UML

pour les bases de données

Christine Plumejeaud
IR CNRS à Poitiers, UMR 7301 Migrinter

ANF RBDD – UML pour les bases de données

Ce(tte) oeuvre est mise à disposition selon les termes de la Licence Creative Commons

Attribution - Partage dans les Mêmes Conditions 4.0 International.

1. Vous êtes autorisé à :

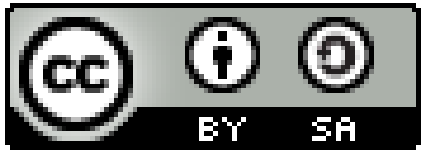
- **Partager** — copier, distribuer et communiquer le matériel par tous moyens et sous tous formats
- **Adapter** — remixer, transformer et créer à partir du matériel

2. Selon les conditions suivantes :

- **Attribution** — Vous devez mentionner le nom de l'auteur de la manière suivante :

  « Christine Plumejeaud-Perreau, MIGRINTER CNRS, 2022 »

- **Partage dans les Mêmes Conditions** — Si vous modifiez, transformez ou adaptez cette œuvre, vous n'avez le droit de distribuer votre création que sous une licence identique ou similaire à celle-ci.



Programme de l'ANF

8h30-12h30 / 10h

14h-17h50 / 16h (20 min)

1. Présentation RBDD - Introduction
2. Diagramme de cas d'utilisation
3. Diagramme d'activité/séquence

Lundi après-midi

4. Fil rouge : analyse fonctionnelle

Mardi matin

5. Diagramme de classe

Mardi après-midi

6. Persistance BDD
7. Héritage et POSTGRES

Mercredi

8. QCMs pour vous tester

Jeudi matin

9. Le patron composite
10. Lire UML - Evaluation en ligne.

Plan Lundi

14h-17h50
Pause à 16h (20 min)

1. Présentation RBDD : 15 min
2. Introduction : 30 min
3. Les cas d'utilisation : 30 min
   - Un exercice : 30 min

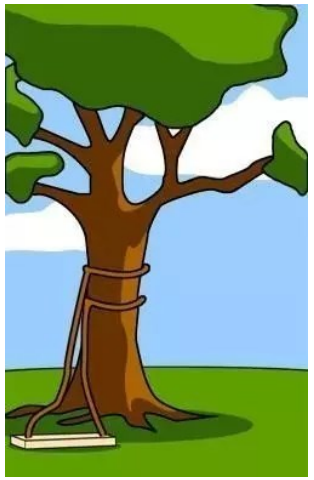
4. Diagramme d'activité : 15 min
   - Un exercice : 30 min
5. Diagramme de séquence : 15 min
   - Un exercice : 30 min

MOTIVATIONS POUR ENSEIGNER UML

Introduction

Une histoire bien connue - 1/5



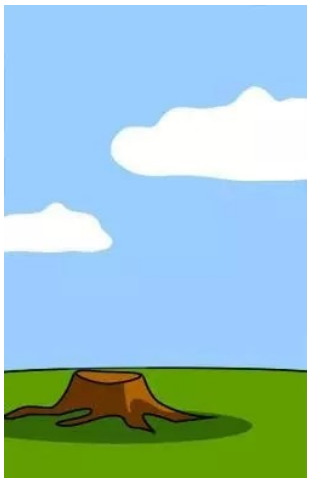
1. **Ce que le client a expliqué**
2. **Ce que le chef de projet a compris**
3. **La première d'analyse des ingénieurs**
4. **Ce que les ingénieurs ont conçu**

https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue - 2/5

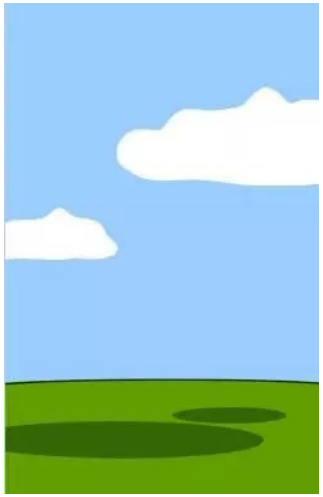
5. La présentation de la maquette au client
6. La première version : prototype
7. Les délais
8. Le soutien



https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue - 3/5

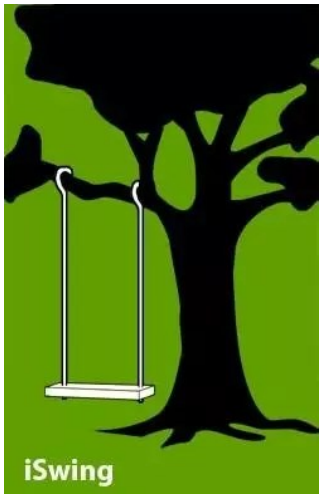


9. Les rapports

La documentation est essentielle pour capitaliser le savoir et le transmettre aux personnes arrivant dans l'entreprise.

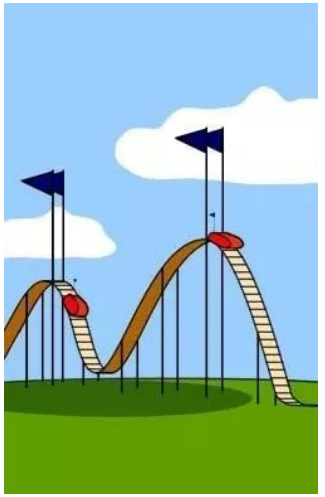
10. La promesse du commercial

Le commercial met en avant sa solution. La balançoire sera luxueuse, un véritable canapé connecté et intelligent. Le soleil levant est en prime !



11. La publicité de l'équipe marketing

Le marketing propose quelque chose d'innovant, suscitant l'émotion et l'adéquation à des valeurs telles que l'écologie, ou simplement une expérience de plaisir !



12. Ce que le client a payé

En termes de prix, le client a payé au delà de son budget.

https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

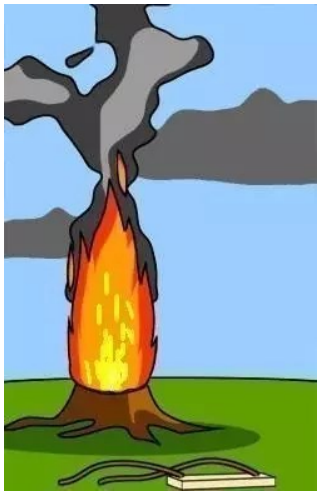
Une histoire bien connue - 4/5

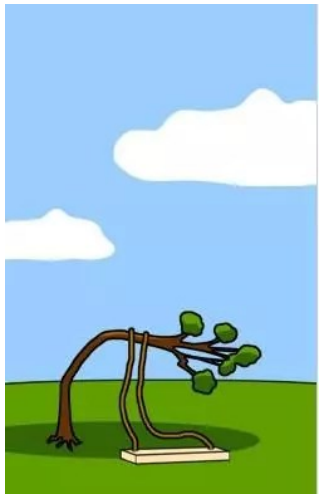
13. Les tests

14. La mise en place initiale

15. Le plan de reprise d'activité

16. La tierce maintenance applicative





https://www.astuces-pratiques.fr/electronique/analyse-du-besoin-ce-que-le-client-voulait

Une histoire bien connue – 5/5

Ce que le client voulait vraiment

Le gaspillage dans les bases de données

1. La production d'une base de données correspond à un besoin, un usage, voire à une question d'analyse
2. Le gaspillage peut être :
   - En amont : ne pas bien réfléchir à ses besoins, et choisir de faire une usine à gaz pour répondre à un besoin basique
   - En aval : récupérer un historique technique et fonctionnel, sans documentation appropriée

A propos de la documentation

A l'aire de l'open-data et de l'open-source

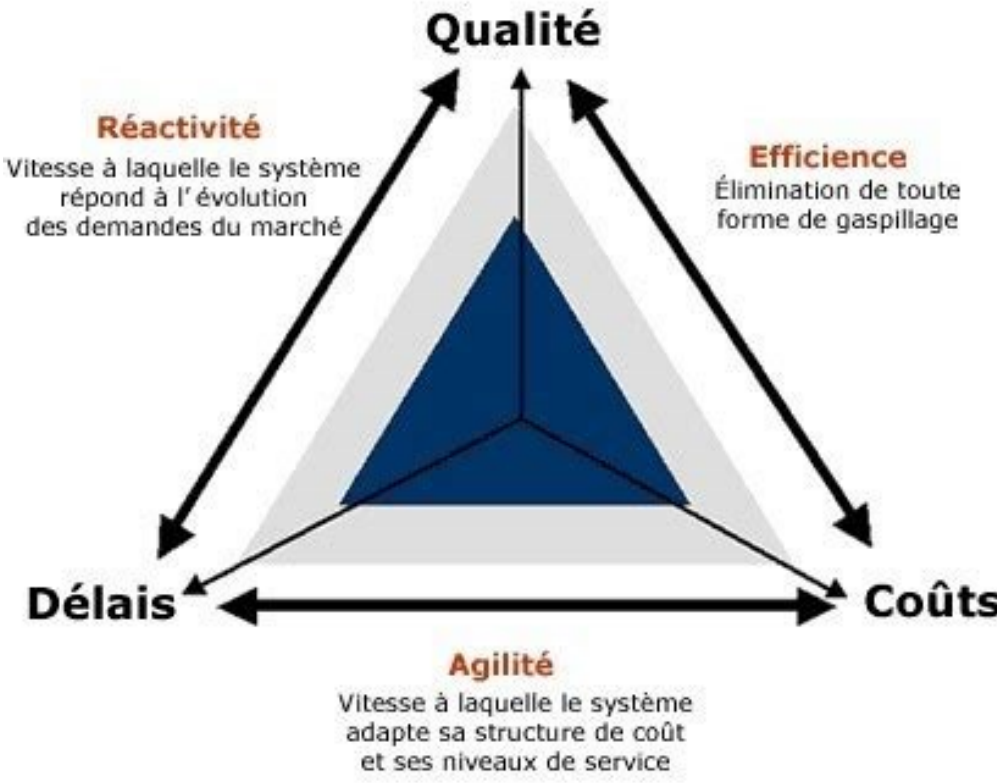
Les diagrammes et la documentation coûtent un temps précieux aux développeurs et deviennent rapidement obsolètes.

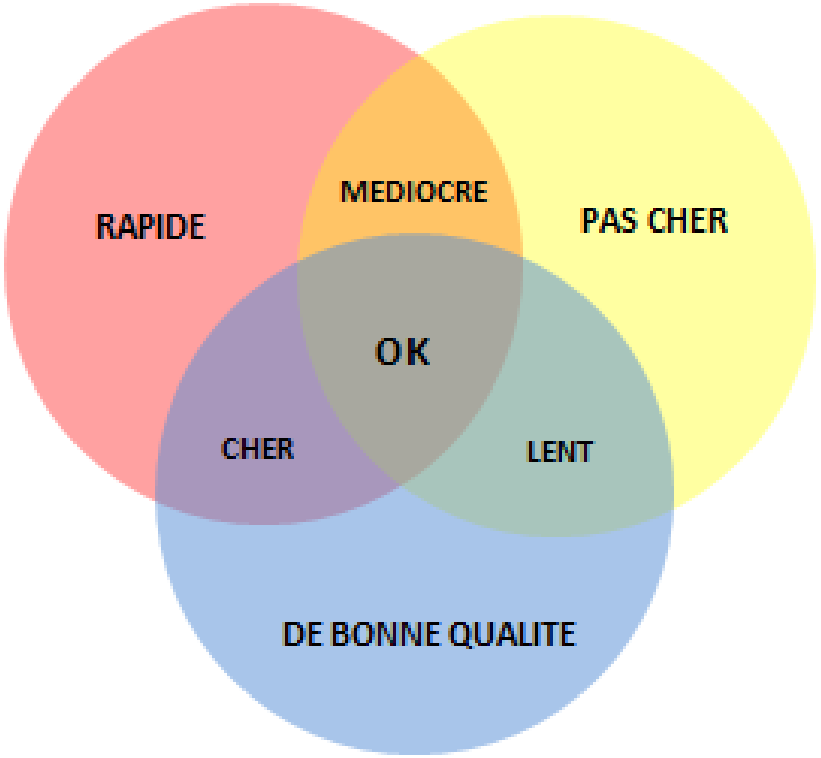
Mais l'absence de diagrammes ou de documentation ruine la productivité et nuit à l'apprentissage organisationnel.

https://mermaid.js.org/intro/

Mermaid répond à ce problème en permettant aux utilisateurs de créer des diagrammes facilement modifiables, qui peuvent également être intégrés à des scripts de production (et à d'autres éléments de code).

Gestion de projet





http://blog.ametragroup.com/excellence-operationnelle-demarche-damelioration-continue-triptyque-qualitedelaicout

https://www.la-rache.com/

- Accueil / News
- Présentation
- La F.A.Q
- Lexique
- Publications
- Consulting
- Formations
- Certificat de reussite
- En images
- Témoignages
- Liens

LA RACHE

/lamethoderache

/larache

Bienvenue sur le site de l'IILaR

(l'International Institute of La RACHE)

Le but de l'IILaR est de promouvoir la méthodologie de La RACHE. La RACHE, solution globale de génie logiciel, est un ensemble de techniques, de méthodes et de bonnes pratiques décrivant - des spécifications à la maintenance - comment produire du logiciel dans des conditions à peu près satisfaisantes et approximativement optimales.

Nous ne prétendons certes pas qu'il s'agisse d'une nouvelle méthode révolutionnaire, bien au contraire : La RACHE est mise en pratique depuis les débuts de l'informatique par de nombreux particuliers, étudiants et entreprises. Il semblerait même que la métempsychose récurrente du concept de la RACHE soit intrinsèquement axiomale chez certains grands éditeurs. Cependant, nous considérions que cette méthode nécessitait une formalisation ainsi qu'une normalisation rigoureuse afin d'être enfin utilisée fièrement et officiellement.

La RACHE est particulièrement adaptée à l'informatique, industrie qui a fait émerger le concept et l'a vulgarisé. Nonobstant, le corpus méthodologique, extrêmement flexible, est facilement transposable à de nombreux autres domaines.

News

01/01/2024

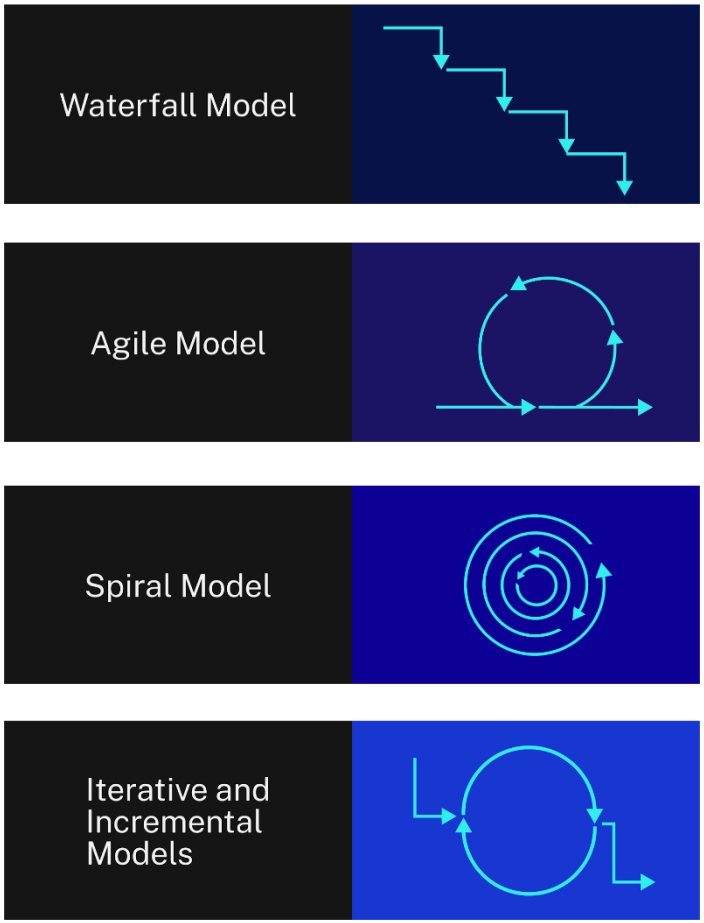
- 🎉🎈🎊 API NIOÙ HYÈRE et bons Jeux Olympiques à La RACHE !!! 🥇🥈🥉

01/05/2023

- Création de notre certification à La RACHE

Des méthodes de développement

https://www.door3.com/fr/blog/different-software-development-models



+ RUP

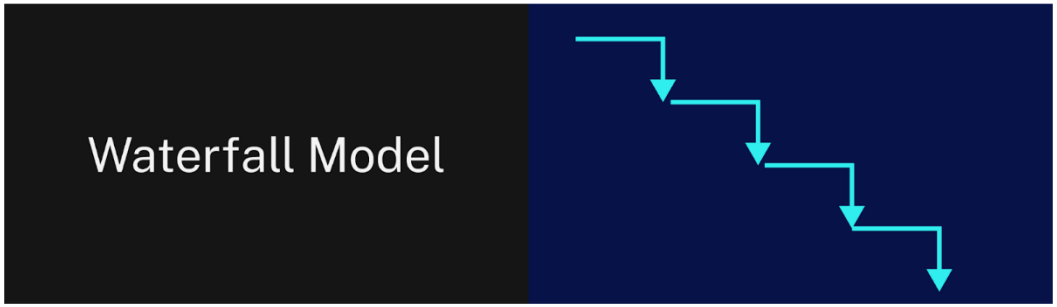
- limiter les effets de la contingence d'un projet informatique,
- à maîtriser au maximum le tryptique qualité/délai/coût

La rationalité contre les hasards

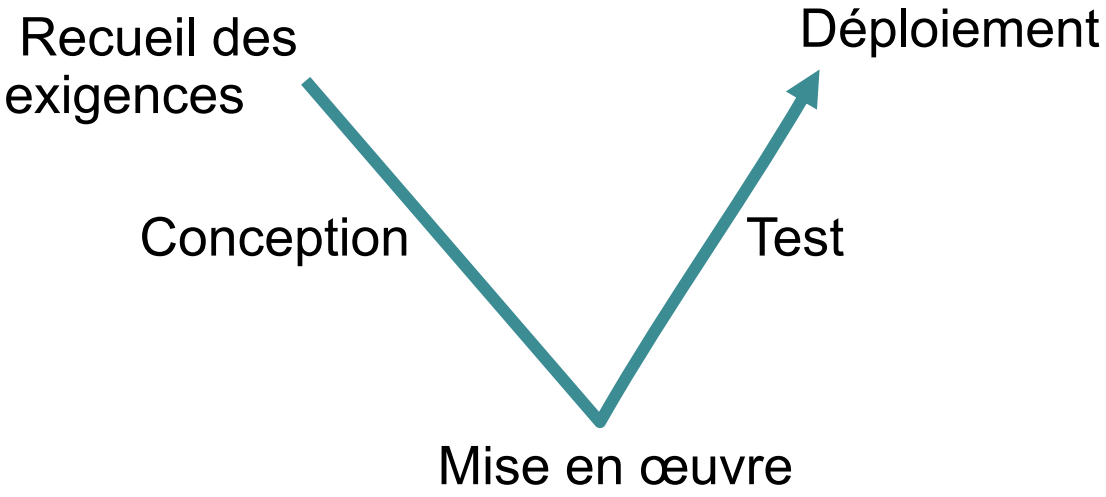
La coordination contre la spontanéité

Le respect

CHUTE d'EAU / V

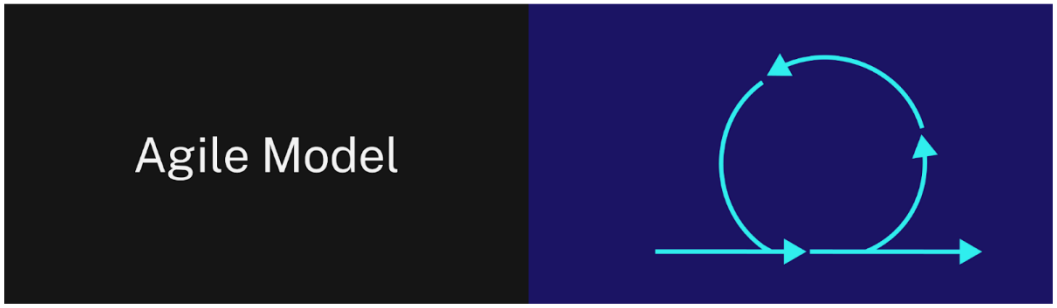


- **Recueil des exigences**
- Conception
- Mise en œuvre
- Test : unitaires, d'intégration, de recette
- Déploiement



Et après, c'est terminé ?

AGILE

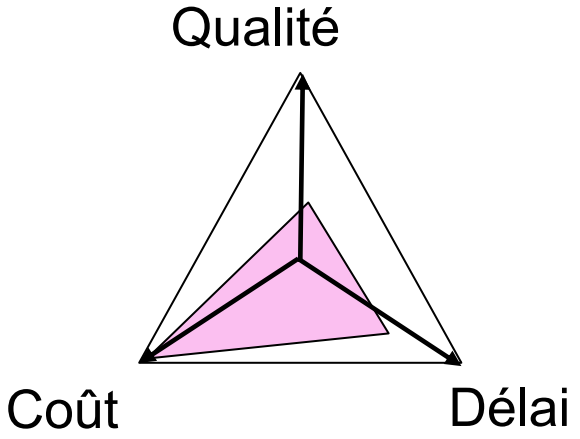


Sprint = une itération

- **Planification du sprint**
- Exécution du sprint
- Intégration et test continus
- **Révision et retour d'information** (implication des clients / utilisateurs)
- Rétrospective du sprint

« idéal pour les projets dynamiques dont les exigences évoluent

et il encourage la livraison rapide de solutions logicielles »



SPIRALE

- **Planification**
- Analyse des risques
- Prototypage
- Évaluation

Dans le modèle en spirale, la phase de planification commence par l'identification des objectifs, des exigences et des contraintes du projet. L'équipe de développement collabore avec les parties prenantes pour recueillir les exigences initiales et définir la portée globale du projet.

- Développement
- Test et intégration
- Déploiement
- Maintenance

La maintenance suppose une permanence du service informatique (vous ?) dans le temps…

https://www.door3.com/fr/blog/different-software-development-models

Itératives et incrémentales

Engagement des utilisateurs ➔ disponibilité ?

- Planification initiale
- Itération 1
- Tests et retours d'information
- Itération 2
- Test et retour d'information
- Itération 3
- Test et retour d'information

*Les modèles itératif et incrémental sont des modèles de développement de logiciels qui permettent à l'équipe de développement de livrer un logiciel fonctionnel de manière incrémentale au fil du temps. Chaque itération ajoute de la valeur au projet, en s'appuyant sur les bases existantes et en intégrant les commentaires des utilisateurs et des parties prenantes. Cette approche permet un développement rapide, **un engagement précoce des utilisateurs** et une flexibilité permettant de s'adapter à l'évolution des besoins.*

La méthode RUP

RUP comme Rational Unified Process

Processus unifié rationnel

RUP est un processus de développement logiciel de *Rational*, une division d'IBM en 4 phases :

1. **Début** - L'idée du projet est énoncée. L'équipe de développement détermine si le projet mérite d'être poursuivi et quelles ressources seront nécessaires.
2. **Élaboration** - L'architecture du projet et les ressources requises sont ensuite évaluées. Les développeurs considèrent les applications possibles du logiciel et les coûts associés au développement.
3. **Construction** - Le projet est développé et complété. Le logiciel est conçu, écrit et testé.
4. **Transition** - Le logiciel est mis à la disposition du public. Les derniers ajustements ou mises à jour sont effectués en fonction des commentaires des utilisateurs finaux.

Il doit **éviter le gaspillage** des ressources et réduit les coûts de développement inattendus.

https://techlib.fr/definition/rup.html#:~:text=Signifie%20%C2%ABProcessus%20unifi%C3%A9%20rationnel%C2%BB,les%20tests%20et%20le%20d%C3%A9ploiement

Un bon cahier des charges

1. Définit le contexte autour du projet : évoquer la situation prévalant au moment donné : décrire l'état du parc logiciel et informatique, les motivations pour le nouveau projet (**POURQUOI**)
2. **Établit les besoins** et les contraintes liées au projet

   compatibilité informatique de la solution avec l'architecture de l'entreprise et les compétences des utilisateurs
3. Mentionne les résultats attendus

   - livraison de quoi (code, données, graphiques),
   - et comment (fichier/URL/plateforme), et où (dispositif fixe/mobile)
4. Définit un calendrier, des échéances
5. Définit les utilisateurs finaux

   nombre et caractéristiques des utilisateurs : ergonomie adaptée (accessibilité), Langues (FR, EN, ES, …), puissance de calcul…

UML

Le **Langage de Modélisation Unifié**, de l'anglais *Unified Modeling Language* (**UML**), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.

Les objectifs de UML

1. UML
   - notation graphique de modélisation à objets (9 types de diagrammes)
   - UML n'est PAS une méthode de développement, ni un langage de programmation
2. Objectifs :
   - Représenter des systèmes entiers par des concepts objets (pas juste les logiciels)
   - Lier explicitement les concepts et le code
   - Modéliser à différents niveaux de granularité
   - Langage utilisable à la fois par les humains et les machines par un formalisme bien établi ($\rightarrow$ génération de code)
   - Encourager l'utilisation des outils Orientés Objets
   - Offrir des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base

Intérêts de l'approche objet

1. « *Naturelle* » car basée sur le domaine d'application. Facilite la communication avec les utilisateurs.
2. Meilleur support pour *l'évolution* des besoins
   - Modélisation plus stable
   - Les évolutions du système ne remettent pas l'architecture en cause
3. Facilite la *réutilisation* de composants

•Encapsulation	→ systèmes modulaires
•Héritage	→ système extensibles
•Association	→ systèmes complexes

L'unification des méthodes objet

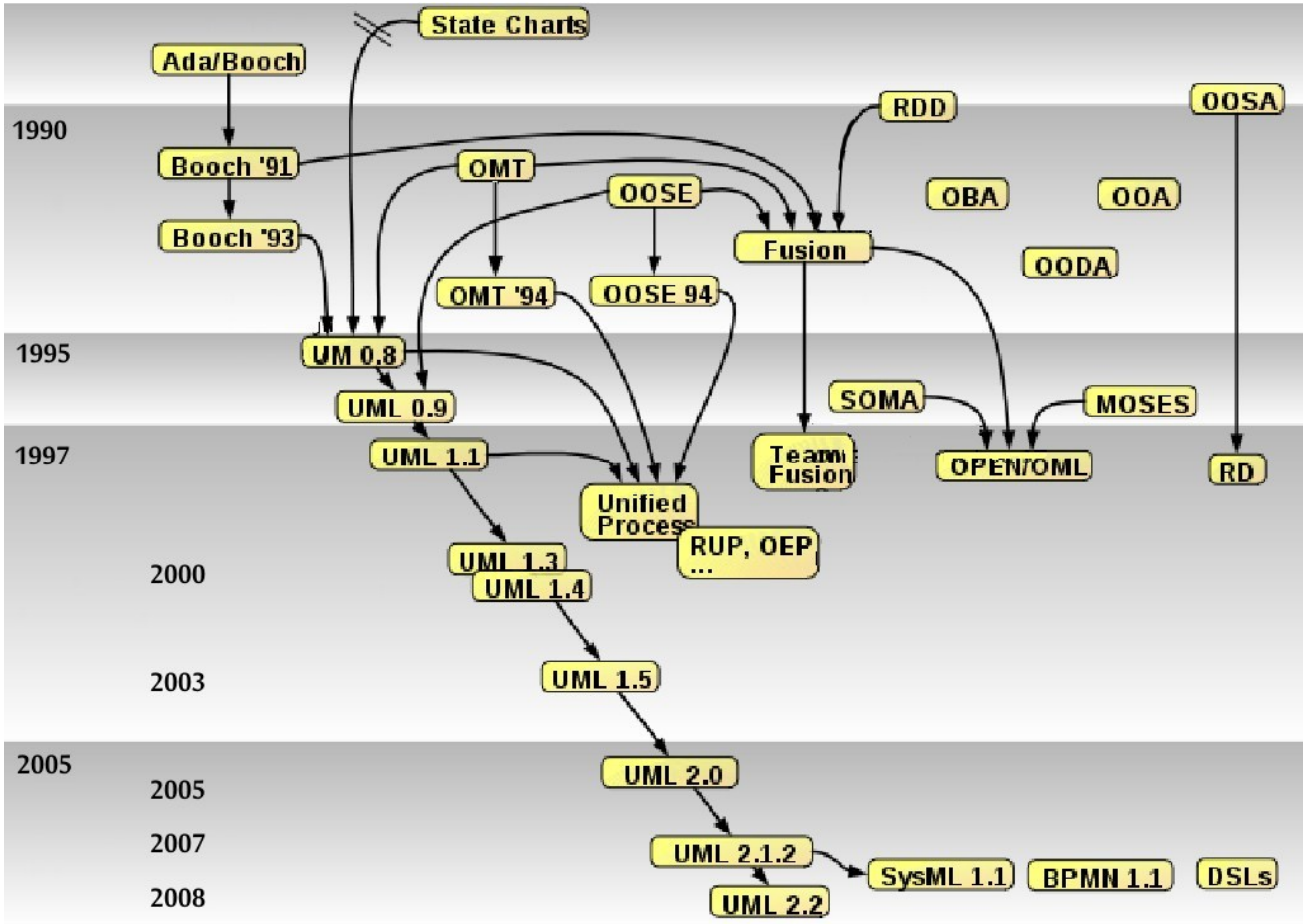
OMG (*Object Management Group*) (http://www.omg.org/) fondé en 1989. Consortium d'industriels pour la standardisation des méthodes objets, à l'origine du standard CORBA

Objectifs : maximiser la portabilité et la portabilité des logiciels par la définition de standards industriels, pour le développement d'applications commerciales orientées objet.

570 membres en 1995

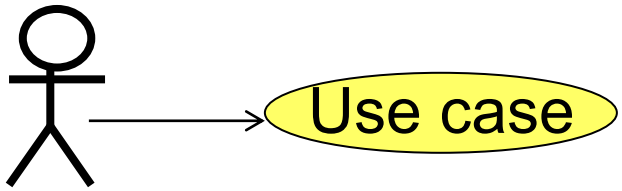
La genèse d'UML

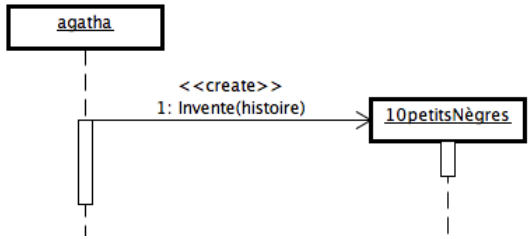
1. Foisonnement
2. Unification - 1995
3. Standardisation -
4. UML 2.0 - 2005



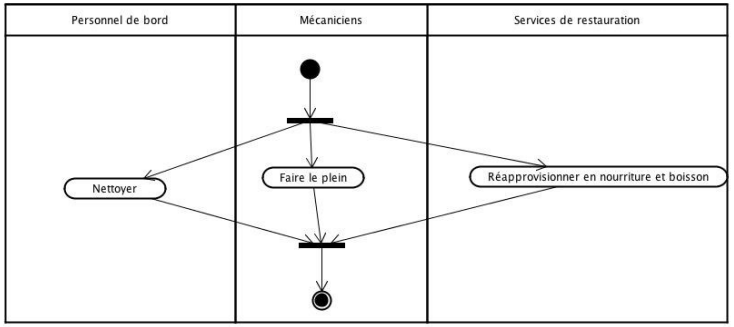
Les 9 diagrammes UML

1. Aspects fonctionnels:
   - cas d'utilisation
   - (diagrammes de séquence)
   - (diagrammes d'activité)

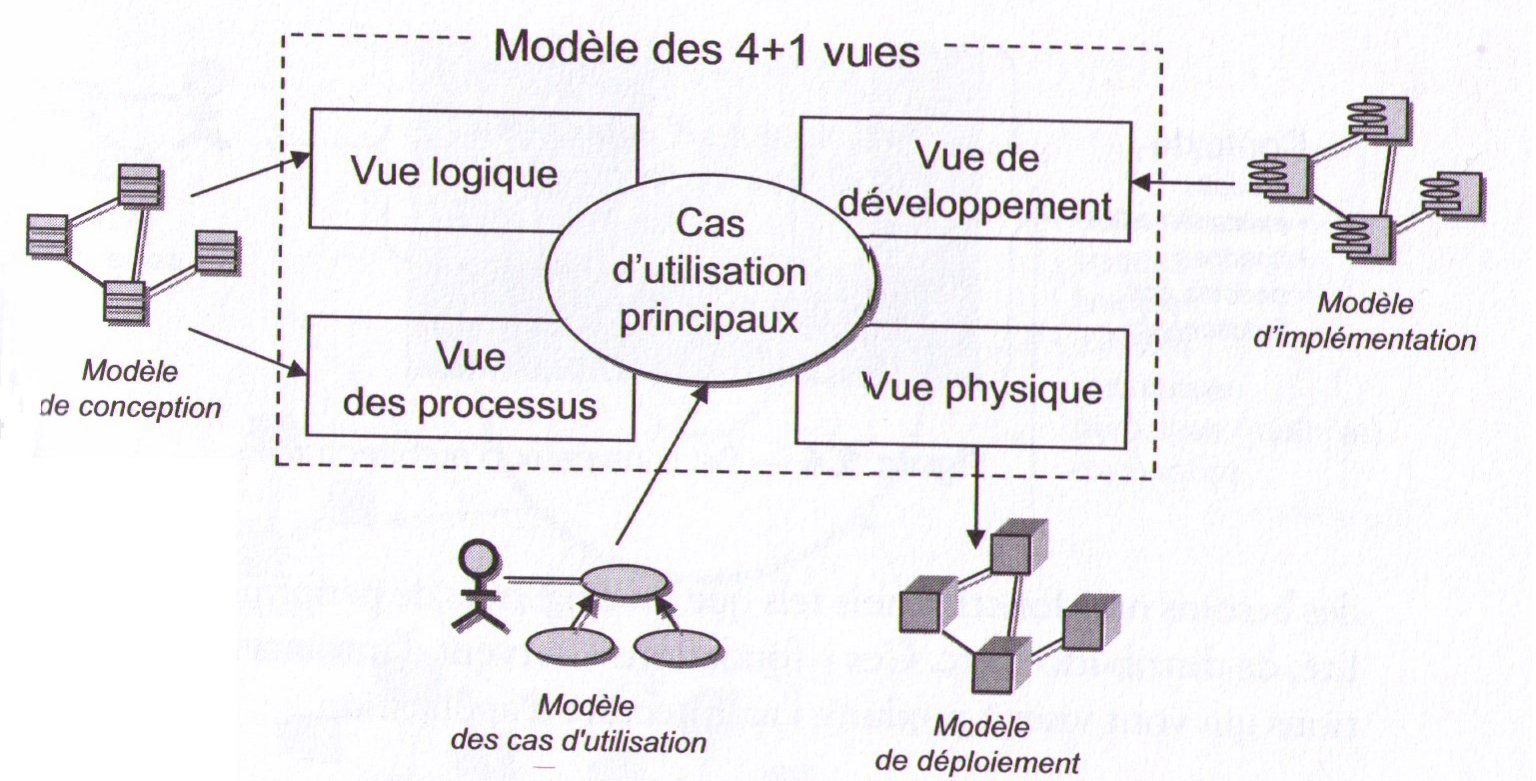




2. Aspects statiques
   - diagrammes de classes
   - diagrammes d'objets
   - Les diagrammes de composants / packages
   - Les diagrammes de déploiement
3. Aspects dynamiques
   - diagrammes de séquence
   - diagrammes d'activités
   - automates d'états-transitions
   - diagrammes de collaboration



Intégration d'UML au RUP



- Piloté par les cas d'utilisation
- Centré sur l'architecture (la structure du système)
- Itératif et incrémental

Modéliser les interactions autour d'une base de données

Décrit les interactions entre objets, et la façon dont ils collaborent ➔ comportement global du système

1. Cas d'utilisation

   capturent les besoins, et les interactions du système avec l'extérieur

2. Les diagrammes de séquence

   représentent la séquence des interactions entre les objets dans le temps

3. Les diagrammes d'activité

   illustrent le flux de contrôle entre les étapes d'un traitement

Diagramme de cas d'utilisation

Acteur/système/use case/include/extend

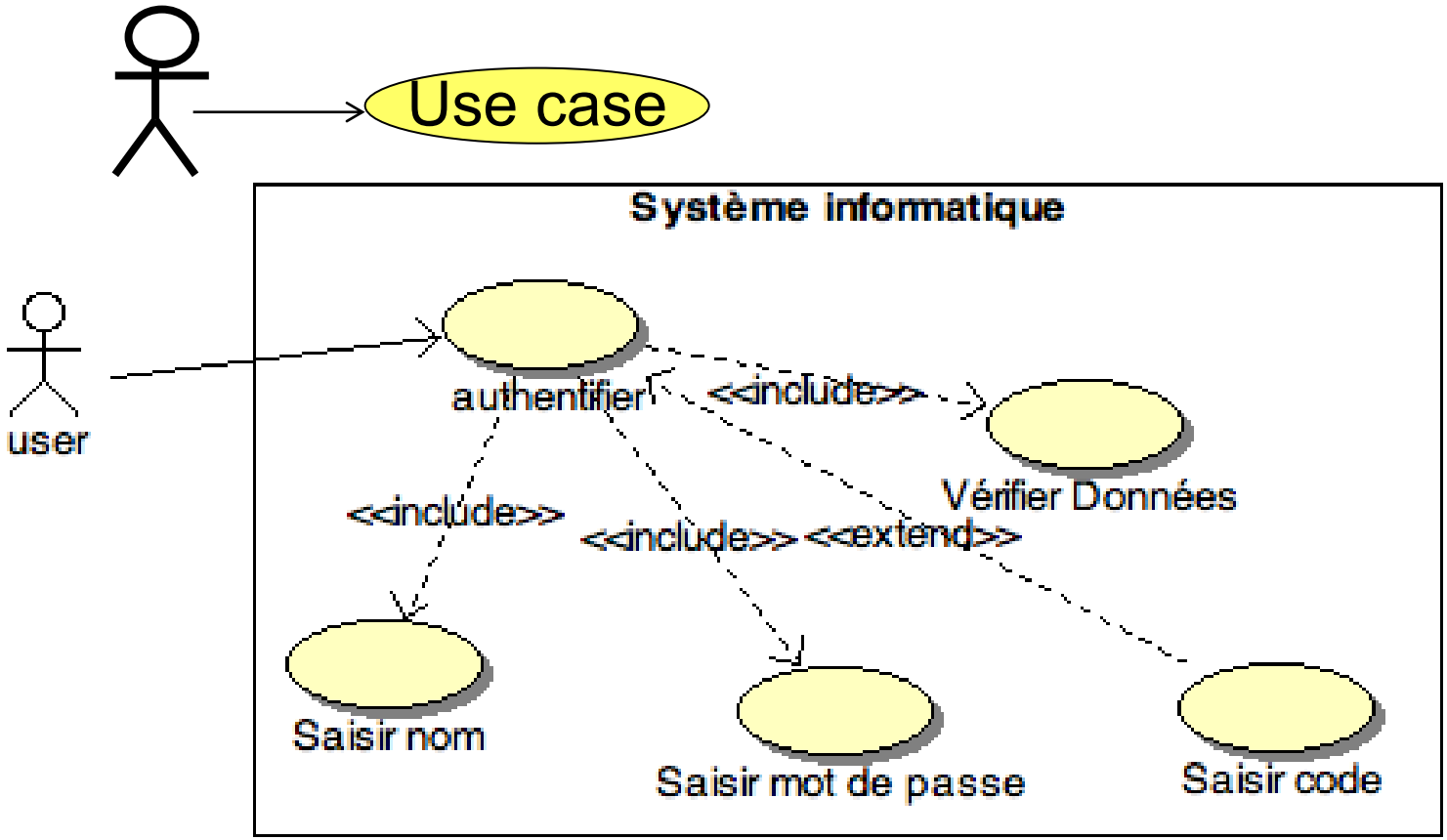


Diagramme de séquence

Objet/fil de vie/fin de vie/loop/opt
message/synchrone/asynchrone/retour

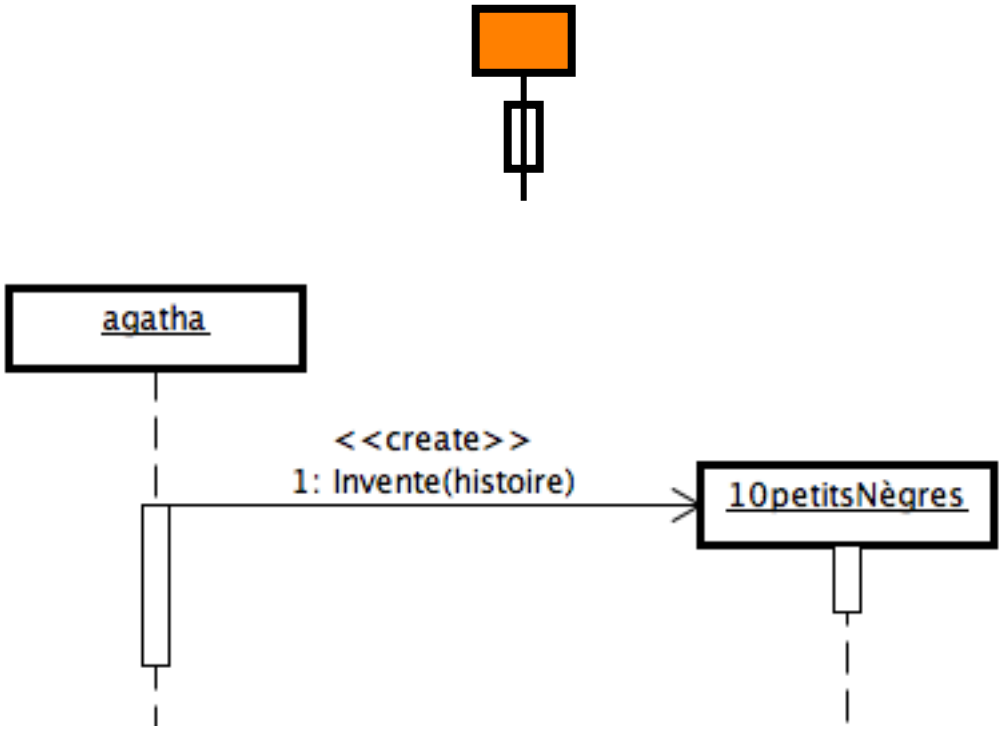
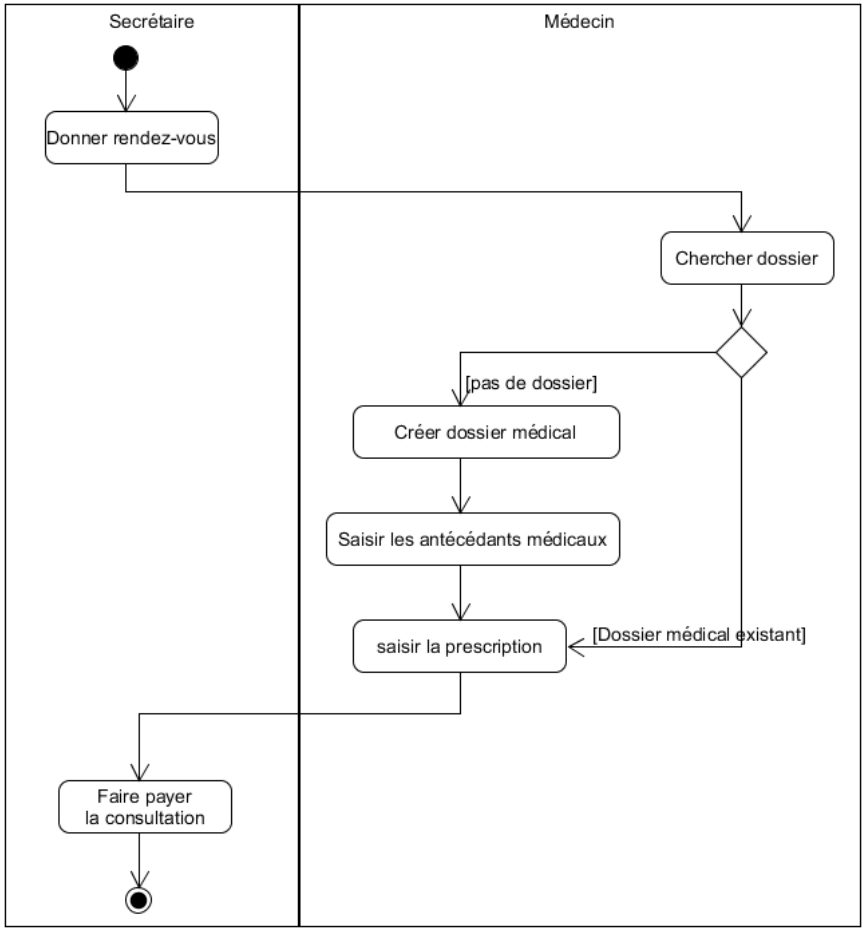


Diagramme d'activités

Départ/activité/fin/embranchement/garde/fourche/synchronisation

Couloir/responsabilité



Avec quels outils

1. Du papier et un crayon
2. Sinon…
   - UMLet https://www.youtube.com/watch?v=3UHZedDtr28
   - BOUML
   - ArgoUML
   - plantUML : https://github.com/plantuml/plantuml
   - MermaidJS

Références UML

https://laurent-audibert.developpez.com/Cours-UML/

Pascal Roques. **UML2 par la pratique (étude de cas et exercices corrigés)**. Eyrolles, 5e édition, 2006.

James Rumbaugh, Ivar Jacobson, and Grady Booch. **UML 2.0 Guide de référence**. CampusPress, 2004.

DÉCRIRE LES CAS D'UTILISATION

Avec UML

Lundi 9 Septembre 2024 - 14H45-16H

Capturer les besoins des utilisateurs

Un SI doit répondre aux besoins des utilisateurs. Leur capture nécessite :

- l'étude du système existant
- l'acquisition des nouveaux besoins
  - Fonctionnels : que doit faire le futur SI ?
  - Non fonctionnels : performances, sécurité, etc.
  - Utilisabilité : contexte de l'usage
- l'utilisation de techniques d'acquisition
  - Analyser la documentation existante
  - Questionner, observer les utilisateurs

La vue des cas d’utilisation

1. Cette vue définit les besoins des clients du système à l'aide de scénarios et de cas d'utilisation
2. Elle justifie l'architecture du système sur la satisfaction (la réalisation) de ces besoins.
3. Elle est prioritaire et accompagne toutes les autres vues pour leur donner leur cohérence.

Décrire les besoins des utilisateurs

Spécifier avec UML avec les diagrammes de cas d'utilisation pour modéliser et documenter les besoins

- Un cas d'utilisation n'est pas un besoin, mais une fonctionnalité du système devant répondre à un besoin
- Les diagrammes de cas d'utilisation ne permettent pas de répondre à des besoins non fonctionnels
- Il est nécessaire d'établir la liste des besoins non-fonctionnels pour compléter la spécification des besoins

Diagramme de cas d'utilisation (*use case*)

1. Formalisé par Ivar Jacobson
2. Décrivent sous la forme d'actions et de réactions le comportement d'un système du *point de vue utilisateur*
3. Définissent les *limites du système* et les relations entre celui-ci et son environnement.
4. Manière spécifique d'utiliser un système : image d'une fonctionnalité du système déclenchée en réponse à la *stimulation d'un acteur externe*.

Exemple de besoin

Gestion des malades dans un cabinet médical

1. Un médecin doit pouvoir
   - Créer des dossiers médicaux des patients, les modifier, consulter, supprimer.
   - Créer des ordonnances, modifier, imprimer et supprimer
   - Créer et modifier des instructions à l'attention du secrétariat
   - Créer et modifier les informations sur le patient (informations non médicales)
2. Une secrétaire doit pouvoir
   - Créer et modifier des informations sur un patient (si informations non médicales)
   - Editer une feuille de soin
   - Imprimé un récépissé lors de l'encaissement d'un paiement
   - Etc.

Diagramme d'utilisation : concepts

- Il comprend les acteurs, le système et les cas d'utilisation eux-mêmes.
- Les acteurs déclenchent des cas d'utilisation
- les cas d'utilisation sont contenus dans le système

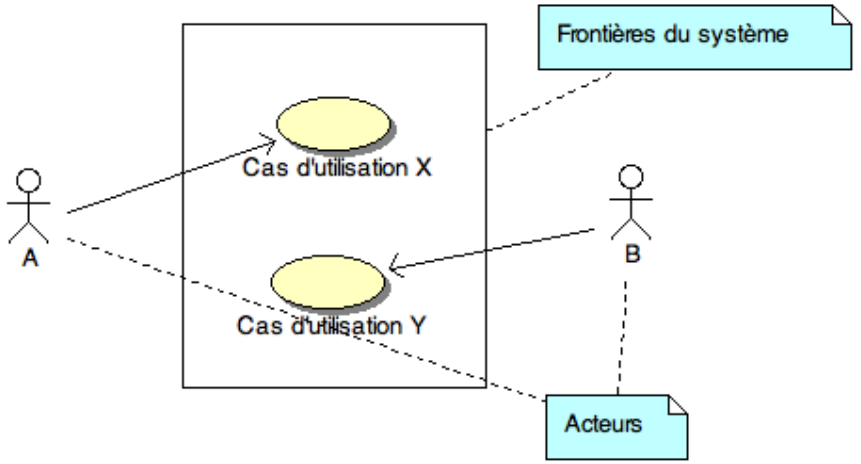


Diagramme d'utilisation : l'acteur

1. Un acteur représente tout ce qui est *externe au système*, humain ou non, qui *interagit* avec le système et qui correspond à une catégorie d'utilisateurs (plus précisément à un *rôle*).
2. Les acteurs se déterminent en observant les utilisateurs directs du système ainsi que les autres systèmes interagissant avec le système en question
3. La même personne physique peut jouer le rôle de plusieurs acteurs (vendeur, client). Plusieurs personnes peuvent jouer le même rôle et donc agir comme le même acteur (tous les clients). Le nom de l'acteur décrit le rôle joué par l'acteur.

Diagramme d'utilisation : l'acteur

Dans le cas du cabinet médical, nous avons 2 acteurs :

- la médecin
- le secrétaire

Diagramme d'utilisation : les cas d'utilisation

Un cas d'utilisation

- représenté graphiquement par une ellipse avec le nom du cas en dessous
- décrit une séquence d'action effectuée par le système pour livrer un résultat à l'acteur

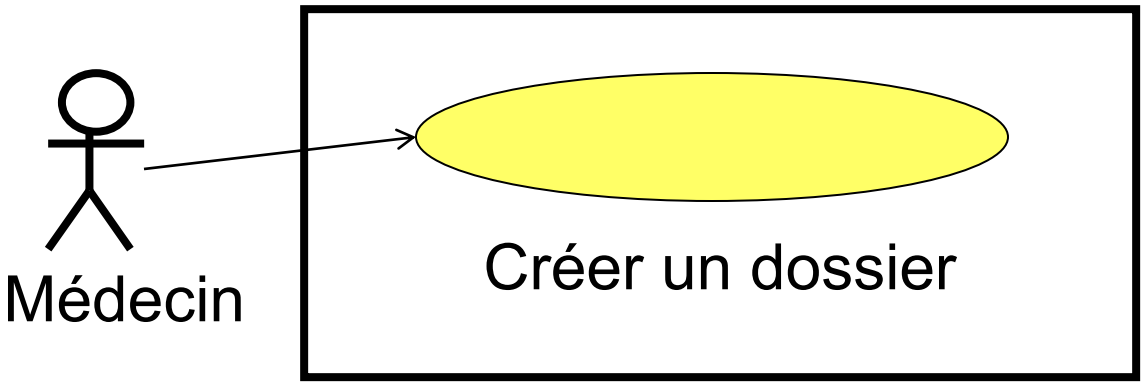


Diagramme d'utilisation : les cas d'utilisation

- Les cas d'utilisation se déterminent en observant et en précisant, acteur par acteur, les séquences d'interaction - les scénarios - du point de vue des l'utilisateur.
- La participation de l'acteur est signalée par un lien de communication entre l'acteur et le cas d'utilisation. Ce lien peut être orienté pour signaler l'initiateur de l'interaction.

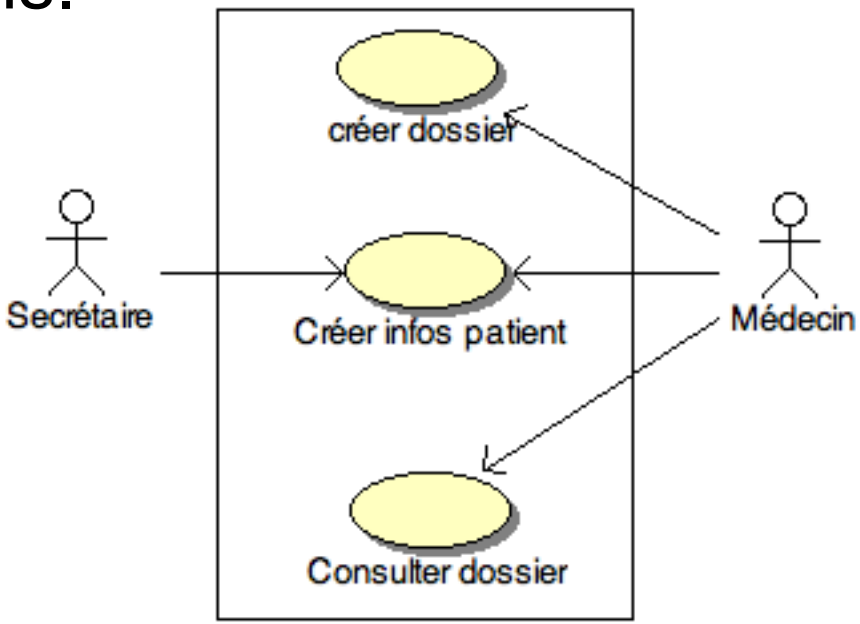
Diagramme d'utilisation : les cas d'utilisation

1. Les cas d'utilisations doivent être vus comme des classes dont les instances sont des *scénarios*. Chaque fois qu'un acteur interagit avec le système, le cas d'utilisation instancie un scénario; ce scénario correspond au flot de messages échangés par les objets durant cette interaction.
2. Les cas d'utilisation servent de fil conducteur pour l'ensemble du projet

Le modèle de cas d'utilisation

Le **modèle de cas d'utilisation**

- Comprend une collection de cas d'utilisation
- Caractérise le comportement de l'ensemble du système et des acteurs externes dans leurs interactions.

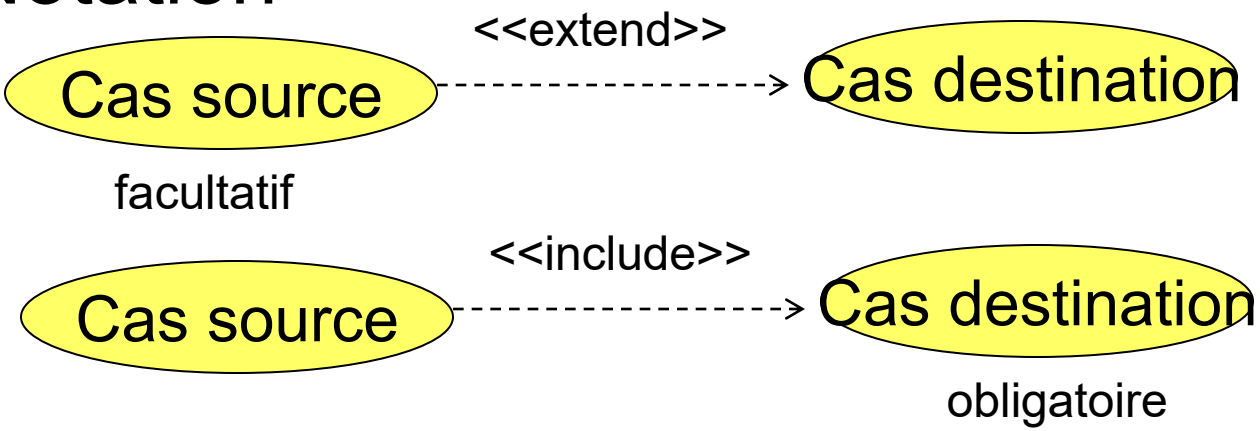


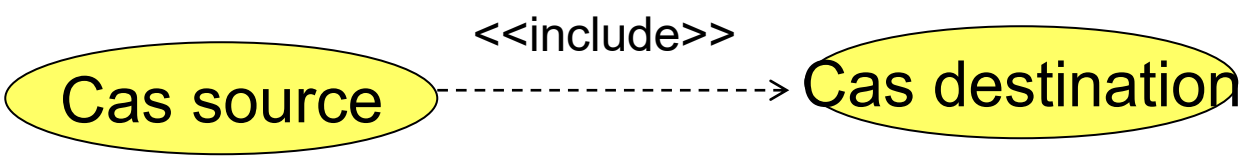
Raffinement des cas d'utilisation

Relations entre cas d'utilisation

- Extend : stéorotype <<extend>>
- Include : stéorotype <<include>>

Notation





La relation « *include* »

L'inclusion entre 2 cas d'utilisation signifie que toute instance du cas source comprend nécessairement le cas destination.

Quand l'utiliser ?

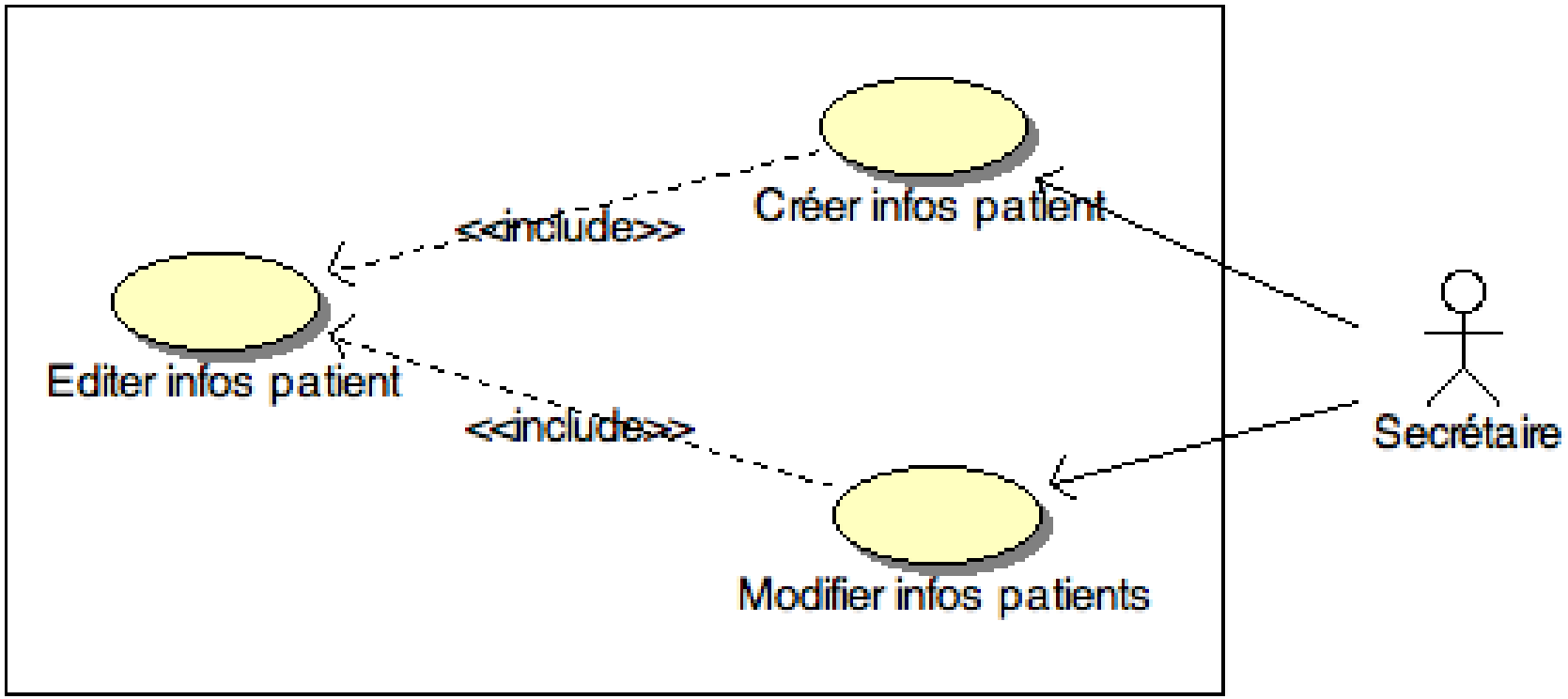
Lorsqu'on veut réutiliser un ensemble d'interactions communes dans plusieurs cas d'utilisation : on regroupe alors les interactions communes dans un cas d'utilisation (destination / obligatoire), lié aux cas d'utilisation qui l'emploient (les cas sources)

Avantage : réutilisation

Par exemple, s'authentifier (cas destination) pour utiliser les applications Web du CNRS (cas source) : janus, notilus, etamine, agate-tempo, sirhus, …

La relation « *include* »

Exemple



Cas source

<<extend>>

Cas destination

facultatif

La relation « *extend* »

L'extension entre 2 cas d'utilisation signifie que le cas source étend le comportement du cas destination

Quand l'utiliser ?

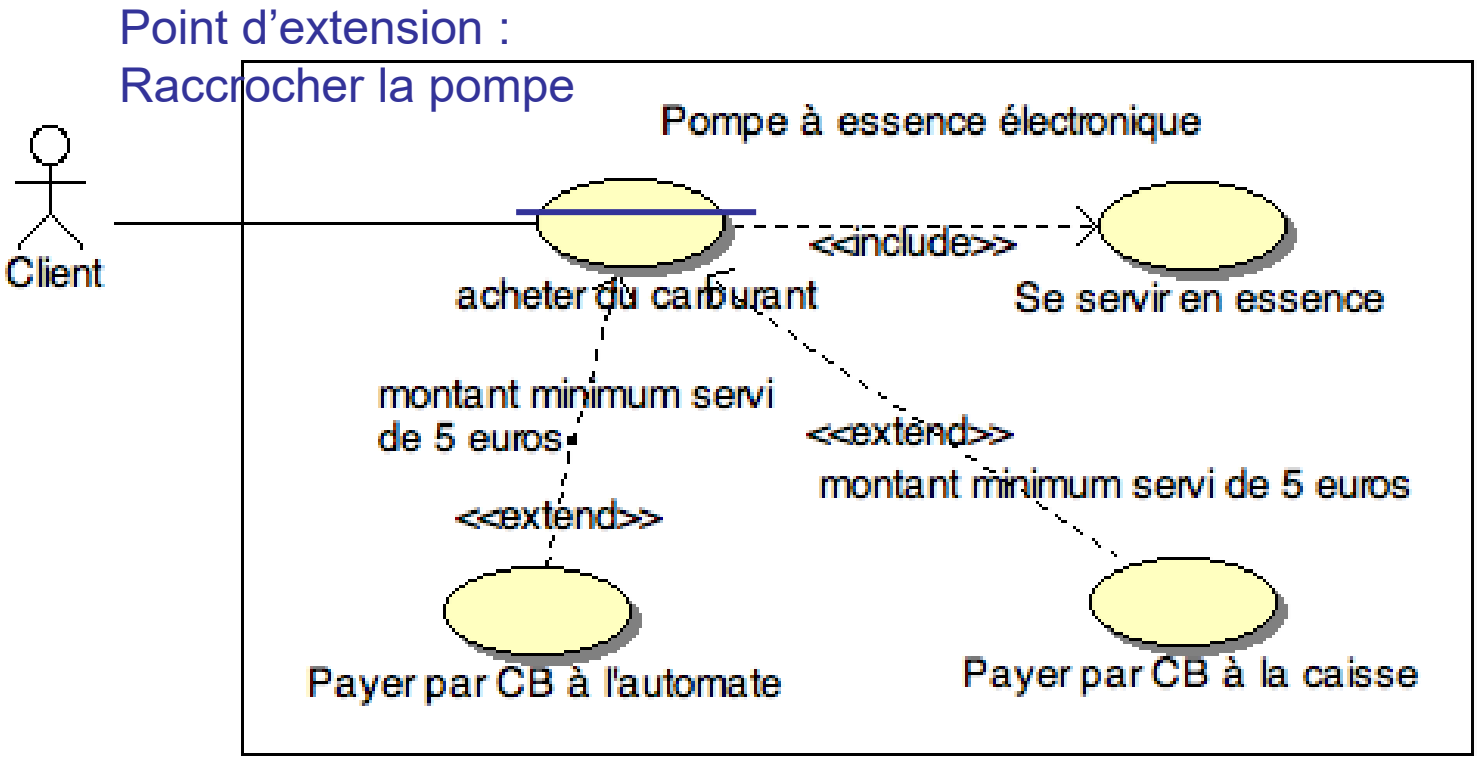
- Lorsque l'on veut enrichir un cas de base avec une possibilité de fonctionnement non décrit dans le cas source.
- Cette extension est décrite dans un cas à part, et n'a pas de sens SEULE (sans son cas source).
- Les 2 cas sont ensuite reliées par la relation d'extension

Les points d'extension montrent à quel moment survient l'extension. Une condition peut-être rajoutée à la relation d'extension pour la préciser.

Par exemple, utiliser une application Web sur un dispositif nomade comme un smartphone (cas source) est une possibilité qu'offrent certaines applications Web (cas destination)

La relation « *extend* »

Exemple : pompe à essence



Description des cas d'utilisation

La description peut-être sous une forme textuelle et peu structurée

Exemple :

« le médecin cherche le patient dans la liste des patients. Si son dossier existe, il l'édite, sinon il le crée. Le médecin introduit les informations sur les antécédents du patient, ses allergies, la liste des substances auxquelles il est allergique, la liste des traitements qu'il suit et la durée de ces traitements.

Etc. »

Description des cas d'utilisation

Elle peut être décomposée en précisant les interactions entre le système et l'acteur en distinguant le déroulement de base des déroulements alternatifs.

Créer ordonnance

Acteur	**Réponse du système**
1. Le médecin demande à créer une ordonnance 3. Le médecin choisit le patient souhaité 5. Le médecin sélectionne un médicament dans une liste …	2. Le système lui demande de choisir un patient 4. Le système édite une ordonnance vierge 6. Le système demande à saisir les doses et la fréquence des prises
Déroulement alternatif 5-6 Le médecin entre le nom du médicament, et la posologie	

Description des cas d'utilisation

La description d'un cas d'utilisation comprend les éléments suivants :

- Le *début* : « le cas débute quand X se produit »
- La *fin* : « le cas se termine quand Y s'est produit »
- *L'interaction* entre le cas et les acteurs
- Les échanges *d'informations* : par exemple, « l'utilisateur se connecte au système et donne son nom et son mot de passe »
- La *chronologie* et l'origine des informations
- Les *répétitions de comportement* qui peuvent être décrites au moyen de pseudo-code, avec des constructions du type :

```
Boucle
…faire quelque chose
Fin de boucle
```

```
Tant que
…faire quelque chose
Fin tant que
```

Description des cas d'utilisation

Format préconisé pour accompagner les diagrammes UML (page 127 de Rédiger des cas d'utilisation efficaces, Alistair Cockburn, *Ed. Eyrolles*, 2001)

Nom du cas d'utilisation	Le nom exprime l'objectif sous la forme d'une courte expression verbale exprimant une action	
Portée	Quel système est considéré comme une boîte noire en cours de conception	
Niveau	L'un des niveaux suivants : stratégique, objectif utilisateur, sous-fonction.	
Acteur primaire	Nom de rôle de l'acteur principal ou description	
Intervenants et intérêts	**Intervenant**	**Intérêt**
	Nom de l'intervenant	Intérêt de l'intervenant
Préconditions	Ce que doit être l'état du monde avant le début du cas d'utilisation	
Postconditions	État du monde si objectif rempli	
Description	**Étape**	**Action**
	1	Placez ici les étapes du scénario depuis le déclenchement jusqu'à la réalisation de l'objectif
	2	…
	3	…
Extensions	**Étape**	**Action de ramification**
	1a	Condition provocant la ramification : action ou nom du sous-cas d'utilisation
	…	…

Scénarios des cas d'utilisation

- Un cas d'utilisation
  - Est un ensemble complet d'actions (avec des évènements de début et de fin, les acteurs impliqués dans les actions) et de règles qui régissent l'enchaînement des actions.
  - Il définit les interactions entre le système et les acteurs
  - Il inclut le déroulement normal et tous les déroulements alternatifs
- Un scénario
  - Est un *déroulement spécifique* des évènements. Ce déroulement dépend des évènements à l'origine et à l'issue de chaque action spécifié dans le cas d'utilisation et dont dépend l'enchaînement des actions.
  - Il est possible de dériver plusieurs scénarios à partir d'un cas d'utilisation. Il suffit pour cela que l'enchaînement des actions ne soit pas une simple séquence.

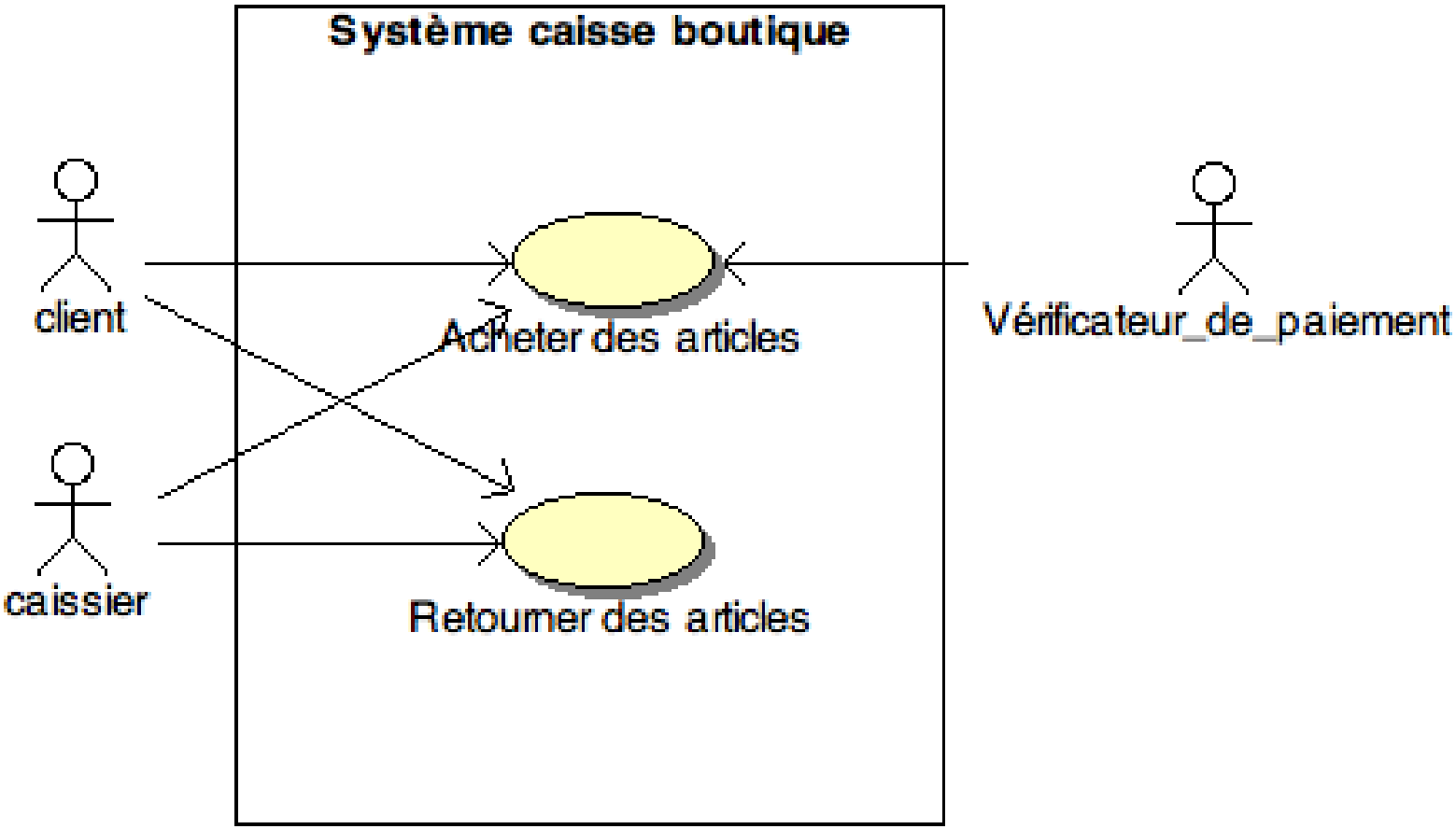
UN EXERCICE USE-CASE

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

Boutique de vêtements

1. Énumérez 3 acteurs impliqués dans la conception d'un système de gestion de caisse. Expliquez la pertinence de chaque acteur
2. Un cas d'usage est l'achat d'articles. Suivant le point de vue du consommateur, citez un autre cas d'utilisation, dont le niveau d'abstraction est identique. Résumez la finalité de chaque cas d'utilisation en une phrase
3. Tracez un diagramme de cas d'utilisation du système de gestion de la caisse d'une boutique
4. Rédigez un scénario standard pour chaque cas d'utilisation.
5. Rédigez un scénario d'exception pour chaque cas d'utilisation

Correction


Système caisse boutique
client
caissier
Acheter des articles
Retourner des articles
Vérificateur_de_paiement

Acheter des articles

Nom du cas d'utilisation	Acheter des articles	
Portée	Système caisse boutique	
Niveau	Objectif utilisateur	
Acteur primaire	Client	
Intervenants et intérêts	**Intervenant**	**Intérêt**
	Caissier	Les vêtement doivent être payés
	Vérificateur de paiement	Le compte du client doit être en solde positif
Préconditions	Les articles sont en boutique	
Postconditions	Le montant des articles est débité du compte du client au profit du compte du magasin	
Description	**Étape**	**Action**
	1	Le client présente des articles au comptoir
	2	Le caissier numérise chaque article du client
	3	Le caissier totalise le montant des articles, TVA comprise
	4	Le caissier demande le mode de paiement
	5	Le client donne une CB
	6	Le vérificateur confirme que la CB est acceptable
	7	Le client tape son code
	8	Le vérificateur valide le code
	9	Le caissier donne la facture au client
Extensions	**Étape**	**Action de ramification**
	2.a	Un article n'est pas reconnu et le caissier se renseigne dans le rayon pour avoir son prix

Retourner des articles

Nom du cas d'utilisation	Retourner des articles	
Portée	Système caisse boutique	
Niveau	Objectif utilisateur	
Acteur primaire	Client	
Intervenants et intérêts	**Intervenant**	**Intérêt**
	Caissier	Les vêtement doivent être remboursés au montant de l'achat
	Vérificateur de paiement	Le compte du client doit être en solde positif
Préconditions	Les articles ont été achetés dans cette boutique	
Postconditions	Le montant des articles est remboursé au client et les articles retournent en magasin.	
Description	**Étape**	**Action**
	1	Le client rapporte à la caisse des articles précédemment achetés
	2	Le caissier demande la facture du précédent achat
	3	Le client la lui donne.
	4	Le caissier vérifie la date et le mode de paiement sur la facture : espèces
	5	Le caissier accepte les articles et rembourse le client en espèces.
Extensions	**Étape**	**Action de ramification**
	3.a	Le client n'a pas la facture.
	3.b	Le caissier refuse de rembourser ou de faire un avoir.

Règles de mise en œuvre des cas d'utilisation

- Un cas d'utilisation décrit une fonctionnalité ou une motivation et aussi une interaction entre un acteur et un système sous la forme d'un flot d'évènements
- La description de l'interaction se concentre sur ce qui doit être fait
- Un cas d'utilisation doit être simple
- Il ne faut pas mélanger les cas d'utilisation
- Un cas d'utilisation doit éviter d'employer des expressions floues et imprécises.

Règles de mise en œuvre des cas d'utilisation

Il est primordial de trouver le *bon niveau d'abstraction*. Les réponses apportées aux 2 interrogations suivantes peuvent servir de gabarit :

- Est-il possible d'exécuter une activité donnée indépendamment des autres, ou faut-il toujours l'enchaîner avec une autre activité :
  - « passer un appel téléphonique »,
  - « composer un numéro de téléphone »
- Est-il judicieux de regrouper certaines activités en vue de les documenter, tester ou modifier ?

Construction des cas d'utilisation

- En règle générale, il n'y a qu'un acteur par cas d'utilisation
- Lors de la construction, il faut se demander :
  - Quelles sont les tâches de l'acteur
  - Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou lire ?
  - L'acteur devra-t-il informer le système des changements externes
  - Le système devra-t-il informer l'acteur des conditions internes
- Les cas d'utilisation peuvent :
  - être présentés au travers de vues multiples
  - être groupés selon leurs séquences de déclenchement types ou en fonction des différents points de vue

Processus d'élaboration des cas d'utilisation

- Définir un guide de style pour la rédaction
- Définir grossièrement les cas d'utilisation
- Approfondir la compréhension et la description d'un cas d'utilisation particulier. Identifier les scénarios
- Un scénario est un chemin particulier au travers de la description abstraite et générale fournie par le cas d'utilisation

DIAGRAMMES D'ACTIVITÉ

Analyser les processus

Lundi 9 sept. 2024 - 16H00-17H50

Diagramme activité / séquence (Christine Plumejeaud)

Les diagrammes d'activité

Le diagramme d'activité est utilisé pour

- Modéliser une tâche (dans la modélisation métier)
- Décrire une fonctionnalité du système modélisée par un diagramme d'utilisation
- Décrire la logique d'une *opération* ou d'un algorithme.
- Décrire les activités et leur enchaînement dans un processus

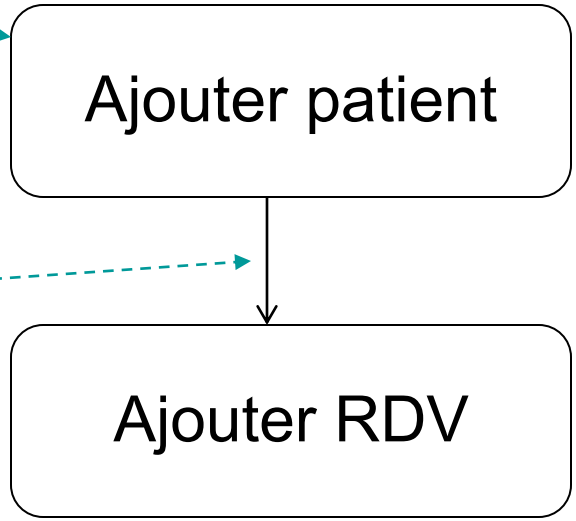
Le diagramme d'activités : concepts

Diagramme d'activités =

- Ensemble d'activités liées par :
  - Transitions (séquentielle)
  - Transitions alternatives (conditionnelle)
  - Synchronisation (disjonction et conjonction d'activités)
- Deux états : initial et final
- Couloirs d'activité : représente le responsable de l'activité.

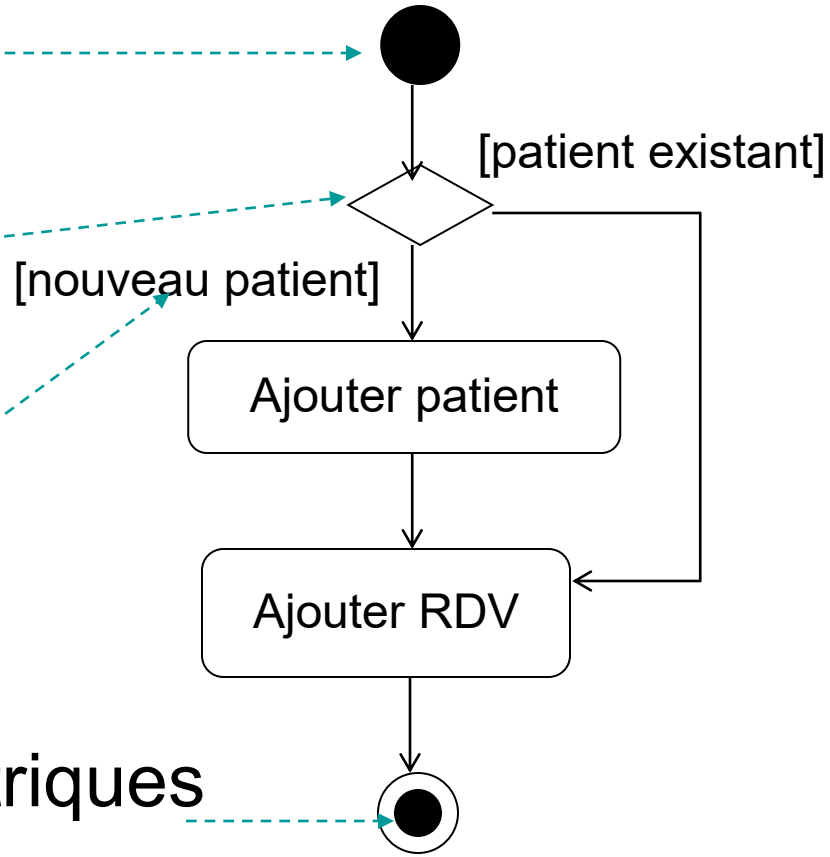
Notations du diagramme d'activité

- Activités
  - Un rectangle arrondi
  - Un nom significatif
- Transitions
  - flèche



Notations du diagramme d'activité

- Etat initial
  - Cercle noirci
- Point de décision
  - Losange
- Condition (garde)
  - Entre crochets
- Etat final
  - Deux cerce concentriques dont un noir

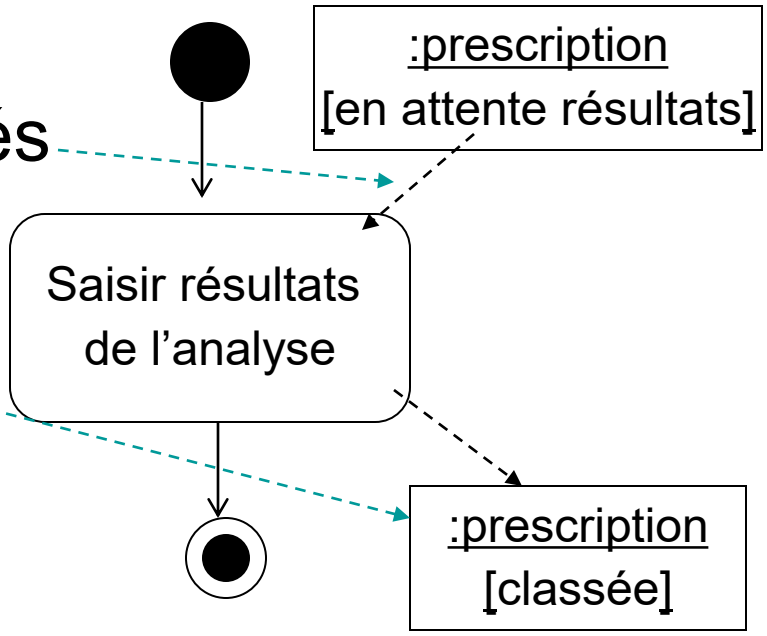


Notations du diagramme d'activité

- Flot d'objets
  - Des flèches en pointillés
- Objets
  - Rectangle
  - Le nom de l'objet

  est souligné
  - On peut montrer l'état

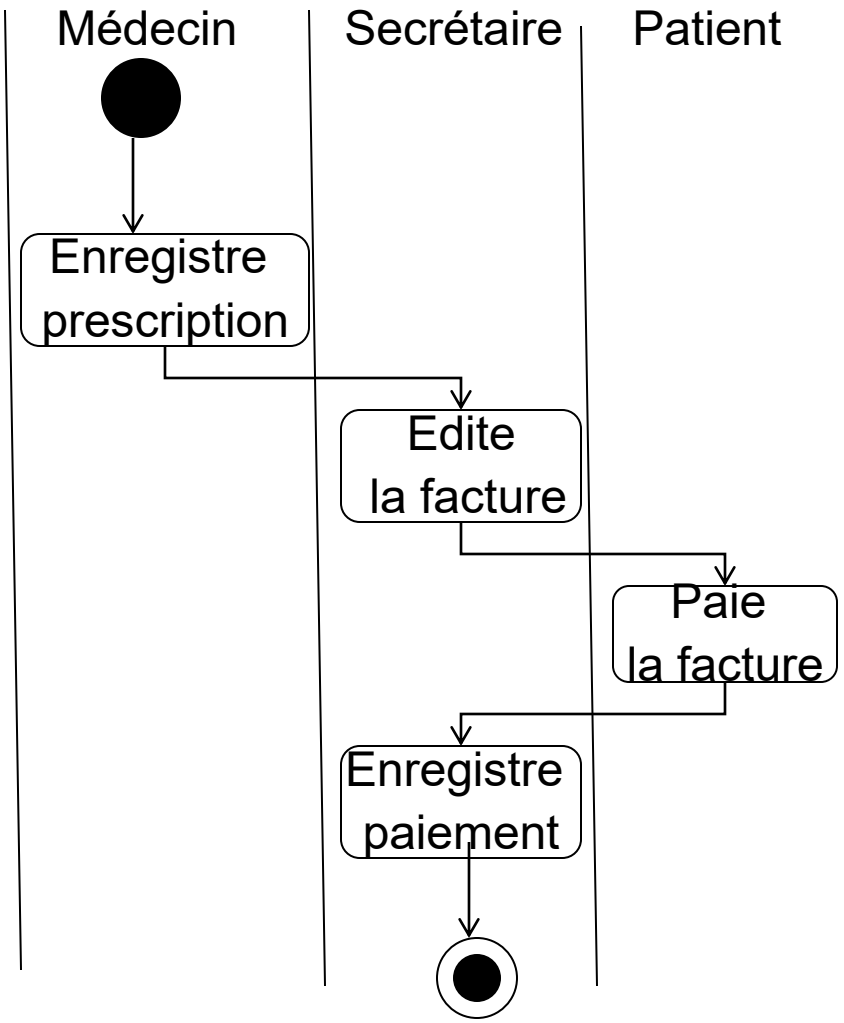
  de l'objet (noté entre crochets)



Notations du diagramme d'activité

1. Couloirs d'activité
   - Colonnes verticales
   - Etiquetés des noms

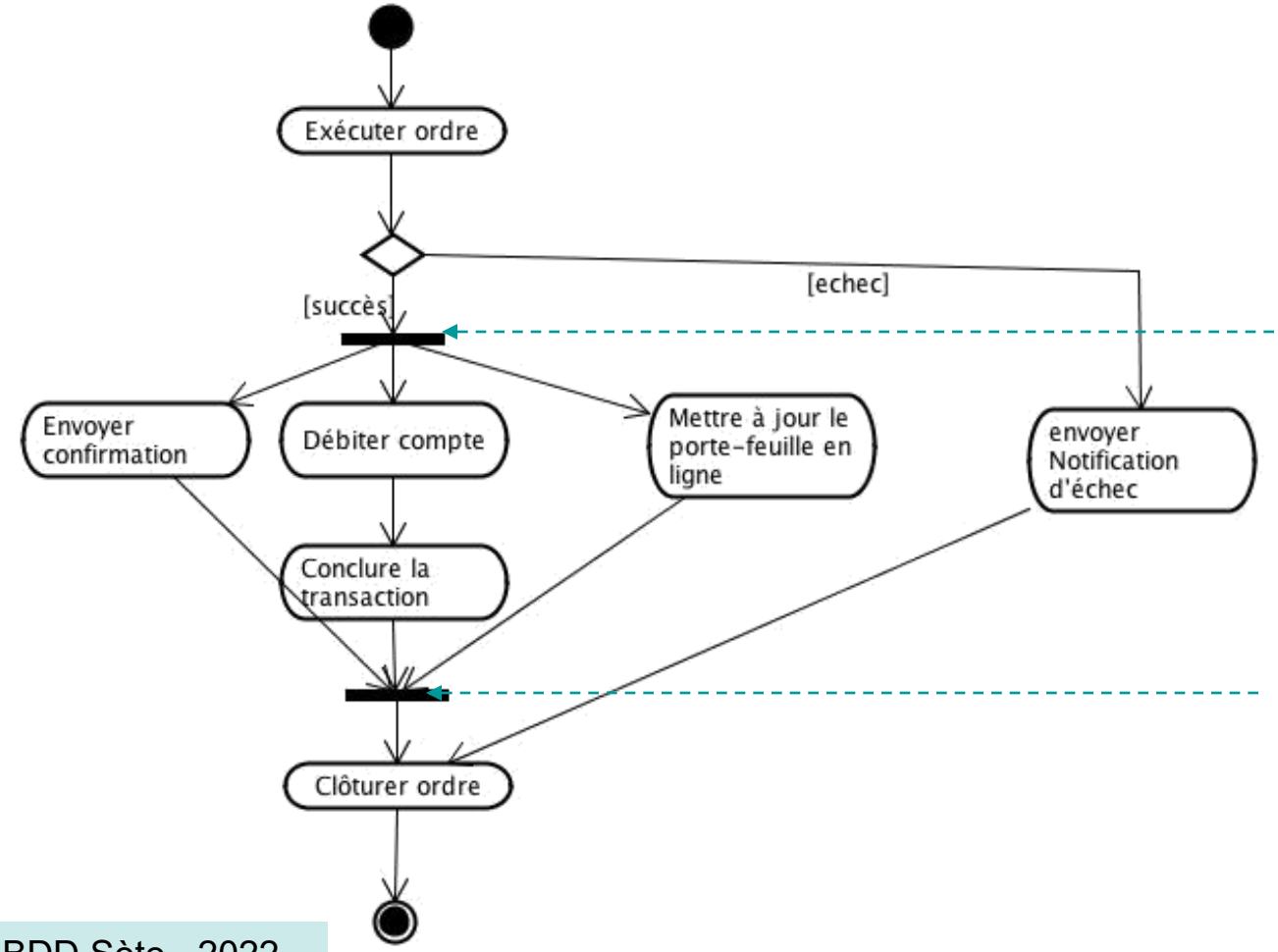
de personnes,
département, etc.,
responsables
de cette activité



Référence :
https://fr.acervolima.com/breve-note-sur-le-diagramme-d-activite-et-de-couloir/

Notations du diagramme d'activité

Fils d'exécutions concurrents

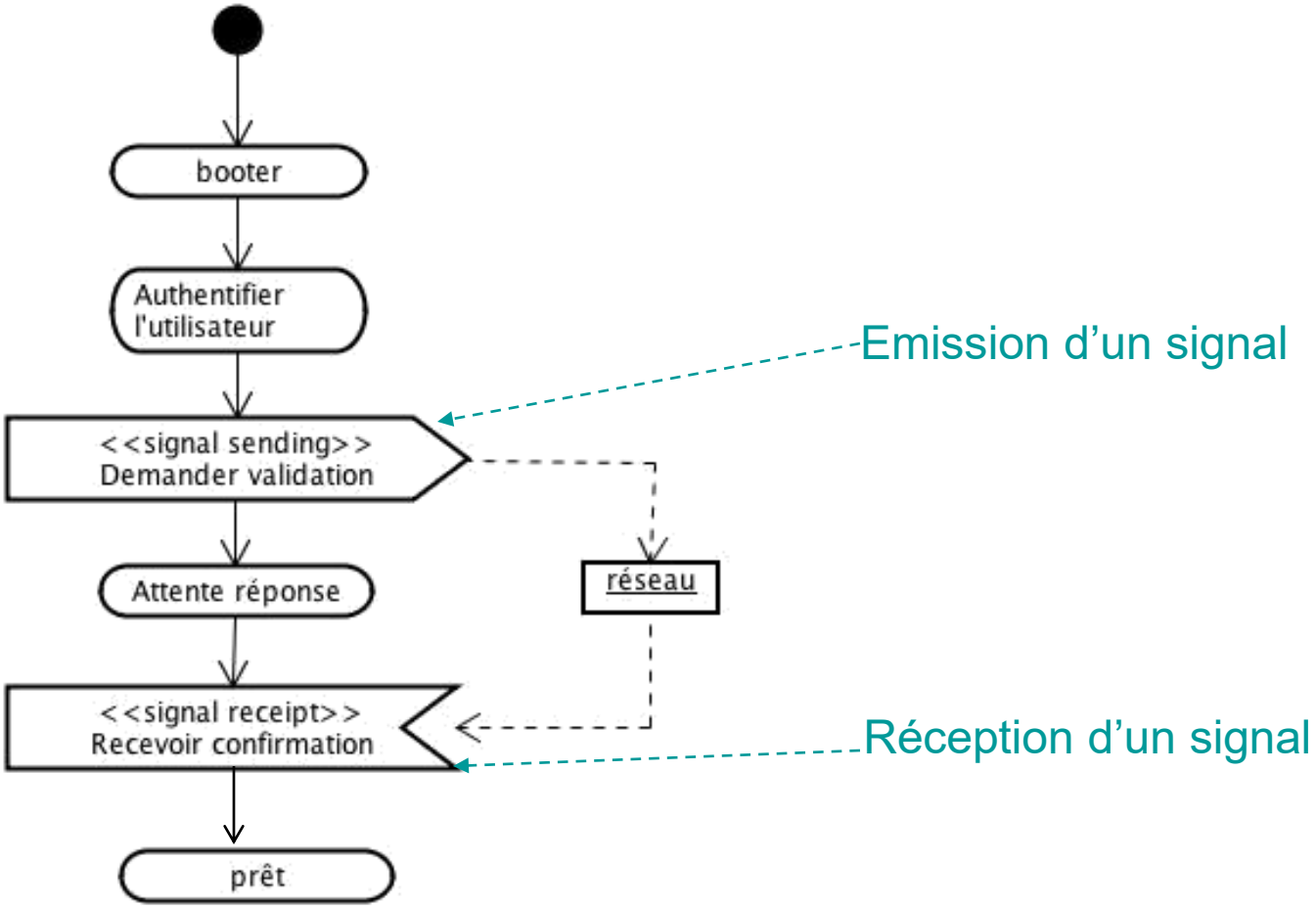


Barre de parallélisation

Barre de synchronisation

Diagramme d'activité

Emission et réception de signaux



Construire un diagramme d'activité

1. Identifier les activités
   - Qu'arrive-t-il lorsqu'un médecin souhaite saisir une prescription ?
     - Créer dossier médical du patient
     - Créer un dossier pour les antécédents
     - Saisir la prescription
2. Ordonner les activités en utilisant les transitions

Attention, le niveau de détails des activités d'un même diagramme doit être identique.

Diagramme d’activité : exemple

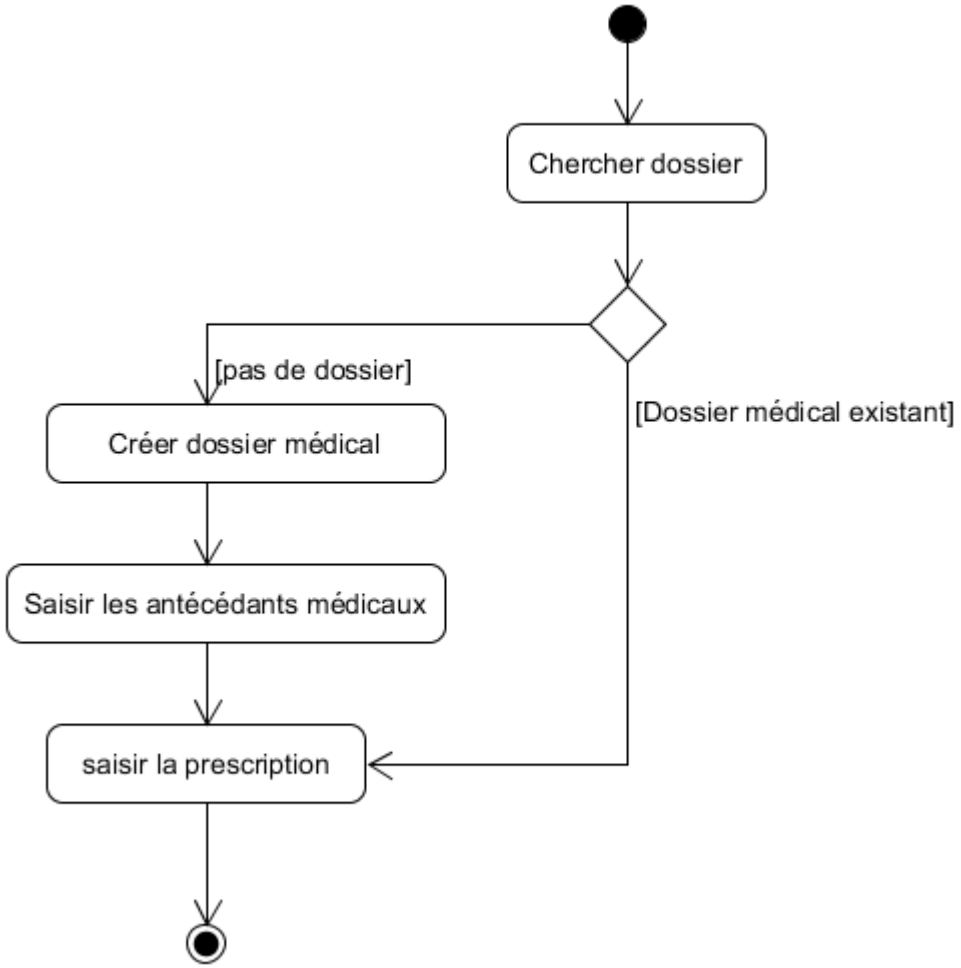


Diagramme d'activité : conseils

1. L'objectif ?
   - Décrire un workflow, des opérations, un algorithme, un flux d'objets.
2. Les critères de qualité ?
   - Un niveau de détails équilibré dans chaque branche
   - Les branchements sont tous documentés et couvrent le domaine. Usage du [sinon]
3. Le niveau de détail souhaité ?
   - Haut niveau d'abstraction ou plutôt détaillé ?

Bilan diagramme d'activité

Nous avons vu :

- L'utilité d'un diagramme d'activité
- Quelles sont les notations associées à un diagramme d'activité
- Comment construire un diagramme d'activité

UN EXERCICE ACTIVITÉ

1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

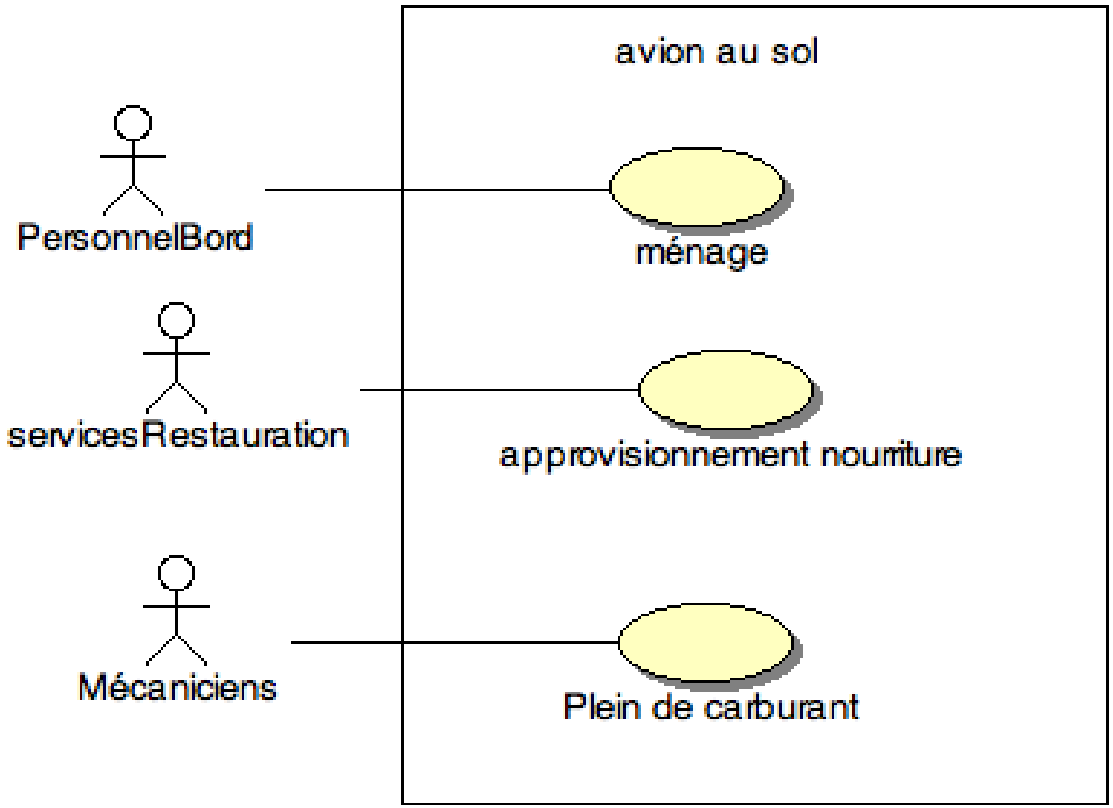
Entretien d'un avion au sol

Le personnel de bord doit nettoyer, le personnel au sol doit faire le plein de carburant, et le service de restauration doit réapprovisionner l'avion en nourriture et en boissons. L'appareil sera alors prêt pour un prochain vol.

- Faites le diagramme de cas d'utilisation autour du système avion, lors de son entretien au sol.
- Donner le diagramme d'activité correspondant

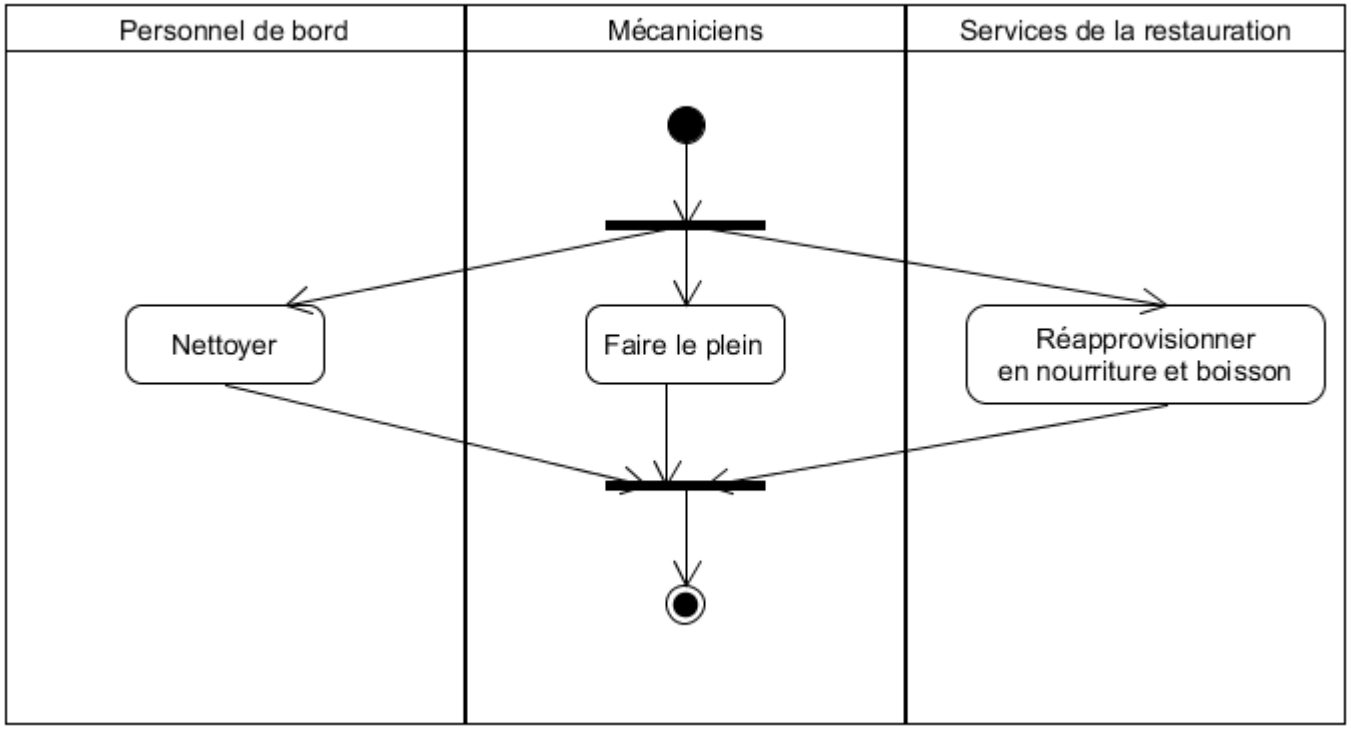
Correction

1. Diagramme de cas d'utilisation



Correction

2. Diagramme d'activité



INTERACTIONS ET SEQUENCES

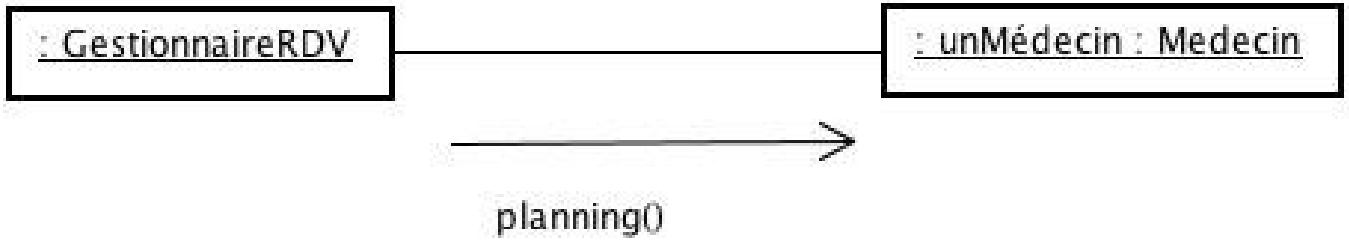
La communication dans un SI : déroulement temporel des actions.

Les interactions entre objets

1. L'envoi de message
   - Les objets communiquent par envoi de messages

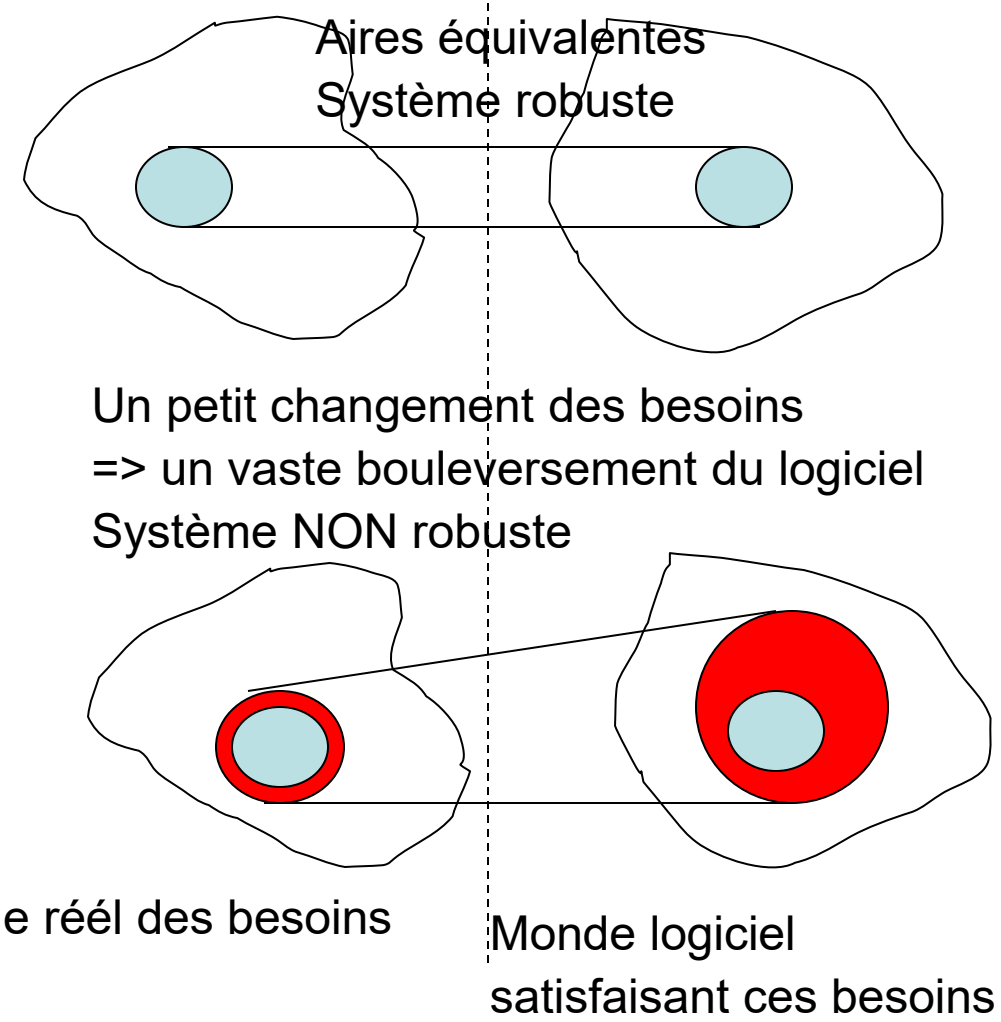
Envoyer le message **`fournir_planning()`** à un objet **`UnMédecin`**, doit utiliser la syntaxe suivante :

planning = UnMédecin.planning()



L’envoi de messages

- Les objets communiquent à travers l’envoi de messages
- L’objectif des diagrammes d’interaction est de préciser les responsabilités des objets pour minimiser l’effet de bord résultant des changements des besoins.



Interaction et collaboration

- On appelle *collaboration* l'ensemble formé par:
  - la structure des instances participant à un comportement
  - les liens qu'elles entretiennent.
- Une *interaction* est définie dans le contexte d'une collaboration.

  Elle spécifie le canevas de communication entre les rôles dans la collaboration. Elle contient un ensemble de *messages* partiellement ordonné. Chaque message spécifie une communication comme par exemple le signal à envoyer ou l'opération à invoquer, ainsi que les rôles joués par la source et la cible du message.

Diagramme de séquence

- Les interactions entre objets selon un point de vue *temporel*
- Le contexte des objets n'est pas représenté de façon explicite. La représentation se concentre sur l'expression des interactions
- Peuvent être utilisés à différentes phases du cycle de vie avec +/- de détails
- Peuvent être utilisés de façon :
  - Informelle : documentation des cas d'utilisation en illustrant les scénarios
  - Informatique : le concept de MESSAGE unifie toutes les formes de communication entre objets (appel de procédure, événement discret, signal entre flots d'exécution)

Diagramme de séquence

- La dimension verticale représente le *temps*
- Les *objets* impliqués sont disposés horizontalement. Ils sont représentés par des *lignes de vie* verticales
- Les *messages* sont représentés par des flèches pleines horizontales
- L'exécution d'une opération est matérialisée par une *activation*

Diagramme de séquence : notation

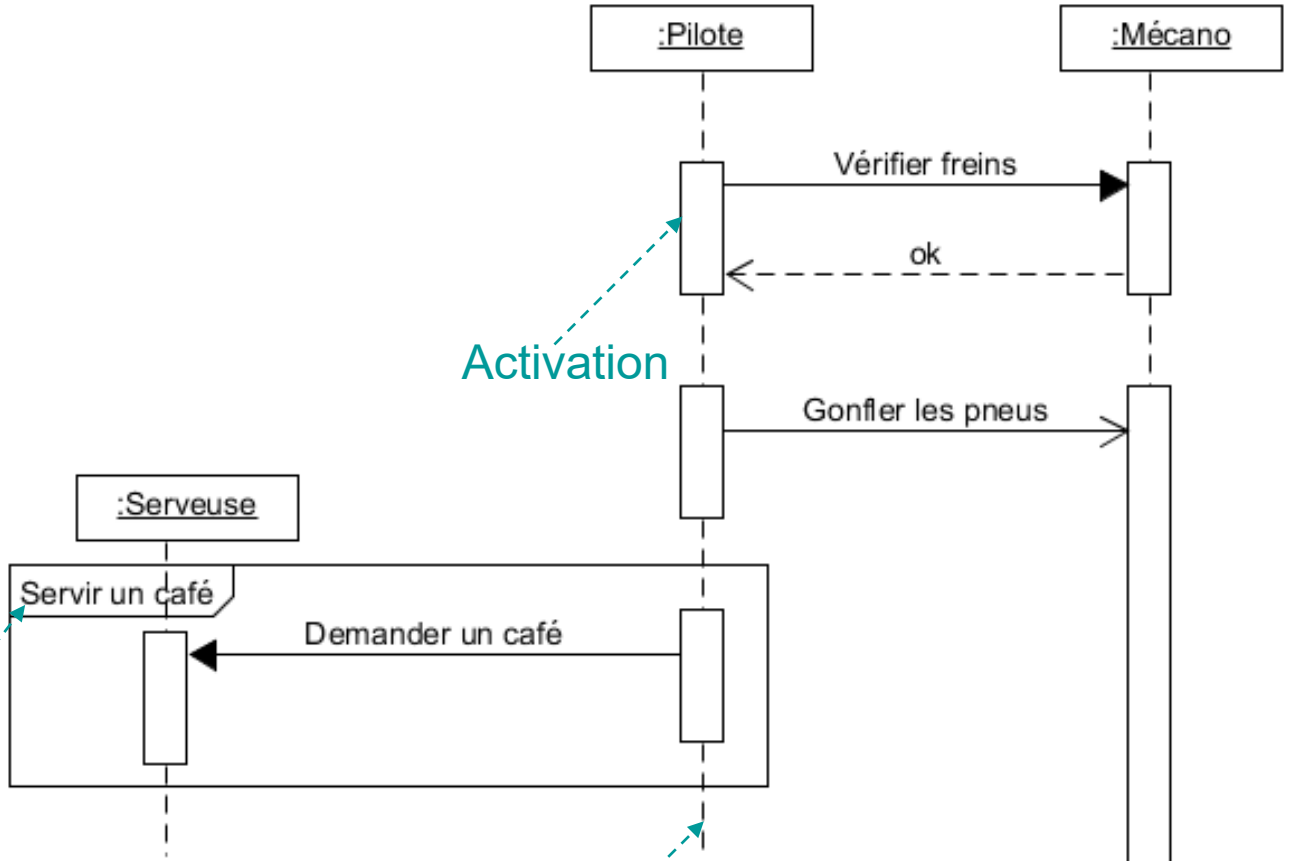


Diagramme de séquence : création/destruction d'objets

1. Création : pointer le message de création sur le rectangle symbolisant l'objet créé.
2. Destruction : fin de la ligne de vie, et une croix X.

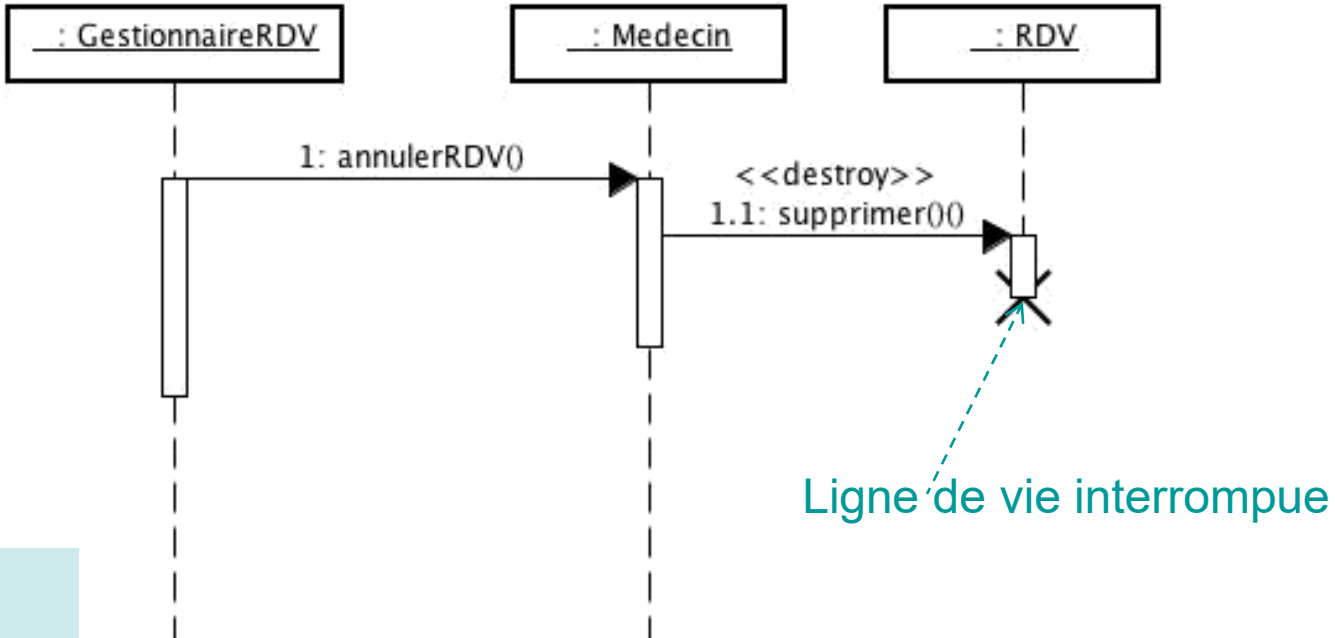


Diagramme de séquence : les différents types de message

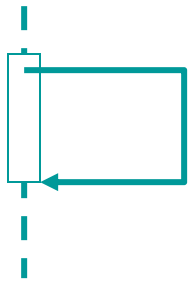
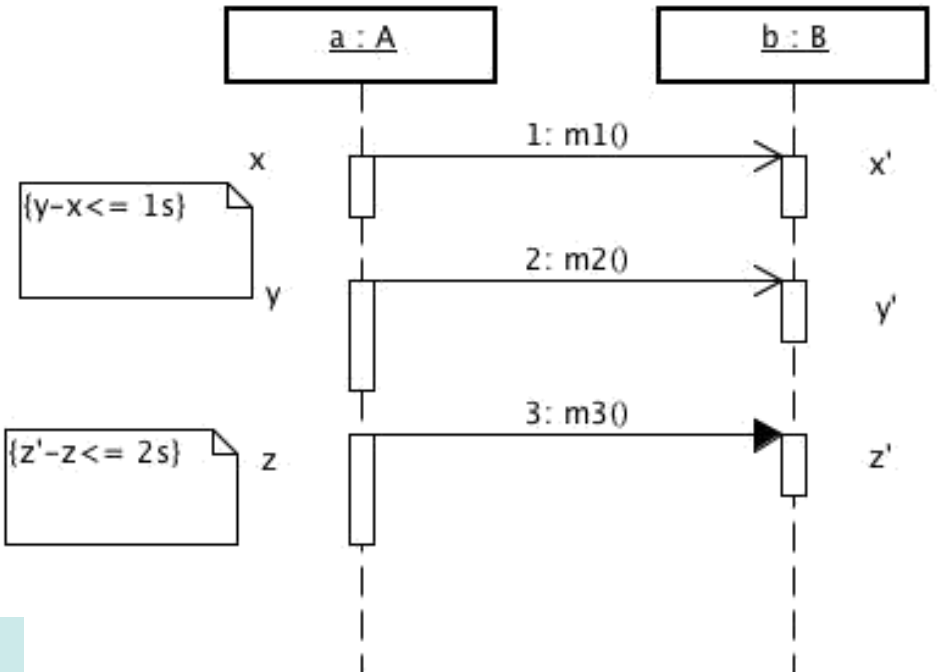
1. Message synchrone : l'appelant est bloqué en attente du retour de message 
2. Message asynchrone : l'appelant continu son flot d'exécution après l'émission. La réception se fait ultérieurement. 
3. Retour de procédure : le retours des messages sont le plus souvent implicite, mais parfois ils sont indiqués (cas asynchrone)
4. Message réflexif : les objets peuvent produire des messages pour eux-mêmes. 

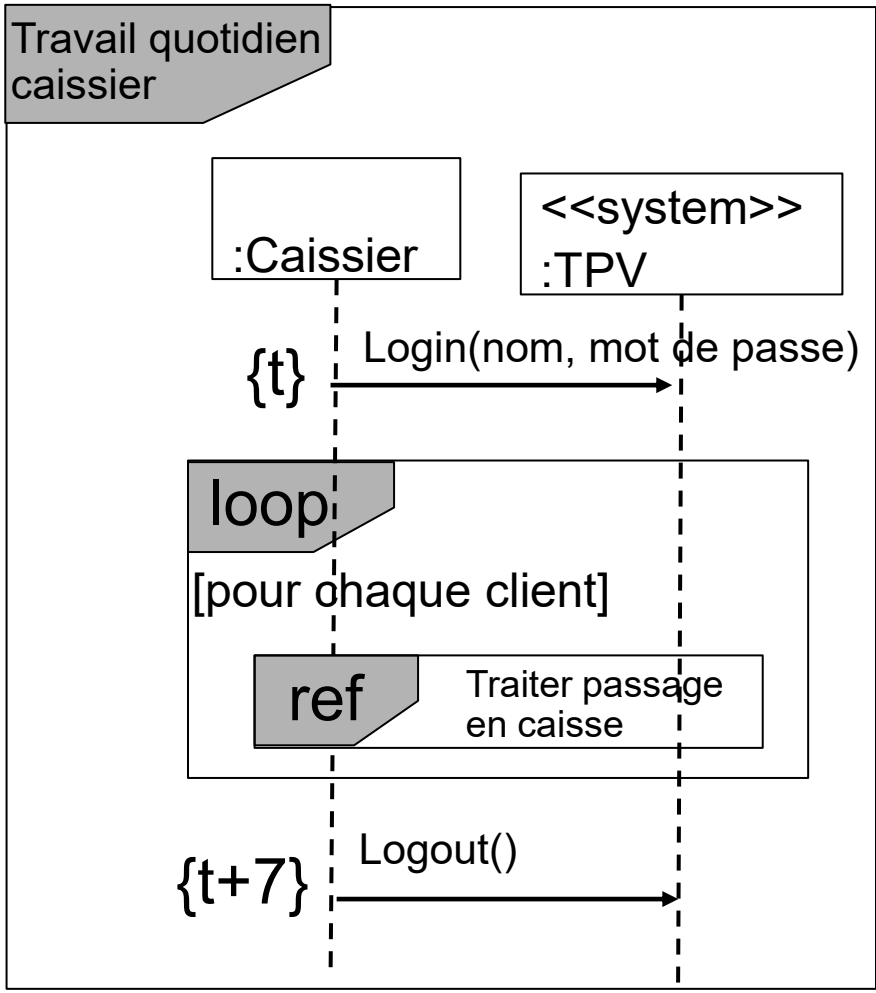
Diagramme de séquence : les contraintes temporelles

- Instant d'émission x
- Instant de réception x'
  - Exemples :
    - Il s'écoule moins d'une seconde entre l'émission de m1 et m2
    - Il s'écoule au moins 2 secondes entre l'envoi et la réception de m3

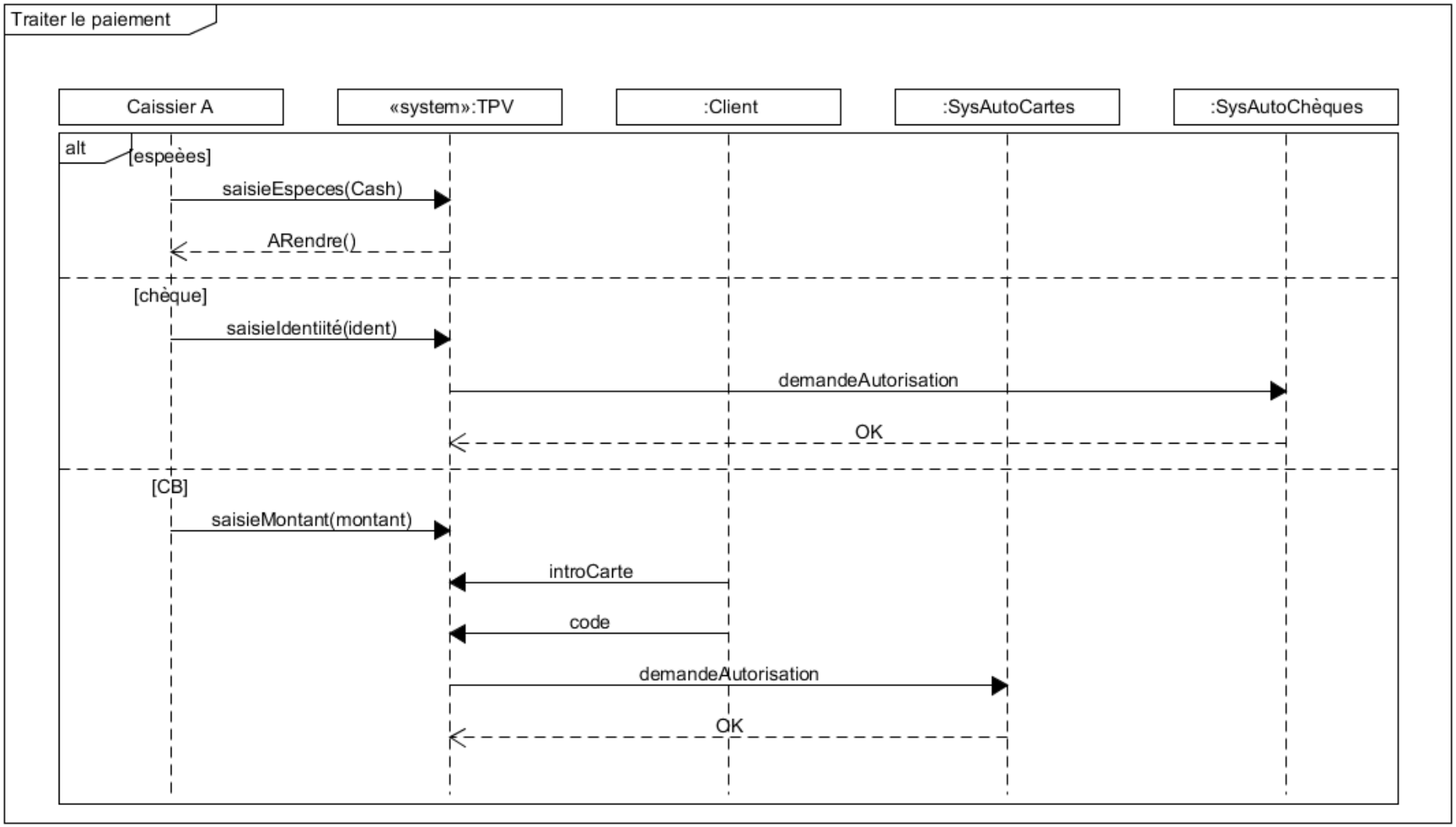


Les fragments : pour gérer la complexité

1. LOOP
   - Enchaînement qui se répète
2. OPT
   - Une action qui se fait sous condition
3. REF
   - Référence à un autre diagramme de séquence
4. ALT
   - 2 ou plus alternatives qui se réalisent
5. PAR
   - Plusieurs actions qui se déroulent en parallèle



Représentation des ALT ou PAR



Les diagrammes de séquence : conseils pour la description des besoins

1. Un scénario par cas d'utilisation : évitez les branchements conditionnels
2. Subdiviser les interactions complexes : décrivez dans un second diagramme une sous-tâche.
3. Un diagramme de séquence par condition d'erreur possible.

Un exercice Séquence

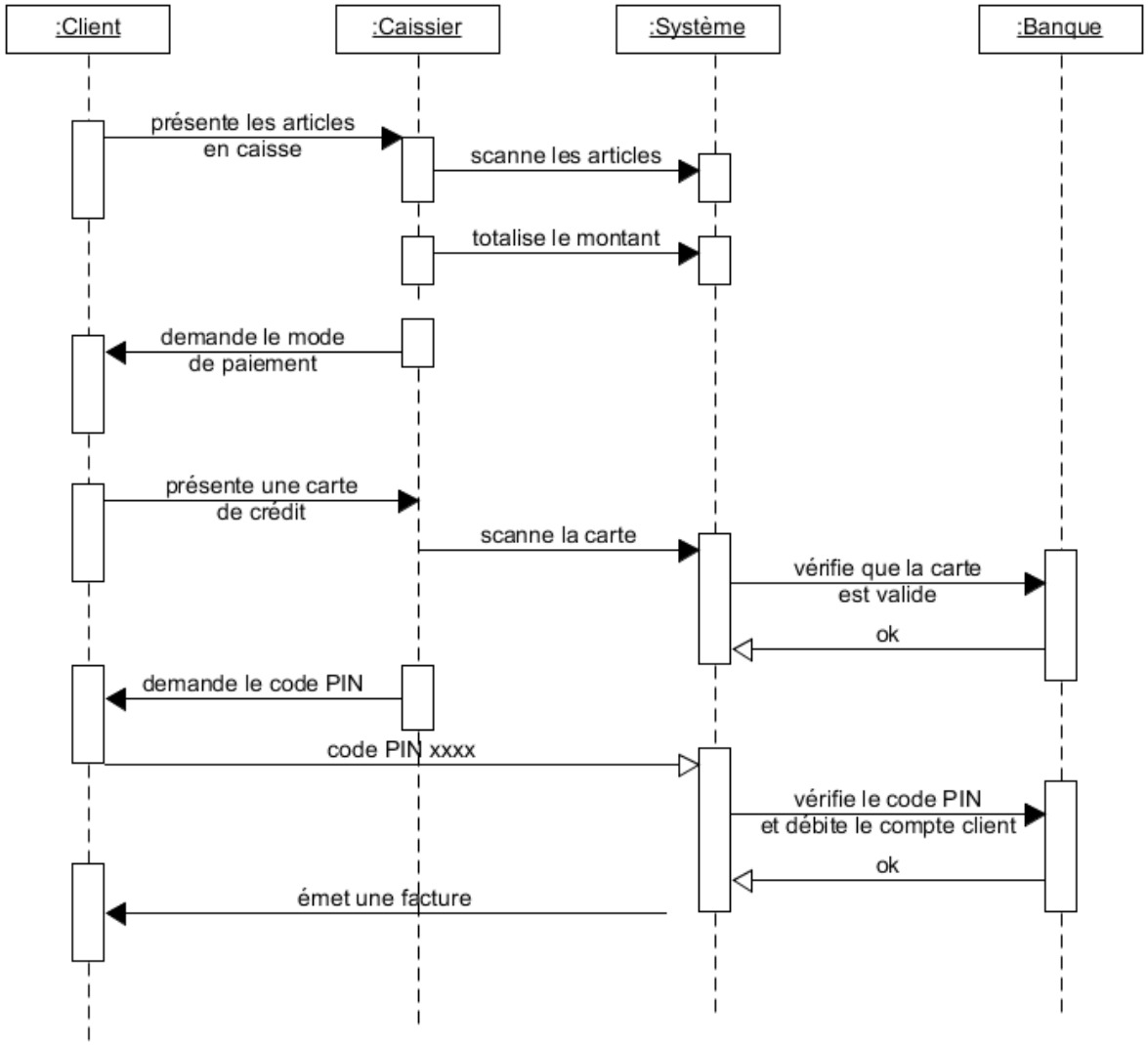
1. Lire l'énoncé : 5 min
2. Elaborer le diagramme sur papier : 10 min
3. Discuter les corrections et propositions : 10 min
4. Rédiger la correction dans un logiciel : 5 min

Boutique de vêtements

1. Dessiner une séquence d’achat standard
2. Dessiner un retour pour remboursement

Boutique de vêtement (suite)

1. Séquence d'achat standard



Boutique de vêtement (suite)

2. Retour de vêtements en boutique

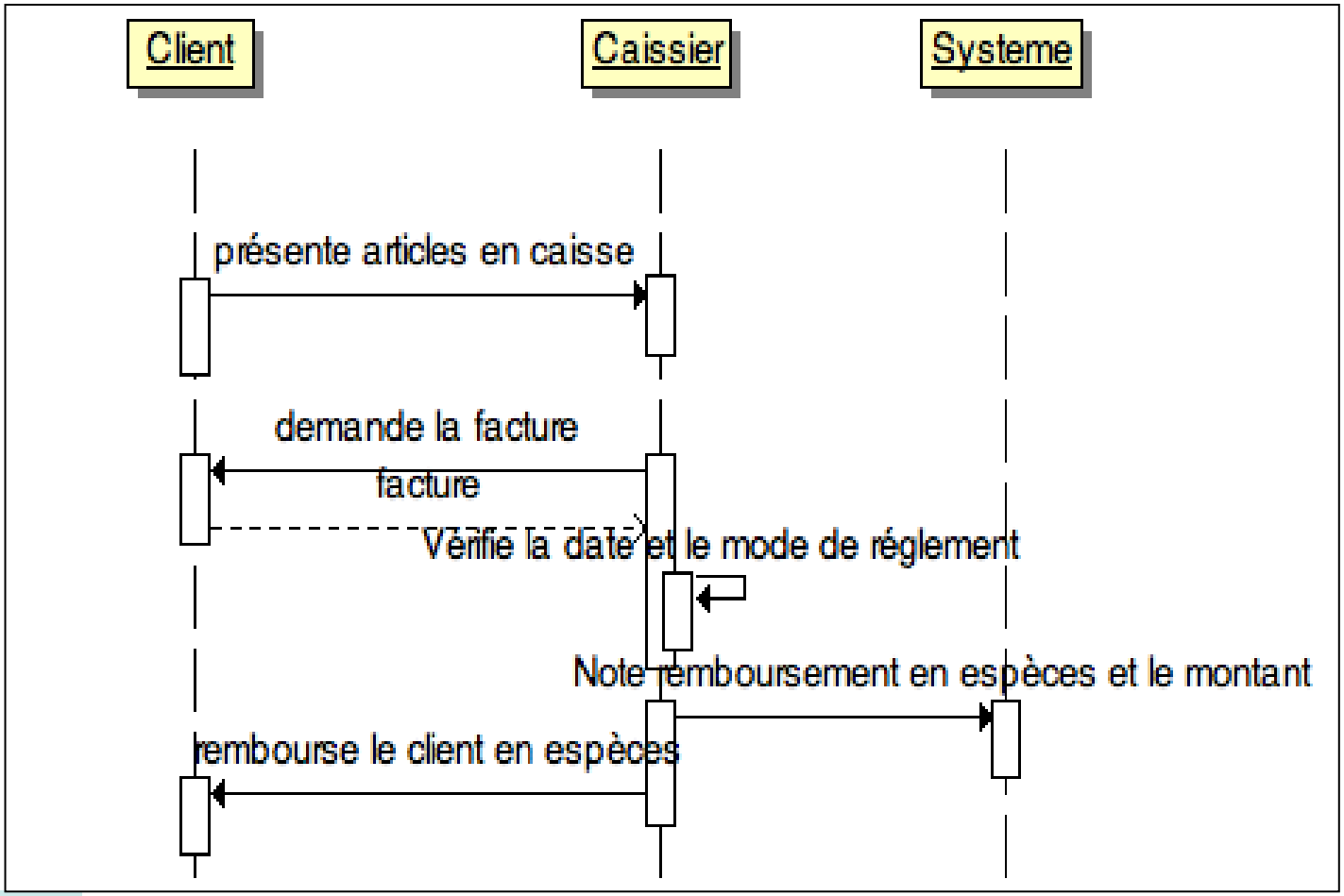


Diagramme de séquence : les branchements conditionnels

A émet un message

soit vers B si la condition 1 est remplie,

soit vers C si la condition 2 est remplie

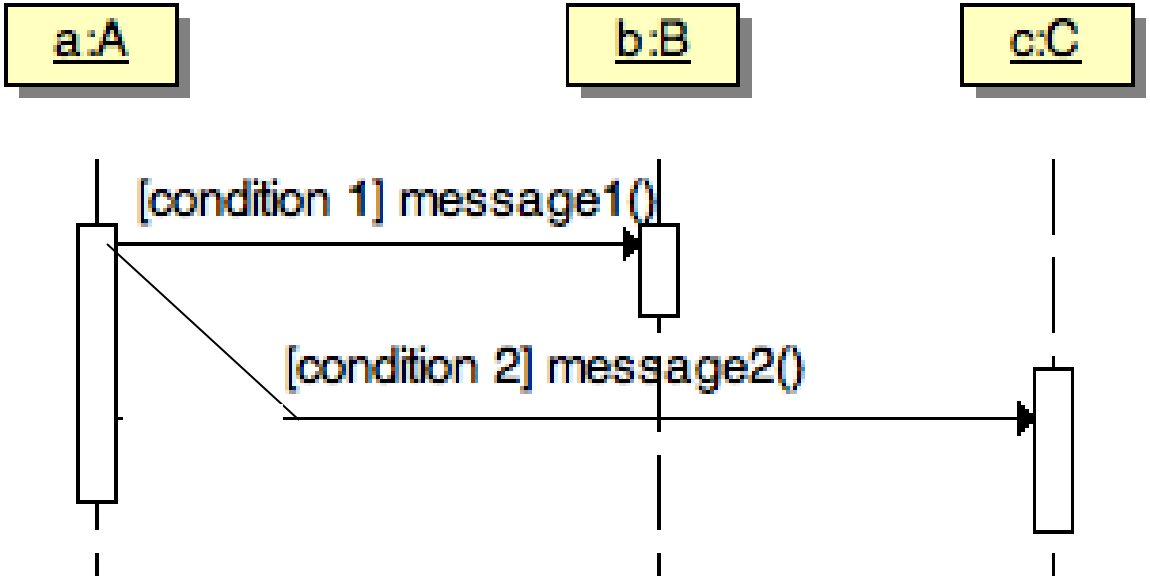


Diagramme de séquence : gérer la complexité

Les diagrammes complexes peuvent être décomposés en plusieurs diagrammes plus simples en précisant convenablement cette décompositions par des annotations.

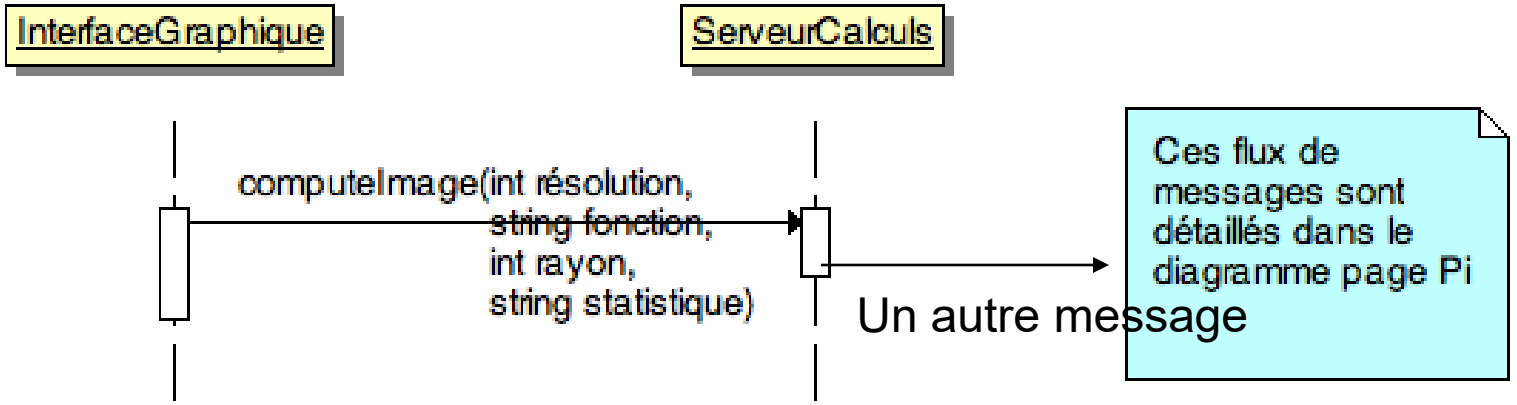


Diagramme de séquence

Exemples de messages synchrones/asynchrones