Cycle de vies

Processus à étapes avancées selon une certaine logique d'enchainement

- Cycle de développement : Analyse, conception, codage, tests
- Cycle de maintenance : corrective, adaptive et evolutive

Modèles

linéaire	hybride	Evolutifs
en V / en Cascade (17 / 7)	→ prototypage :	ou prototypage / increment (13 / 21) spirale (24)
besoins stables + claire = bonne ration coût / qualité	sous systèmes haut risques → linéaires : besoins stables clairs	erreur definition des besoins detectées tôt } Avantage coûteux } inconvenient

Avantages

Inconvenient : erreurs definition des besoins détectées tardivement

type de prototypage

- Maquette (Prototype rapide ou jetable) : sous ensemble de fonctions → squelette du logiciel qui n'est créée que dans un but et dans une phase particulière du dev
- (Non jetable) Evolutif : conçu au tout au long du cycle du développement améliorer et compléter pour obtenir le logiciel final

Analyse et definitions des besoins :

Processus visant à établir quelles fonctionalités le système doit fournir et les contraintes auxquelles il sera soumis

- → rassembler et analyser toutes les infos le problème
- → produire une definition comprehensible

Distinction :
- → but : caractéristique générale (critères subjectifs)
- → besoins : associé a cebut (: commande user accessible du menu)

→ Analyste

5 Cahier de charge : document contenant des indications concernant les phase suivantes du cycle de vie. Ce n'est pas un document de conception, il doit decrire ce que le système doit faire, sans specifier comment.

besoins : complet et coherent

Propriétés : precise "

info non pertinente, superflue / incomplète → éliminée

Structure C.C.:

- Introduction
- Matériel
- Modèle Conceptuel
- Besoin fonctionnels
- besoins en matière BDD
- besoins ⊤ fonctionnels
- info maintenances
- Glossaire
- index
- Remarque

Modèle conceptuels du système → approche Cartésienne (Hiérarchique descendante) 12

Définition besoins :
- → langage naturel 16
- → langage structuré 17 DS / PSA , SADT , RSL etUML
- → langage de spécification formel 19

réalisé valide — complet cohérent

Besoins non fonctionnels : Contraintes , restrictions / vitesse d'exec , Mémoire (Matériels)

Anticipable

Pas de conflit entre besoins (Analyseur de notoriété)

Simulateur : Simul l'exécution / Model checkers / theorem Provers

↓

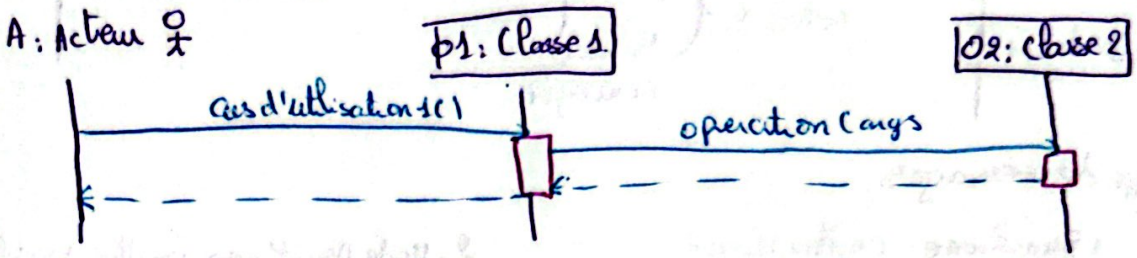
Not good → Prototypage

Diagrammes de Séquences

Elements: Acteur / objets (instances) / Message (Cas d'utilisation, appels d'opérations)

Représentation chronologiques des echanges de message avec le système ou au sein du système

↑ Vie" de chaque entité représenté verticalement, Echange des messages horizontalement



Utilisation en phase de conception:

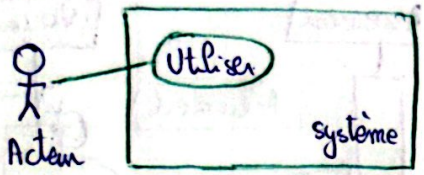


Diagramme de cas d'utilisation.

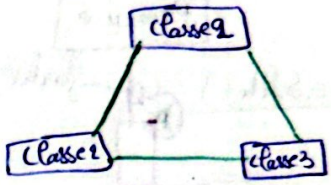
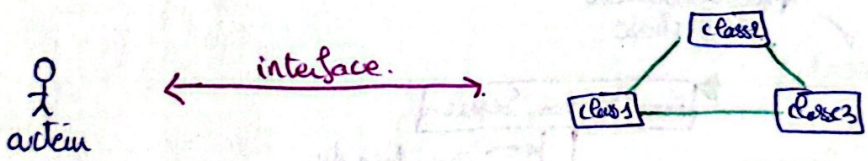
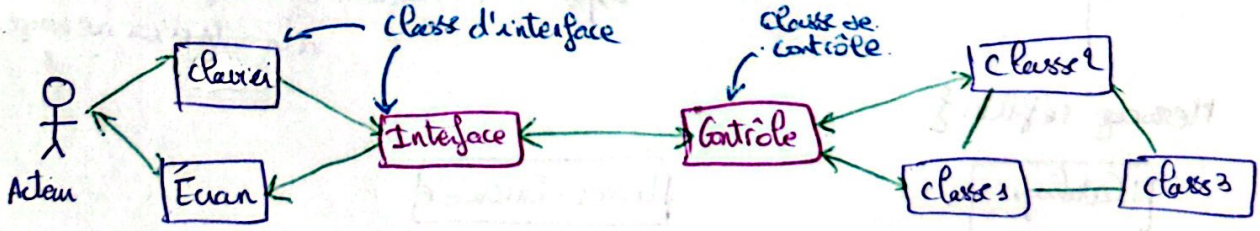


Diagramme de classes du système

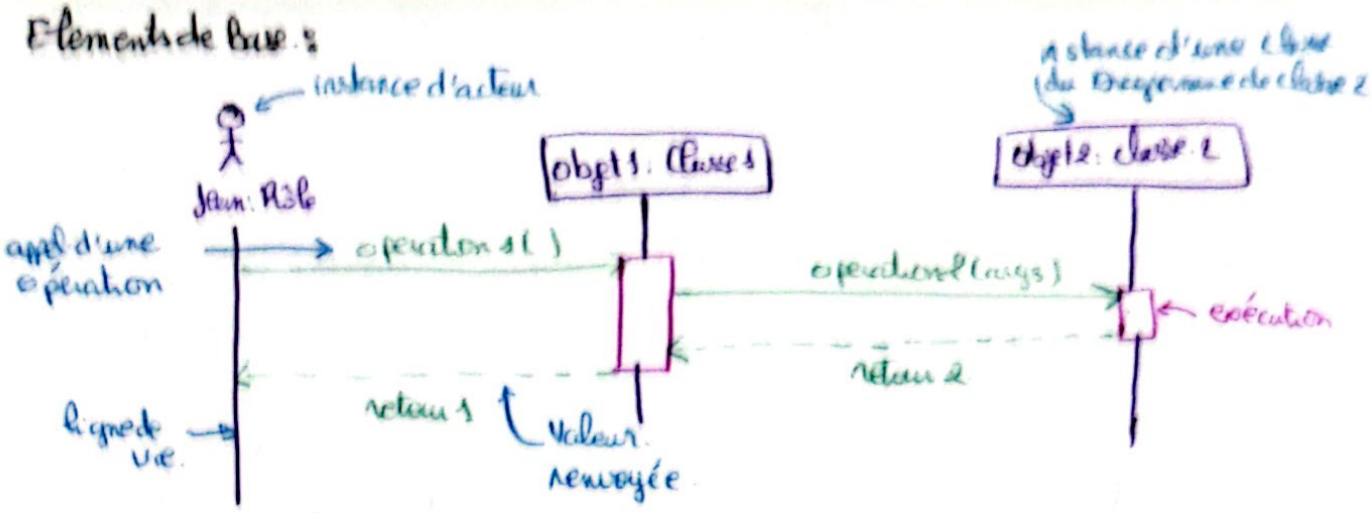
Acteur ne peut pas communiquer directement avec des objets



Solution: Création de classe de contrôle et d'interface qui gèrent les interaction avec les acteurs encapsulent le résultat des opérations

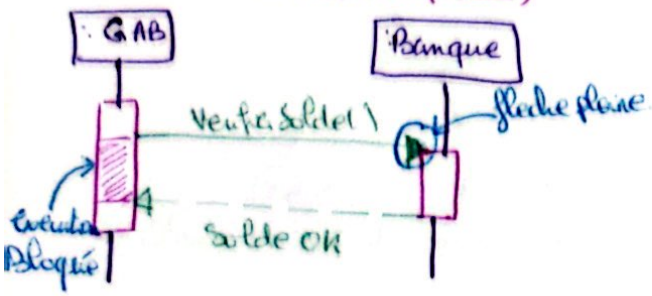


Elements de base :

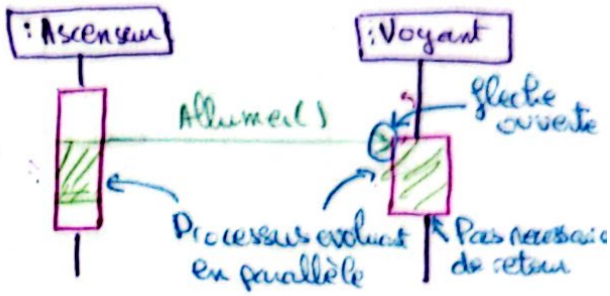


Type de message :

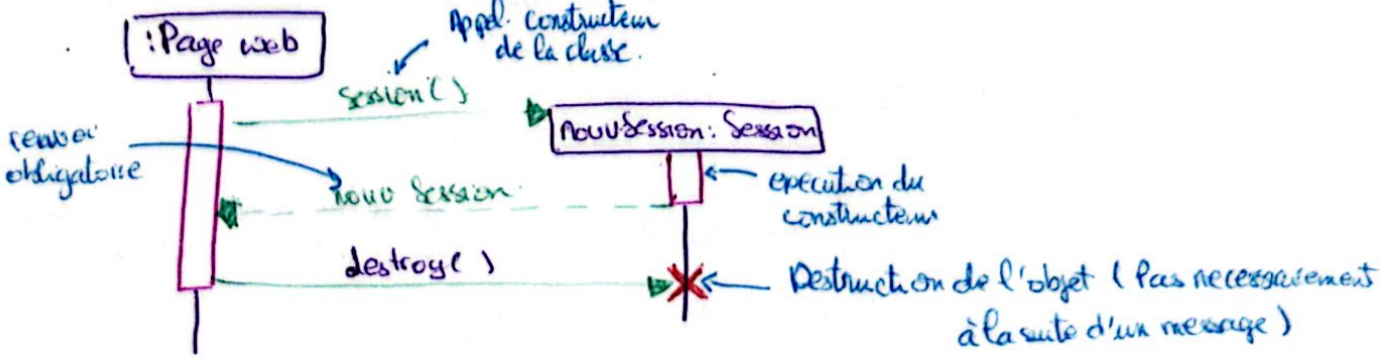
1 Synchrone : emetteur bloqué en attente du retour (Séquentiel)



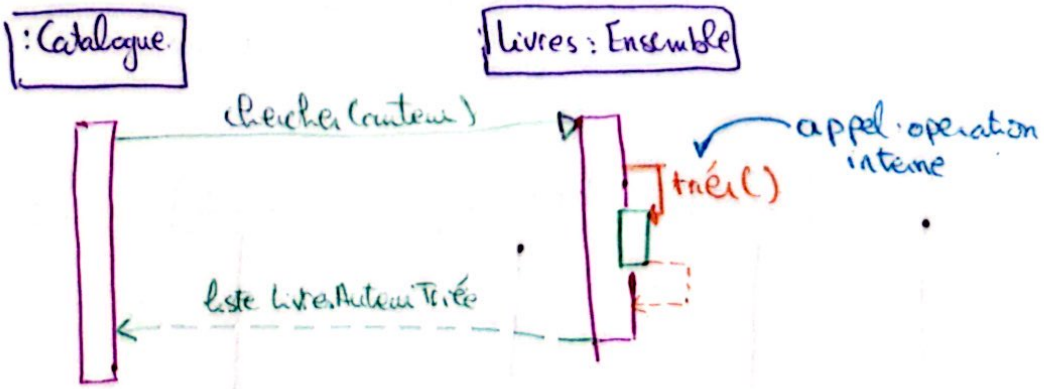
2 - Mode Asynchrone emetteur non bloqué, continue son execution.



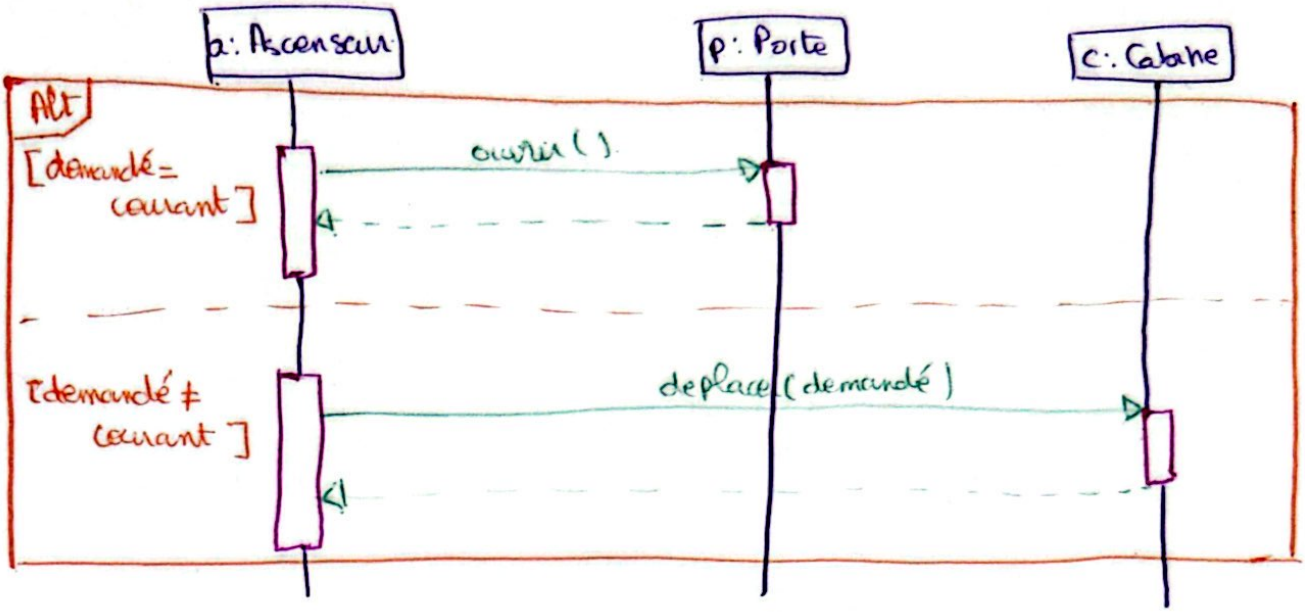
Gestion et destruction d'objet :



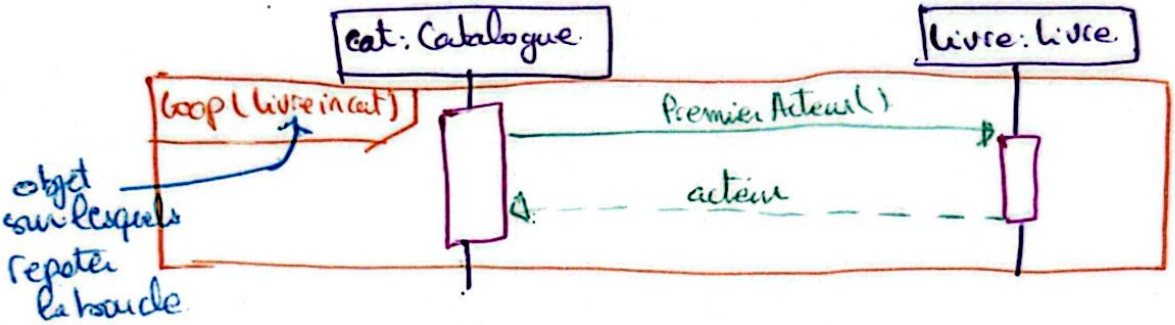
Message réflexif



Condition :



Boucle :



Référence :