèle de l'interface
    - » cycles de vie
    - » opérations
  - Valider les modèles
- En alimentant le DATA-DICTIONARY (référentiel)

Complétude

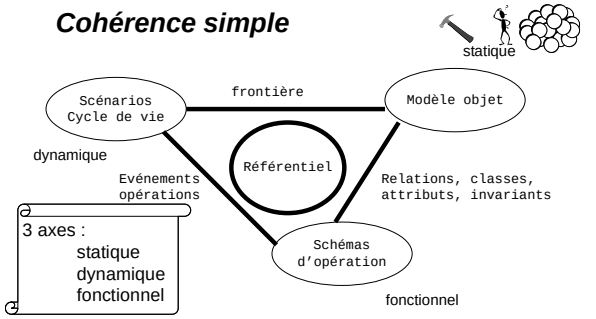
- Relire l 'analyse des besoins, clarifier avec le client
- Vérifier que
  - tous les scénarios (information dynamique) sont couverts par le cycle de vie
  - toutes les opérations sont décrites par un schéma
  - toute l 'information statique est contenue dans le modèle objet
  - tout le reste (invariants, définitions techniques, etc.) est dans le référentiel

Fusion une méthode orientée objet

Cohérence simple

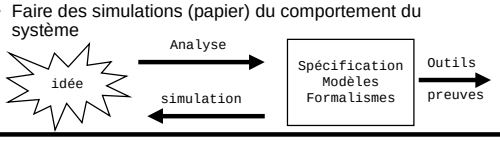
- Vérifier les intersections entre modèles en s 'assurant que :
  - toutes les classes, relations, attributs mentionnés dans les schémas d 'opérations apparaissent dans le modèle objet
  - les autres concepts (prédicats) sont présents dans le référentiel
  - toutes les opérations du cycle de vie ont un schéma
  - la frontière du système est cohérente avec le cycle de vie
  - tous les identificateurs de tous les modèles ont une entrée dans le référentiel

Cohérence simple



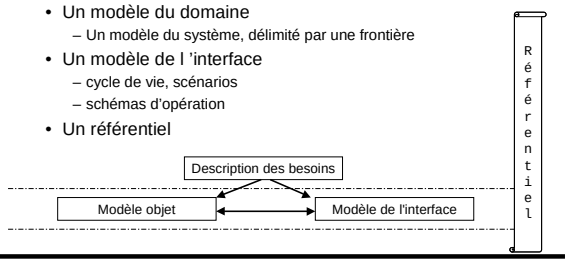
Cohérence sémantique

- S 'assurer que les conséquences des modèles sont cohérentes
  - les événements de sortie du cycle de vie et les opérations doivent être cohérents ; le schéma d 'une opération doit générer l 'événement de sortie qui la suit dans le cycle de vie
- Faire des simulations (papier) du comportement du système



Produits de l 'analyse

- Un modèle du domaine
  - Un modèle du système, délimité par une frontière
- Un modèle de l 'interface
  - cycle de vie, scénarios
  - schémas d'opération
- Un référentiel



Ce qui n'est pas analysé

- Les besoins
- La manière dont sont réalisées les opérations
- Pas d 'analyse fonctionnelle, pas d 'analyse du flux des opérations
- On ne s 'intéresse pas aux méthodes…
- Ce qu'on pourrait réutiliser

Des questions sur l 'analyse ?

Fusion une méthode orientée objet

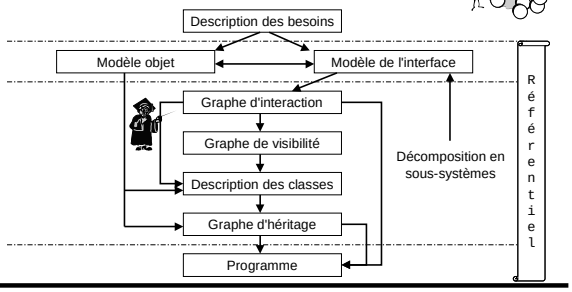
Objectif de la conception

- Raffiner les modèles abstraits de l 'analyse
- Détailler le fonctionnement des opérations
  - faire des choix techniques (architecturaux)
- Etudier les interactions entre objets
  - visibilité
  - navigabilité
  - ...

Conception

- Démarche
  - Construire un graphe d 'interaction
    - » pour chaque opération comment les objets communiquent entre eux
  - Construire un graphe de visibilité
    - » qui référence qui ? Temporairement ou non ?
  - Décrire chaque classe
    - » attributs, méthodes
  - Construire le graphe d 'héritage

La méthode

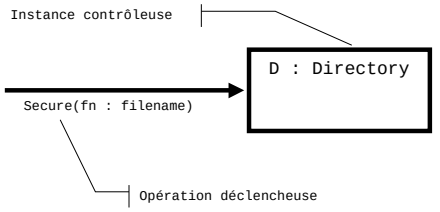


Etude des interactions

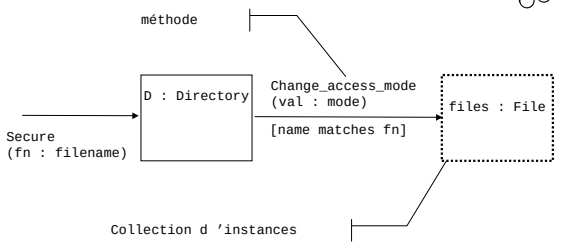
- Pour chaque opération on construit un graphe qui représente :
  - les objets impliqués
  - les messages qu'ils échangent
  - l 'ordre d enchaînement de ces messages
- On construit ainsi l'interface *nécessaire* des objets

> Je préfère construire des objets « cohérents » plutôt que minimum…dans un objectif de réutilisation !

Notations



Envoi de message



Fusion une méthode orientée objet

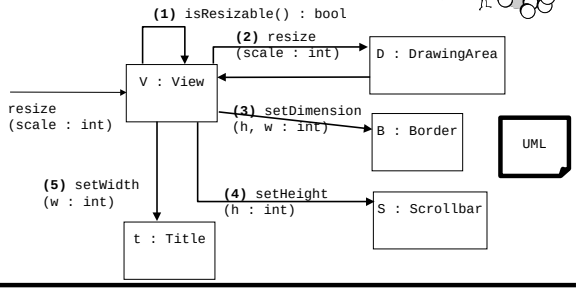
Condition

[Prédicat de sélection ; Prédicat d 'arrêt]

- prédicat de sélection
  - filtre les destinataires des messages
- prédicat d 'arrêt
  - une invocation qui rend ce prédicat valide empêche toute autre invocation

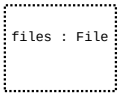
Quelle hypothèse faire sur l 'ordre des invocations ???

Séquencement

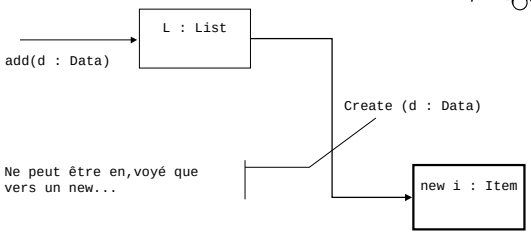


Séquencement

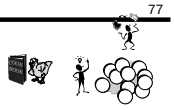
- Ordre lexicographique pour la séquence :
  (1) (1.1) (1.1.1) ...(1.2)…(2) (2.1) (2.2) …(2.3)
- Décoration pour les choix :
  (2) (2 ') (2 ' ') (2 ' ' ')
- Une astérisque pour les itérations
  - implicite vers une collection



Création d 'objets



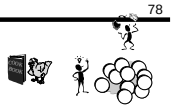
Elaboration du graphe des interactions



- Pour chaque opération :
  - identifier les objets impliqués dans le calcul (Reads, Changes, Sends)
  - Etablir le rôle de chaque objet
    - » qui contrôle ? Qui collabore ?
  - Déterminer les messages entre objet
  - Construire le graphe
    - » schéma et
    - » description textuelle des méthodes dans le référentiel
- On itère pour toutes les méthodes introduites (raffinement)

Essayer
- des contrôleurs
- des collaborateurs
- des messages différents !

Validation du graphe des interactions



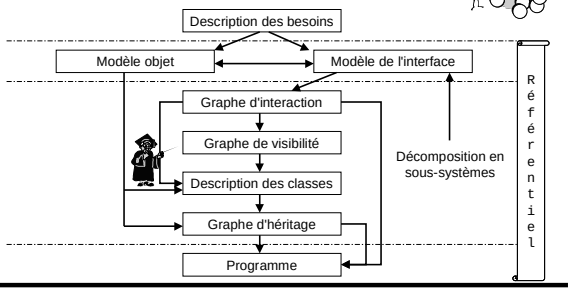
- Cohérence avec les modèles d'analyse
  - chaque classe du modèle objet est présente au moins une fois dans un graphe d'interaction
  - les classes apparaissant dans le graphe d'interaction sont présentes dans le modèle objet
- Vérifier les fonctionnalités
  - Chaque opération du modèle des opérations est décrite par un graphe d'interaction

Fusion une méthode orientée objet

Conception

- Démarche
  - Construire un graphe d 'interaction
    - » pour chaque opération comment les objets communiquent entre eux
  - Construire un graphe de visibilité
    - » qui référence qui ? Temporairement ou non ?
  - Décrire chaque classe
    - » attributs, méthodes
  - Construire le graphe d 'héritage

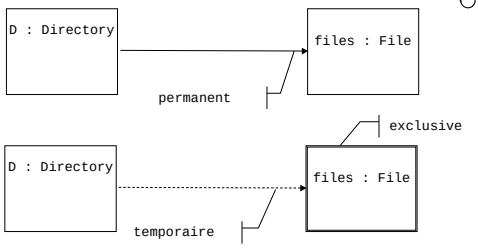
La méthode



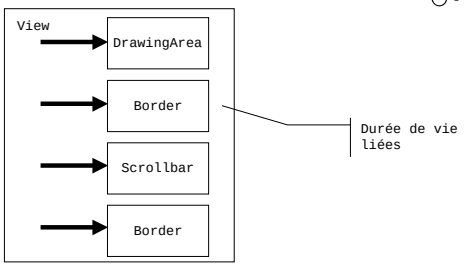
Etude de la visibilité

- On fait l'hypothèse que tous les objets se voient
- Les relations de visibilité :
  - Durée de vie : permanente ou temporaire
  - Exclusive ou partagée
  - Dépendance de durée de vie : création - destruction liées
  - mutabilité

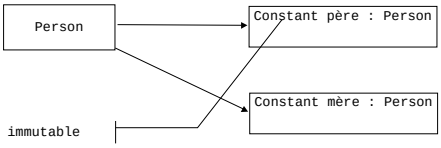
Notation



Notation



Notation



Fusion une méthode orientée objet

Elaboration du graphe de visibilité

- Pour tous les objets du graphe d'interaction, chaque flèche partant d'un objet donne lieu à une référence entre la classe de l'émetteur et celle du récepteur
- On choisit la nature de la dépendance selon :
  - la durée de vie
  - l'exclusivité de l'accès (ce n'est pas de l'exclusion mutuelle !)
  - le lien entre les durée de vie
  - la mutabilité

Validation du graphe de visibilité

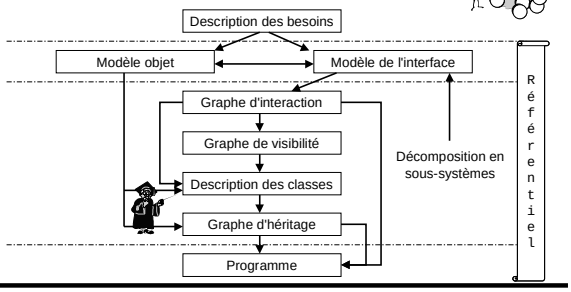
- Cohérence avec les modèles d'analyse
  - les relations du modèle objet doivent créer des relations de visibilité
- Cohérence mutuelle
  - les serveurs exclusifs sont référencés par un seul client
- Complétude
  - toutes les opérations sont réalisables à partir des relations du graphe de visibilité (Navigabilité)

UML

Conception

- Démarche
  - Construire un graphe d 'interaction
    - » pour chaque opération comment les objets communiquent entre eux
  - Construire un graphe de visibilité
    - » qui référence qui ? Temporairement ou non ?
  - Décrire chaque classe
    - » attributs, méthodes
  - Construire le graphe d 'héritage

La méthode



Description des classes

- On dérive des graphes précédents une description littérale des classes :
  - méthodes (nécessaires)
  - attributs
- On cherche les relations d'héritage

Langage

Class ::=
  class Name Inherit* Property* **endclass**
Inherit ::=
  isa Name
Name ::=
  Name++
Property ::=
  Attribute | Method
Attribute ::=
  [**attribute**] Mutability name : Sharing Binding Type
Method ::=
  [**method**] Name Arglist [: Type]

Fusion une méthode orientée objet

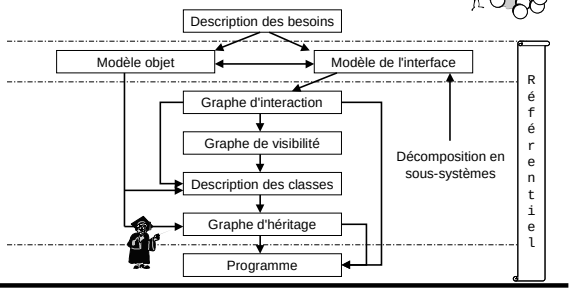
Langage

Sharing ::=
[**shared** | **exclusive**]
Binding ::=
[**bound** | **unbound**]
Mutability ::=
[**constant** | **variable**]

Conception

- Démarche
  - Construire un graphe d 'interaction
    - » pour chaque opération comment les objets communiquent entre eux
  - Construire un graphe de visibilité
    - » qui référence qui ? Temporairement ou non ?
  - Décrire chaque classe
    - » attributs, méthodes
  - Construire le graphe d 'héritage

La méthode

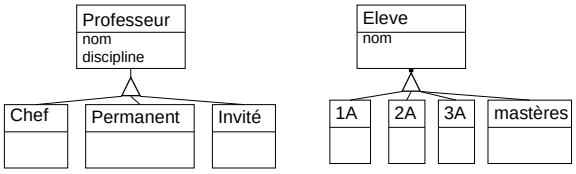


Etude de l'héritage

- Phase d'analyse : Etude de la relation de sous-type
  - spécialisation, généralisation
  - prescriptif
- Phase de conception
  - orienté implantation, réutilisation
  - descriptif

Notation

- Celles de l'analyse ...



Elaboration du graphe d'héritage

- Partir du modèle objet
- Analyser les opérations identifiées dans le graphe d'interaction
  - ajout de classe pour factoriser des opérations communes (classes abstraites)
- Analyser les références identifiées dans le graphe de visibilité
  - ajout de classe pour factoriser des relations/références communes (classes abstraites)

Fusion une méthode orientée objet

Validation du graphe d'héritage

- Préservation des relations de spécialisation, généralisation du modèle objet
- Toutes les classes du graphe d'interaction et de visibilité sont dans le graphe d'héritage
  - le graphe d'interaction ou de visibilité peut avoir été altéré suite à l'étude du graphe d'héritage (en ajoutant par exemple un classe abstraite)

Des questions sur la conception ?

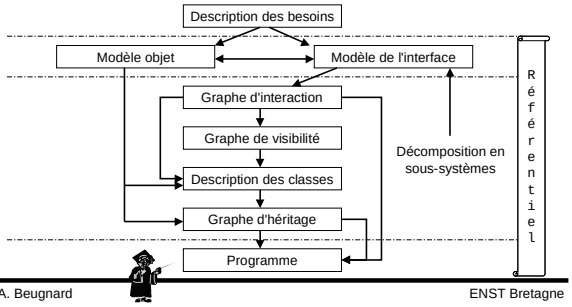
Objectifs de la réalisation

- Raffiner les modèles de la conception
- Coder les opérations en C++, Eiffel, Smalltalk, Java, …
- Etudier les performances

Réalisation

- Démarche
  - Coder, programmer
    - » le cycle de vie, les classes, les méthodes
  - Améliorer la qualité, optimiser
    - » les performances
      - à l'exécution,
      - de stockage
      - de communication
    - » inspecter le code
      - tests, relectures, standardisation, commentaires

La méthode



Coder, programmer (1/2)

- Traduire les cycles de vie en automate à état fini
  - pourrait être fait en fin d'analyse
  - est le moins orienté objet
  - entrelacement délicat
  - utiliser le pattern State
- Description des classes
  - traduction des attributs en fonction de leur caractéristiques (bound, constant, etc.)

Fusion une méthode orientée objet

Coder, programmer (2/2)

- Ecrire le corps des méthodes en fonction du graphe d'interaction
  - attention aux itérations
- Retour sur le référentiel
  - coder les invariants, les assertions
- Etudier la visibilité pour "packager"

Réalisation

- Démarche
  - Coder, programmer
    - » le cycle de vie, les classes, les méthodes
  - Améliorer la qualité, optimiser
    - » les performances
      - à l'exécution,
      - de stockage
      - de communication
    - » inspecter le code
      - tests, relectures, standardisation, commentaires

Améliorer la qualité, optimiser

- Robustesse
  - attention au détection et recouvrement d'erreur (cycle de vie en particulier)
- Performance
  - recherche d'optimisation langage ; tuning
- Revues
  - le code produit doit être relu et compris par des personnes autres que les auteurs
  - des tests confrontent le résultat attendu au réalisé

Pas de questions sur la réalisation !

Fusion allégé

Phase	*Modèle*	*Recommandations*
Analyse	Objet	identique
	Opération	omettre Assumes et Results
	Cycle de vie	se contenter de scénarios
Conception	Interaction	identique
	Visibilité	se contenter des références permanentes ou temporaires
	Classes	comme visibilité
	Héritage	identique
Réalisation		identique

Conclusion

- Bonne synthèse de concepts récents
- Notations dépassées par UML/OCL
- Démarche très structurée et adaptable
- Peu de prise en compte de la réutilisation
- Parallélisme, distribution, déploiement sont peu ou pas développés

Fusion une méthode orientée objet

Différences UML-Fusion

- UML-Fusion
  - Diagrammes de déploiement
- Fusion -UML
  - cycle de vie
  - la démarche

Bibliographie « méthode »

- Fusion
- JSD
- UML
- Catalysis
- OMT