

Génie Logiciel Avancé

Soutien

Stefano Zacchioli
zack@pps.jussieu.fr

Laboratoire PPS, Université Paris Diderot - Paris 7

8 juin 2011

URL <http://upsilon.cc/zack/teaching/1011/g1a/>
Copyright © 2011 Stefano Zacchioli
License Creative Commons Attribution-ShareAlike 3.0 Unported License
<http://creativecommons.org/licenses/by-sa/3.0/>



1 Spécification informelle des charges

2 Conception à objet

Rappel — Qu'est-ce qu'un système?

- Un **système** est un ensemble d'*éléments* interagissant entre eux suivant un certains nombres de principes et de *règles* dans le but de réaliser un *objectif*.
- La **frontière** d'un système est le critère d'appartenance au système.
- L'**environnement** est la partie du monde extérieure au système.
- Un système est souvent **hiérarchisé** à l'aide de *sous-systèmes*.
- Un système **complexe** se caractérise par :
 - ▶ sa dimension, qui nécessite la collaboration de plusieurs personnes ;
 - ▶ son évolutivité.

Exemple

une fourmilière, l'économie mondiale, le noyau Linux, ...

Rappel — Types des charges

Il y a trois grandes catégories de spécifications de système :

- Les spécifications **fonctionnelles** : on définit les services du système en termes de relation entre les sorties et les entrées.
- Les spécifications **non fonctionnelles** : ce sont les contraintes et les propriétés remplies par le système dans son intégralité, comme, par exemple, l'efficacité, la robustesse, la sécurité, ...
- Les spécifications **liées aux domaines d'activité** : ce sont des spécifications, fonctionnelles ou non fonctionnelles, qui définissent des informations ou des contraintes liées aux règles qui régissent certains domaines.

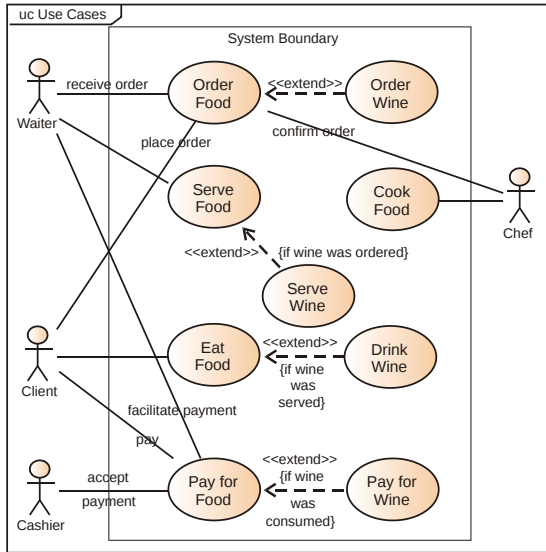
Rappel — Cas d'utilisation

- Les cas d'utilisation (*use-cases*) sont une technique fondée sur les scénarios qui jouent un rôle central dans le *Rational Unified Process* que nous étudierons au prochain cours.
- C'est un des piliers de la notation UML utilisée pour spécifier les systèmes à l'aide d'un modèle à objets.
- *Stricto sensu*, un cas d'utilisation regroupe plusieurs scénarios que l'on regarde à un niveau d'abstraction suffisamment haut pour les assimiler.

Un diagramme de cas d'utilisation en UML explicite :

- Les **acteurs** du système, c'est-à-dire les entités qui interagissent avec lui et lui sont **extérieurs**.
- Les scénarios du systèmes, regroupés et organisés suivant trois relations principales :
 - ▶ la généralisation : explicite des sous-ensembles de scénario ;
« Faire X, c'est une façon particulière de faire Y » \equiv « Y généralise X »
 - ▶ l'inclusion : explicite un préfixe d'un scénario ;
« Pour faire X, il faut d'abord faire Y » \equiv « Y est inclus dans X »
 - ▶ l'extension : explicite des événements supplémentaires.
« Faire Y, c'est faire X et Z » \equiv « Y étend X »

Rappel — Cas d'utilisation

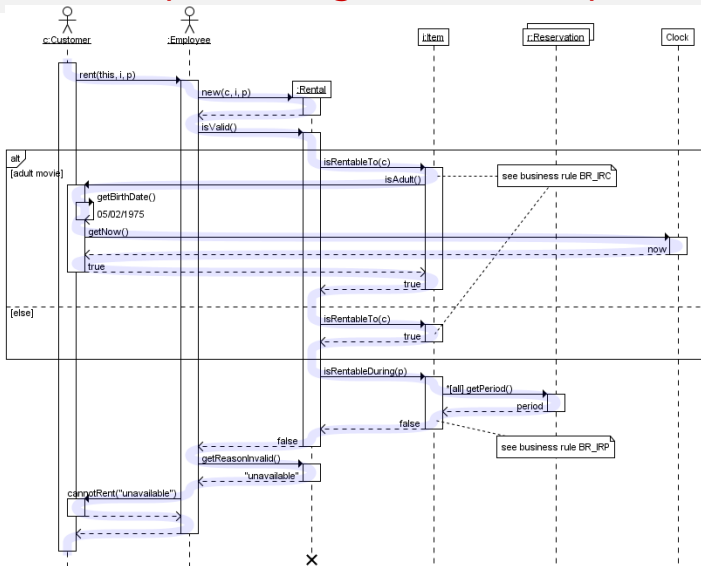


http://en.wikipedia.org/wiki/File:Use_case_restaurant_model.svg, GNU GFDL

Rappel — Diagramme de séquence

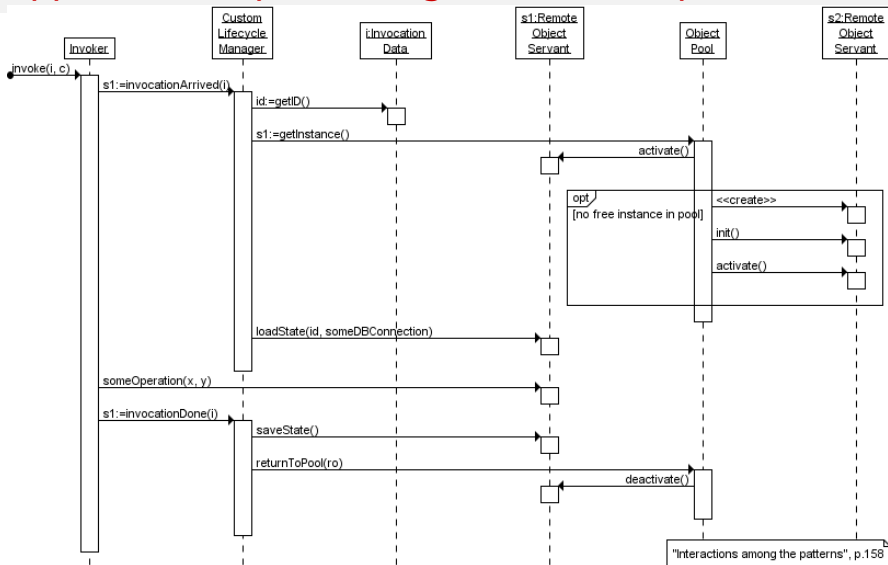
- Un diagramme de séquence présente les interactions entre les objets comme une **succession de message(/réponse)**.
- On peut y dénoter des contraintes de réponses **synchrones ou asynchrones**, des états bloquants, ...
- La **ligne du temps** est bien définie sur l'axe verticale du diagramme
- Cette notation met l'accent sur le **protocole de communication** entre les objets.

Rappel — Exemple de diagramme de séquence



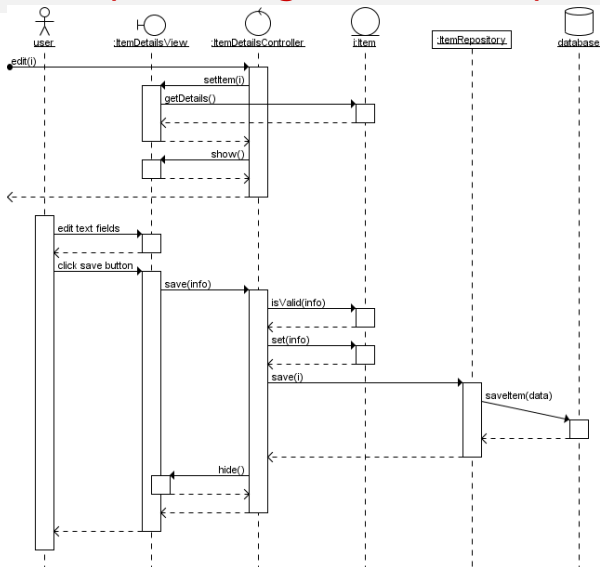
<http://www.tracemodeler.com/gallery/>

Rappel — Exemple de diagramme de séquence (cont.)



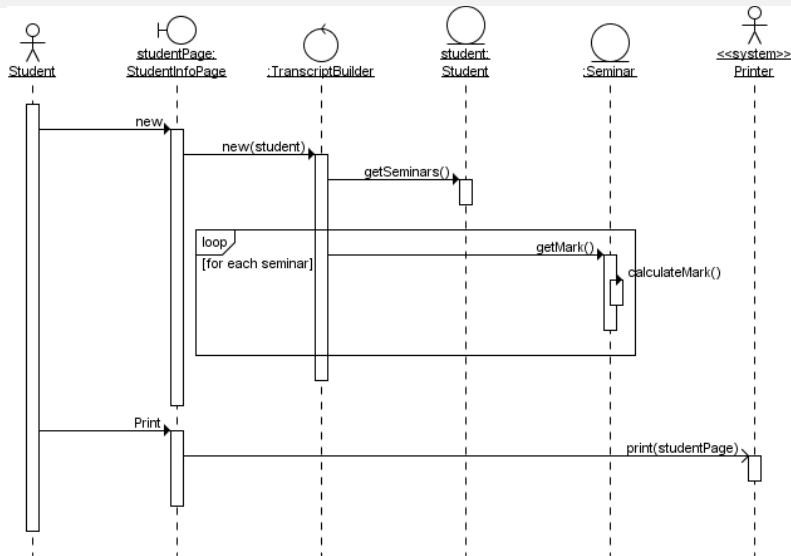
<http://www.tracemodeler.com/gallery/>

Rappel — Exemple de diagramme de séquence (cont.)



<http://www.tracemodeler.com/gallery/>

Rappel — Exemple de diagramme de séquence (cont.)



<http://www.tracemodeler.com/gallery/>

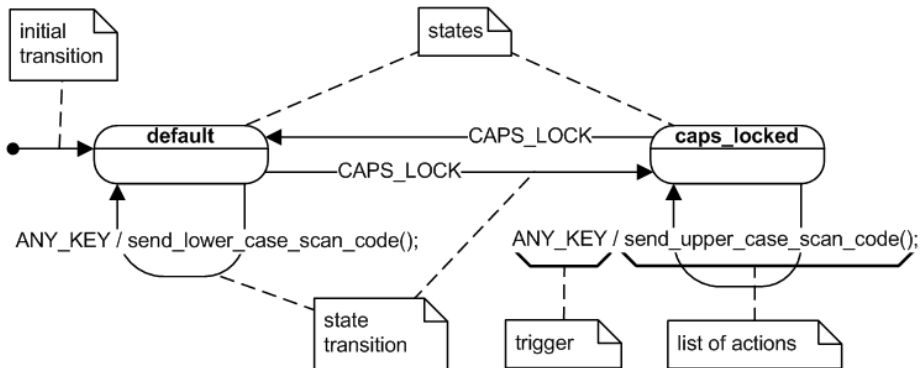
Rappel — Diagramme d'état

- Les diagrammes d'état (*UML state machine* ou *UML statechart*) représentent l'**évolution de l'état** du système (ou d'un sous-système) sous la forme d'un automate.
- Une transition de cet automate est suivie en **réaction à un événement**.
- Elle peut être conditionnée par des **contraintes** exprimées sur le système.

Intuition

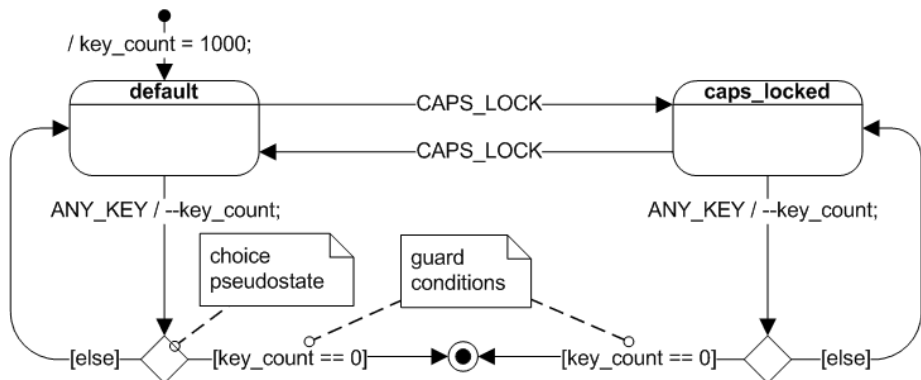
UML statechart	≈	automate fini	
	+	héritage entre les états	
	+	machine de Mealy	(sortie → état)
	+	machine de Moore	(sortie → état + trans.)
	+	variables & gardes	
	+	...	

Rappel — Exemple de diagramme d'état



http://en.wikipedia.org/wiki/File:UML_state_machine_Fig1.png

Rappel — Exemple de diagramme d'état (cont.)

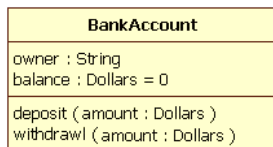


http://en.wikipedia.org/wiki/File:UML_state_machine_Fig2.png

1 Spécification informelle des charges

2 Conception à objet

Rappel — Exemple de diagramme de (une) classe



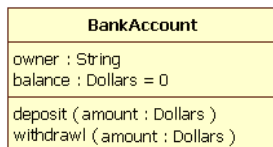
<http://en.wikipedia.org/wiki/File:BankAccount.jpg>

Figure: Diagramme de classe, réduit à une classe

on retrouve, pour chaque classe :

- 1 **nom** de la classe (unique dans le paquet)
- 2 **attribues** avec types (et valeur initial)
- 3 **méthodes** avec noms et types (d'entrée et sortie)

Rappel — Exemple de diagramme de (une) classe



<http://en.wikipedia.org/wiki/File:BankAccount.jpg>

Figure: Diagramme de classe, réduit à une classe

méthodes et attribues peuvent être annotés avec leur **visibilité** (*scope*) ; pour cela, UML offre des **préfixes** standardisés :

- + public (*default*)
- # protected
- private
- ~ package

Rappel — Relation entre classes

Les classes peuvent être mise en **relation**.

UML propose les relations suivantes :

Association un lien sémantique entre deux classes.

Agrégation/composition une relation d'appartenance.

Généralisation/spécialisation une relation d'abstraction.

Instanciation une relation d'affectation de paramètres.

Réalisation une relation de conformité entre une interface et une implémentation.

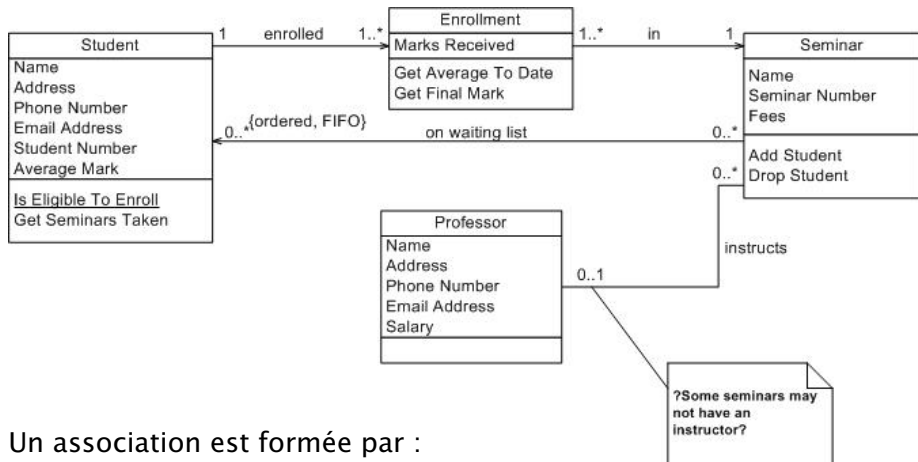
Rappel — Relation d'association

- Il s'agit de la notion mathématique de relation.
- Une relation a une arité à gauche et à droite.
- Chaque objet impliqué a un rôle dans la relation.

Exemples

- Un scénario **est joué** par un joueur dans un partie.
- Une action **est applicable sur** plusieurs objets d'une scène.
- Des objets **sont nécessaires pour** autoriser une action.

Rappel — Diagramme de classe avec association



Un association est formée par :

- un **nom** ;
- des **multiplicités** à gauche et à droite ;
- des **rôles** affectés à chaque objet.

Rappel — Syntaxe des multiplicités

Un entier “ n ” n objets interviennent dans la relation.

L'étoile “ $*$ ” plusieurs objets interviennent.

Le segment “ $n..*$ ” au moins n objets interviennent.

Le segment “ $n..m$ ” au moins n et au plus m objets interviennent.

Rappel — Agrégation/Composition

- L'**agrégation** est une relation d'appartenance
 - ▶ L'agrégation est forcément binaire (l'association non)
 - ▶ Exemples (*container*) :
 - ★ Les pièces d'un échiquier lui appartiennent.
 - ★ Les joueurs d'une partie appartiennent à la partie.
- La **composition** est une relation d'agrégation qui établit une relation de vie ou de mort d'un objet sur un autre.
 - ▶ Exemples
 - ★ Si l'échiquier est détruit alors ses pièces aussi.
 - ★ Si une partie est terminée, les joueurs peuvent en jouer une autre. (Ils survivent à la partie.)

Note

La composition est une relation assez subtile et souvent tributaire de certains choix d'implémentation. Il est souvent préférable de ne pas l'utiliser (sauf en C++ car la gestion explicite de la mémoire nécessite une réflexion précise sur la notion de durée de vie qu'il faut considérer dès la phase de spécification).

Rappel — Exemple de compositions et agrégations

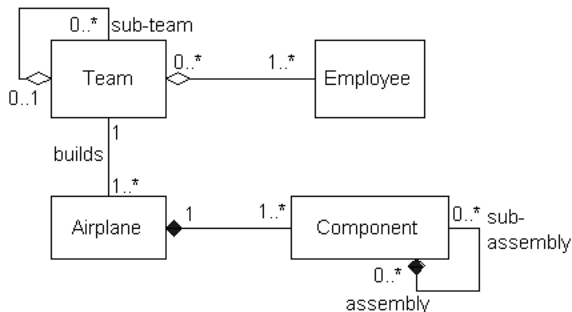


Figure: diagramme de classe avec compositions et agrégations.

- Le losange vide signifie « est agrégé à ».
- Le losange plein signifie « est composé de ».

Rappel — Composite : description

Exemple Dans un logiciel de dessin, il est possible de créer des groupes de formes. Un groupe de formes est alors considéré comme une forme.

Problème Comment voir une composition de formes comme une forme ?

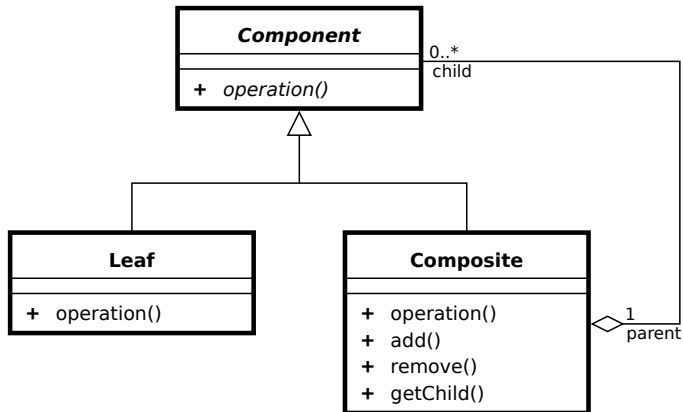
Rappel — Composite : description

Exemple Dans un logiciel de dessin, il est possible de créer des groupes de formes. Un groupe de formes est alors considéré comme une forme.

Problème Comment voir une composition de formes comme une forme ?

Solution Fournir un objet *Shape* qui implémente l'interface d'une forme ainsi que les services *add* et *remove*. Un objet *ShapeComposite*, héritant de *Shape*, correspond à un noeud possédant des sous-composants qui sont des formes. Lorsqu'un message est envoyé à une classe composite, elle le transfère à toutes ses composantes.

Rappel — Composite : UML



[http://en.wikipedia.org/wiki/File:Composite_UML_class_diagram_\(fixed\).svg](http://en.wikipedia.org/wiki/File:Composite_UML_class_diagram_(fixed).svg)

Exemple Composer les filtres n'est pas suffisant, un utilisateur avancé doit pouvoir programmer ses propres filtres (visuellement ou textuellement).

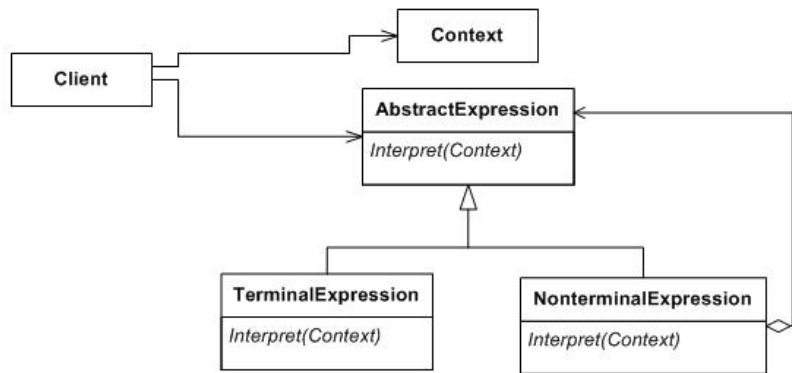
Problème Comment représenter un langage (syntaxe et sémantique à la fois) à l'aide d'objets ?

Exemple Composer les filtres n'est pas suffisant, un utilisateur avancé doit pouvoir programmer ses propres filtres (visuellement ou textuellement).

Problème Comment représenter un langage (syntaxe et sémantique à la fois) à l'aide d'objets ?

Solution Une classe abstraite *Expression* dont dérive les diverses constructions du langage.

Rappel — Interpreter : UML



Rappel — Interpreter : discussion

- il s'agit d'une généralisation du patron Command, pour cas plus complexes
- ce patron partage beaucoup des caractéristiques (et critiques, comme la sûreté de typage) avec le patron Composite
- on utilise le contexte pour stocker les données pas locales, nécessaire pour interpréter le langage