IUT Nantes
Pôle Sciences et technologie
Nantes Université

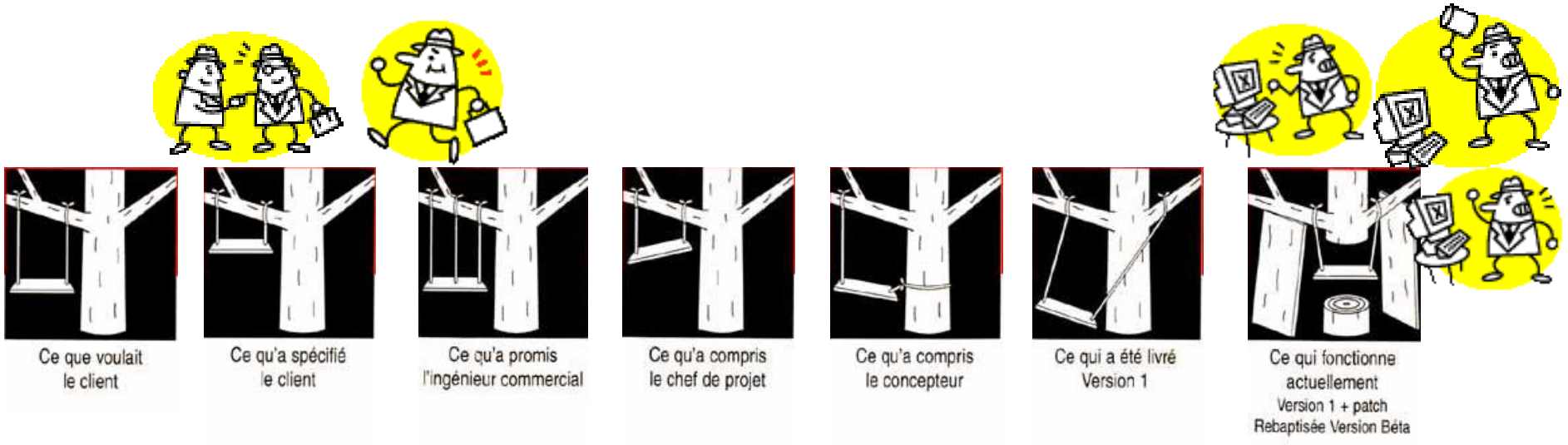
R3.03 – Analyse

2_ DÉVELOPPEMENT LOGICIEL

Dalila TAMZALIT

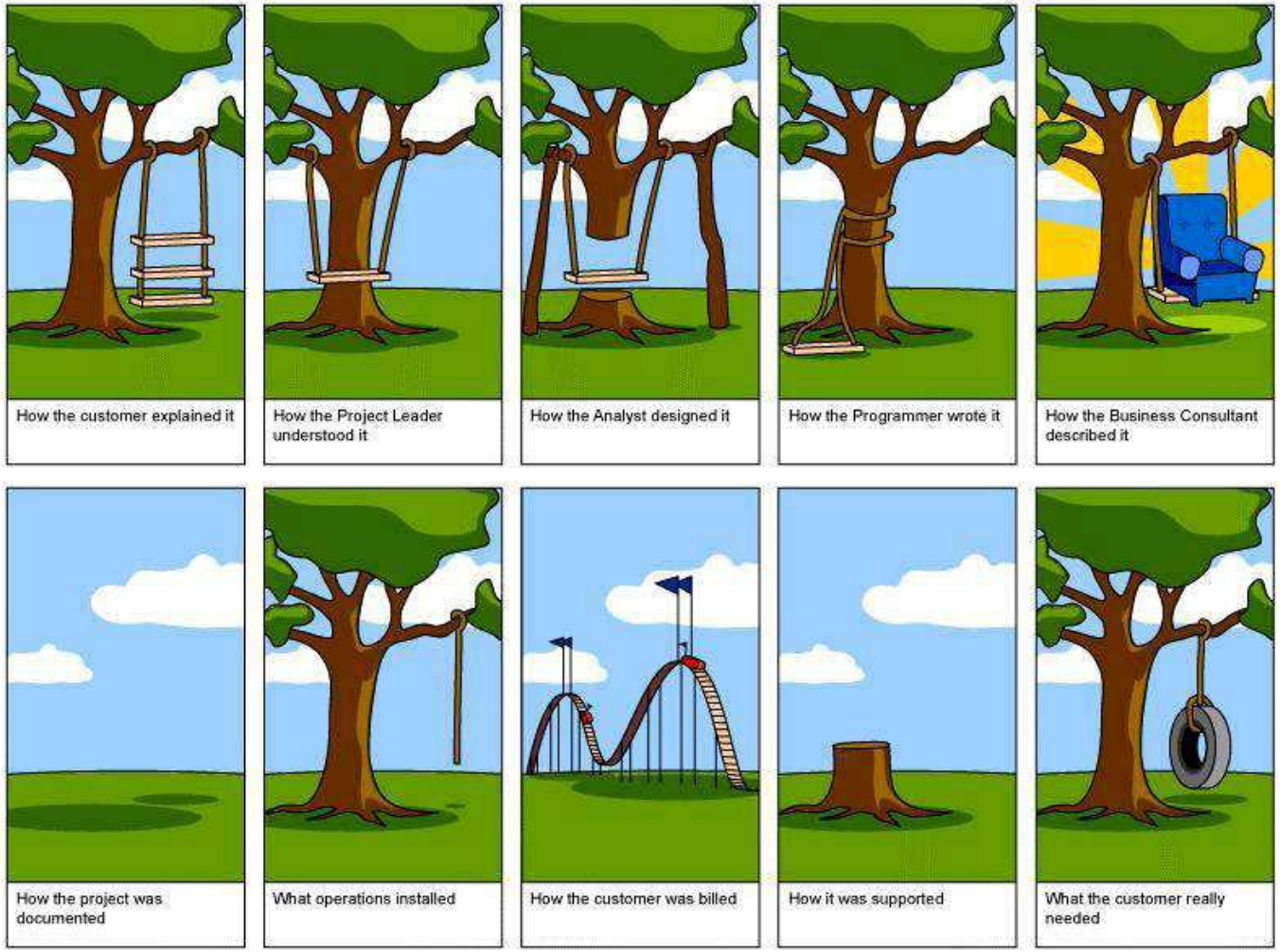
IUT de Nantes – Département Informatique

« Développer, Proposer, Satisfaire les besoins des utilisateurs »



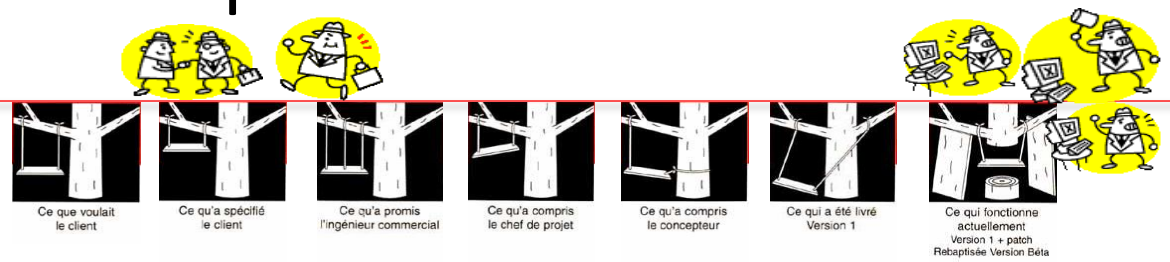
Inspiré de *http://www.bric-a-brac.org/humour/images/informatique*

« Développer, Proposer, Satisfaire les besoins des utilisateurs »

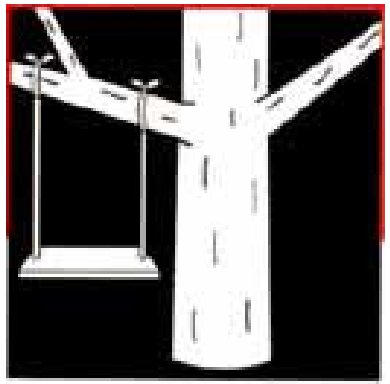


www.agiledeveloper.com

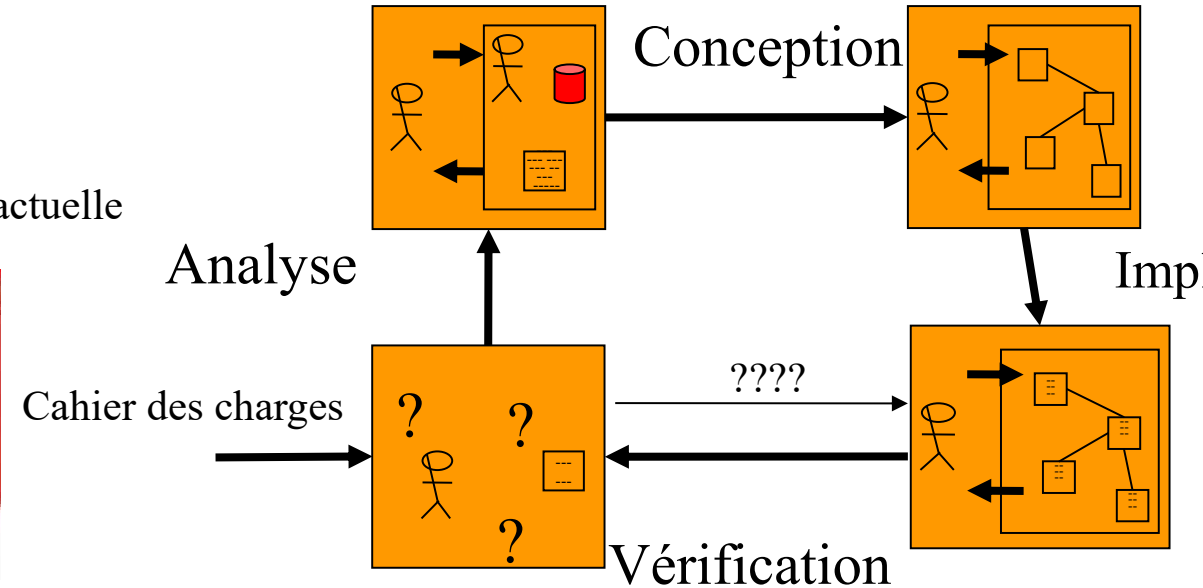
Principales étapes



- Situation actuelle
- Besoins



Ce que voulait le client

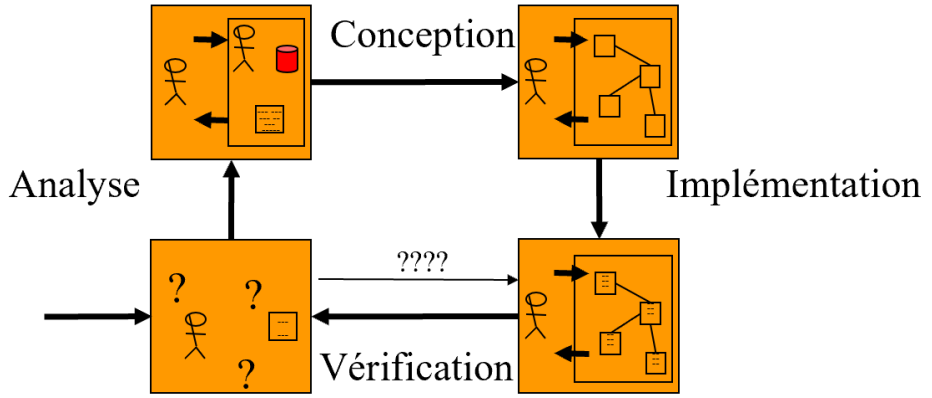


Ce qui fonctionne actuellement
Version 1 + patch
Rebaptisée Version Béta

Activités de développement

Processus de développement

- **Planification** du projet
- Recueil des **exigences**
- **Analyse** et spécification
- **Conception**
- **Implémentation**
- Vérification / **Test**
- Intégration
- Livraison / **Déploiement**
- **Maintenance**



En continu :

- Documentation
- Vérification et validation
- Gestion

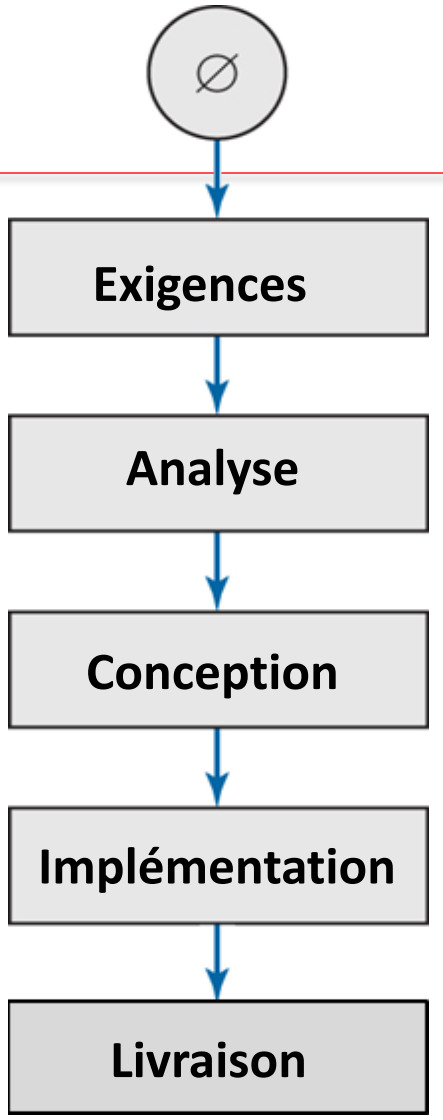
IUT Nantes
Pôle Sciences et technologie
Nantes Université

Développement logiciel idéal

- Linéaire
- Ne commence
- Développeme
- Suppose que
  premier coup

- **Idéaliste!**

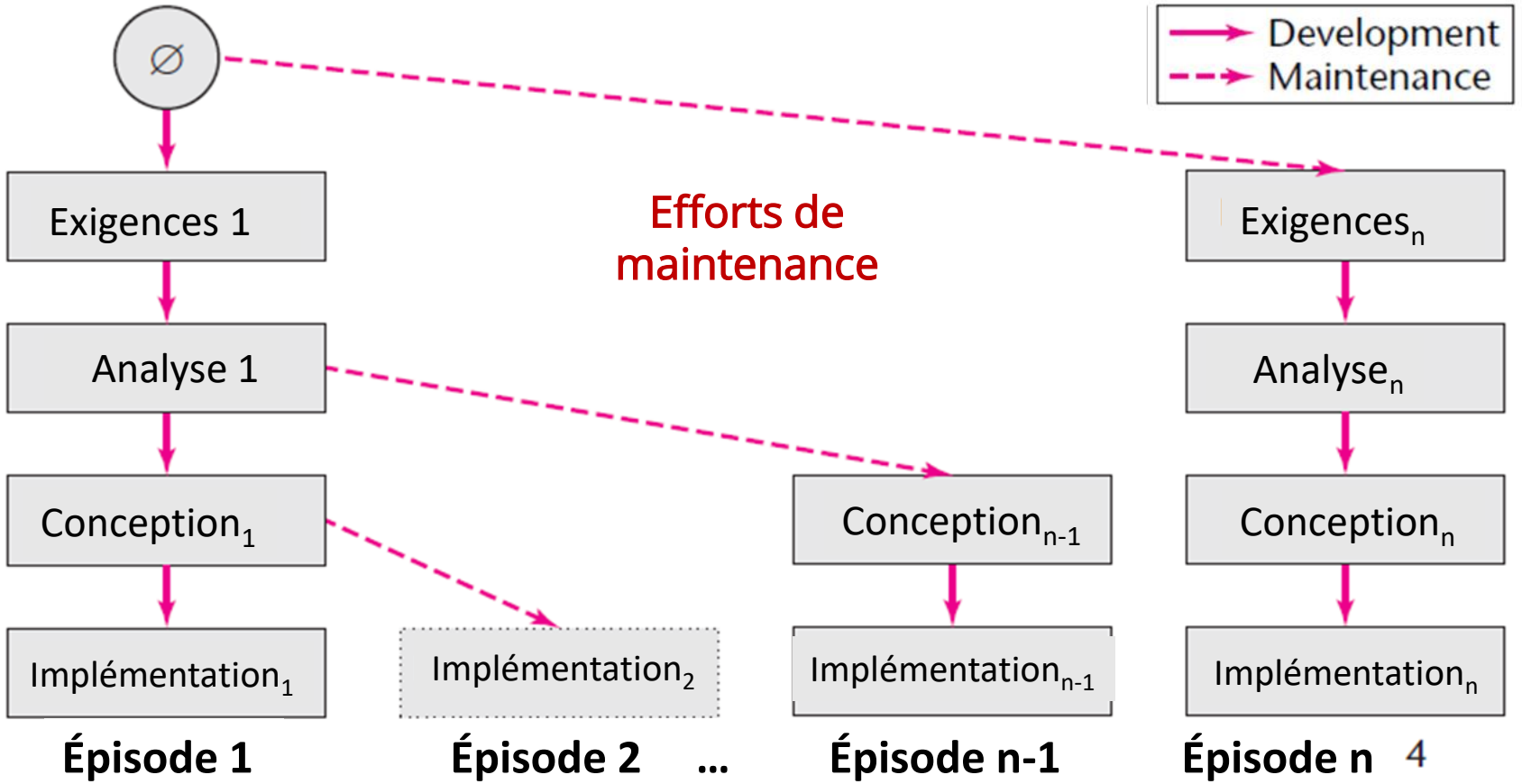
… voire utopique !



En pratique…

- Les développeurs font des erreurs
- Le client change les exigences pendant le développement du logiciel
- …

En pratique…

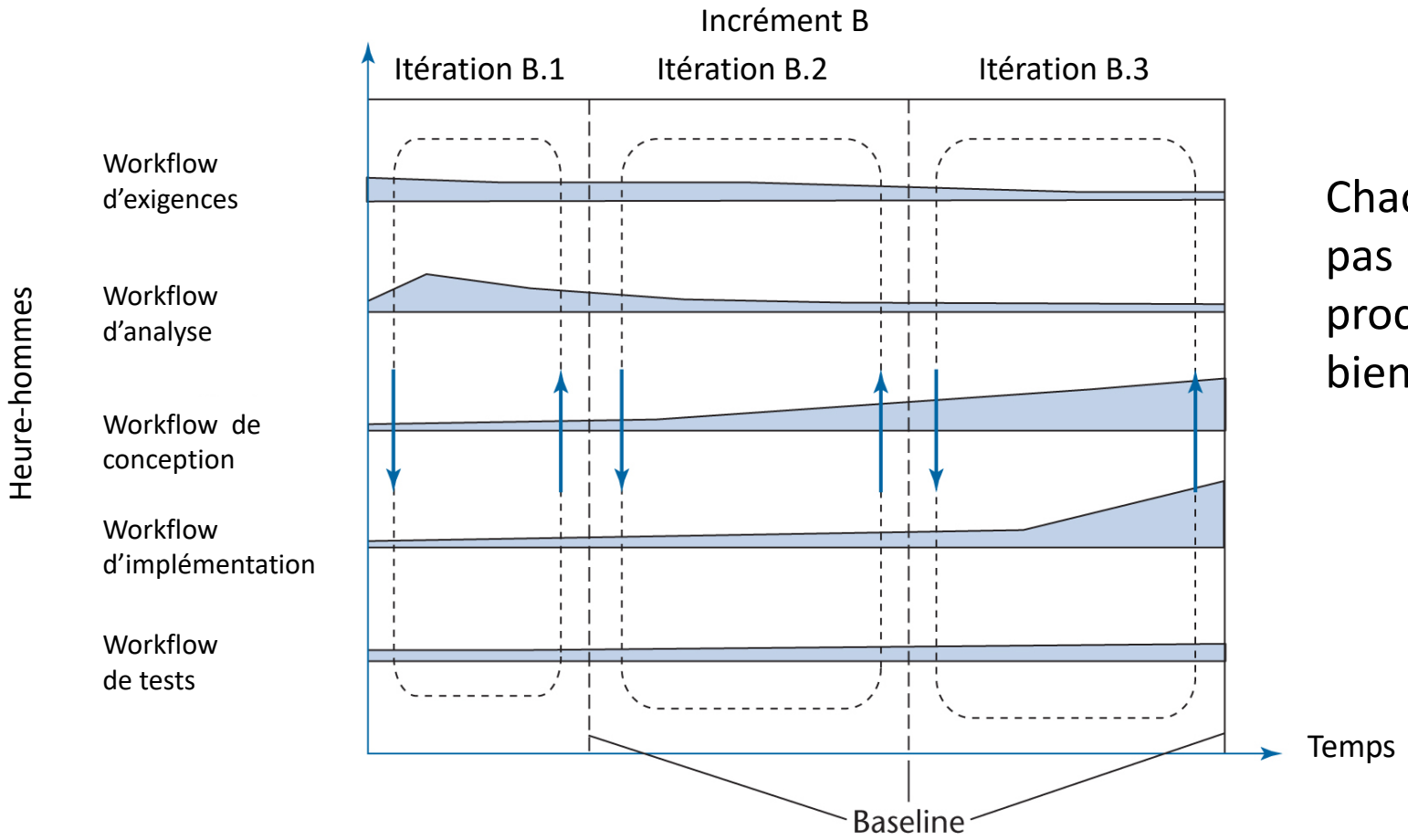

Development
Maintenance
∅
Exigences 1
Analyse 1
Conception$_1$
Implémentation$_1$
Efforts de maintenance
Implémentation$_2$
Conception$_{n-1}$
Implémentation$_{n-1}$
Exigences$_n$
Analyse$_n$
Conception$_n$
Implémentation$_n$
Épisode 1
Épisode 2
…
Épisode n-1
Épisode n
4

Modèles de processus de développement (résumé)

- **Linéaire** (Cascade, V)
  - Petits projets, cadre bien défini, peu de risques, pas de changements de besoins en phase de tests, tout est déployé d'un coup
- **Itératif** (Spirale)
  - Exigences pas encore claires, risque élevé de changements dans les exigences, prototype initial nécessaire, chaque itération améliore et converge vers le produit final suite au retour de la précédente
- **Incrémental** (Phases)
  - Livraison des composants par ordre de priorité pour réduire les risques et augmenter la qualité tout en incorporant les changements tout au long du projet. Les risques de conception et techniques éliminés tôt, pour de gros projet car les tests commencent tôt
- **Itératif et incrémental** (Unifié, agile)
  - Chaque composant est développé dès le 1er incrément, ils sont améliorés dans les incréments suivants, peut rapidement intégrer les changements et changer de priorités

Modèles de processus de développement par Itération et incrémentation (I&I)

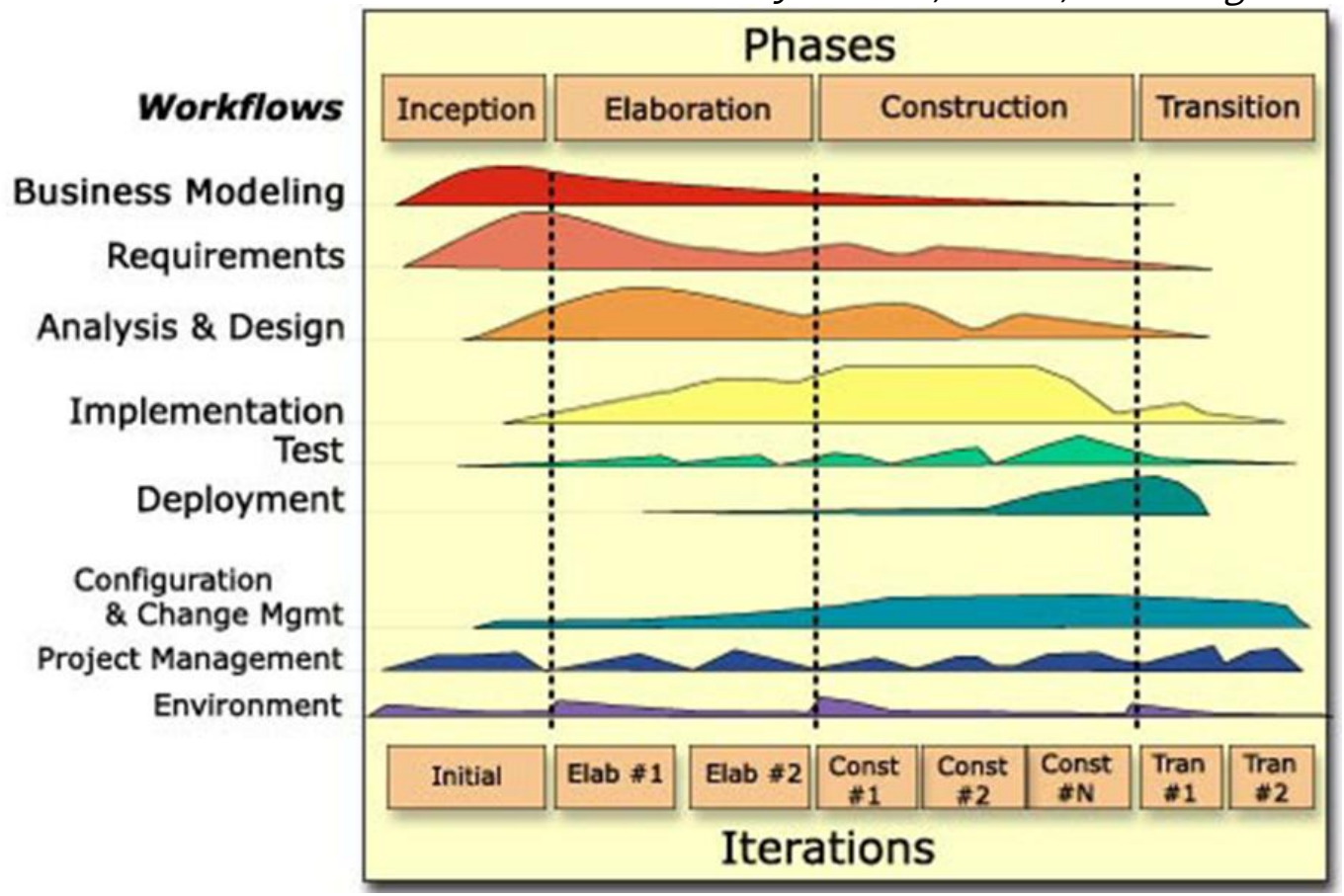
Chaque incrément est le résultat de plusieurs itérations



Chaque incrément n'est pas le résultat d'un processus linéaire, mais bien itératif.

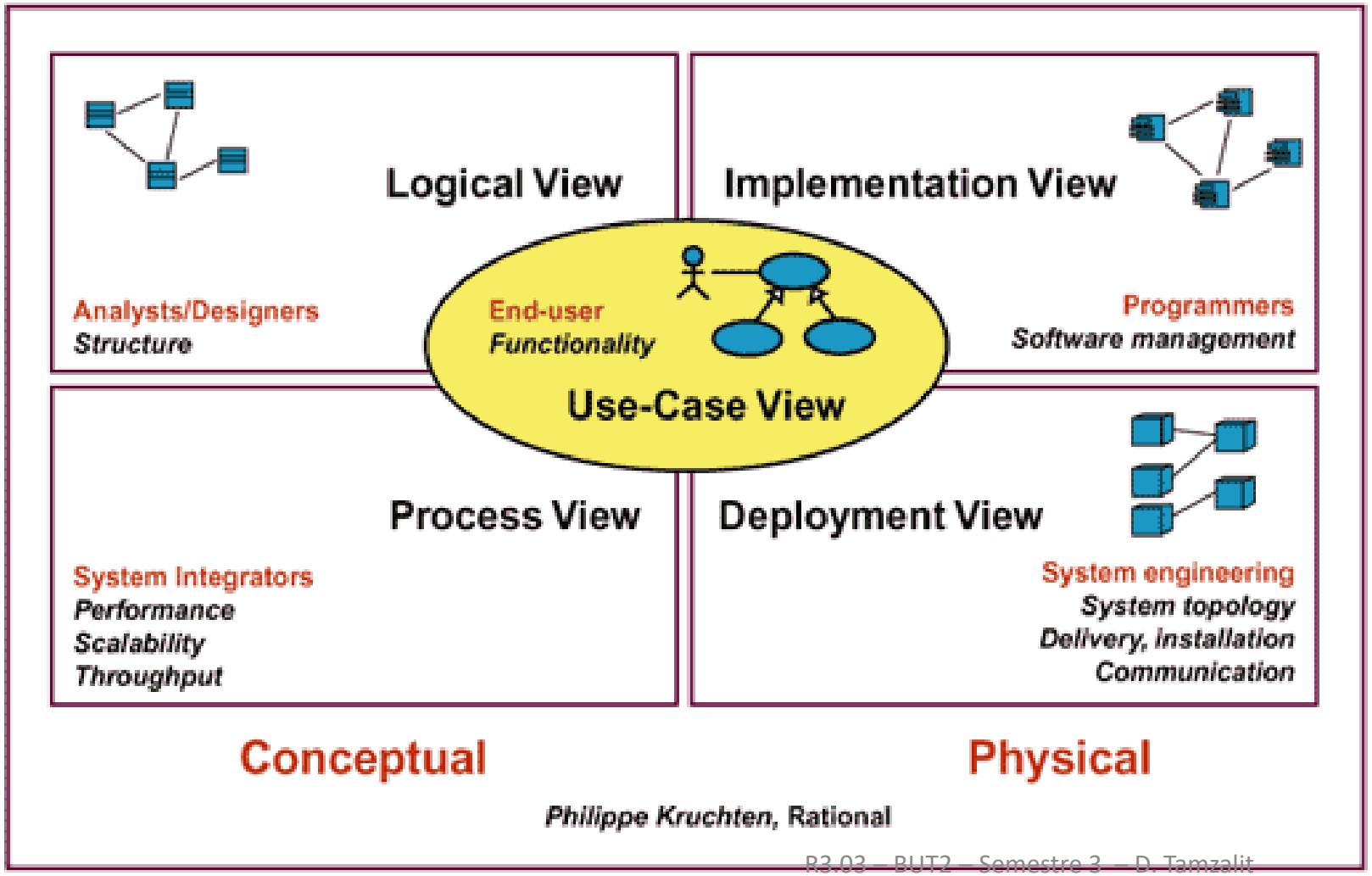
Modèle du Processus Unifié (*Unified Process*)

Jacobson, Booch, Rumbaugh 1999



- À partir duquel les processus I&I ont été élaborés
- RUP ’99 par les "Three Amigos" (Jacobson, Booch, Rumbaugh), pères de l’UML
- Il n’y a pas un modèle qui fonctionne sur tous les projets.
- Modèle adaptable, à modifier pour chaque logiciel à développer

UP et le modèle « 4+1 » vues



Le modèle d 'architecture de *Kruchten*, dit *modèle des 4 + 1 vues*, est adopté dans l'*Unified Process*. La vue des cas d'utilisation établit le lien et dirige la création des autres diagrammes d'architecture.

Fortement couplé à UML.

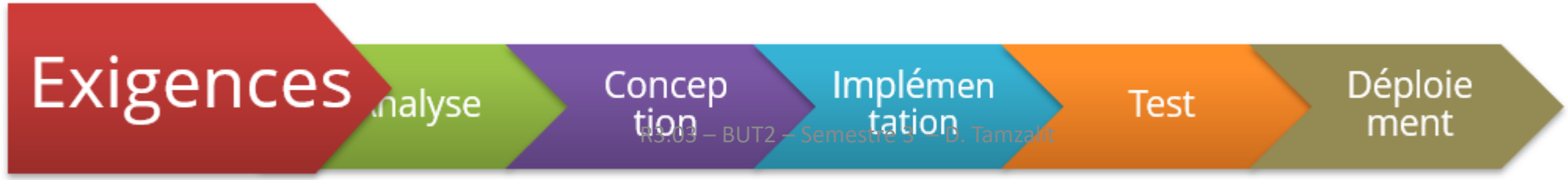
* Philippe B. Kruchten, *The 4+1 View Model of Architecture*, IEEE Software, novembre 1995.

Flux de travail *(Workflows)*

Workflow des exigences

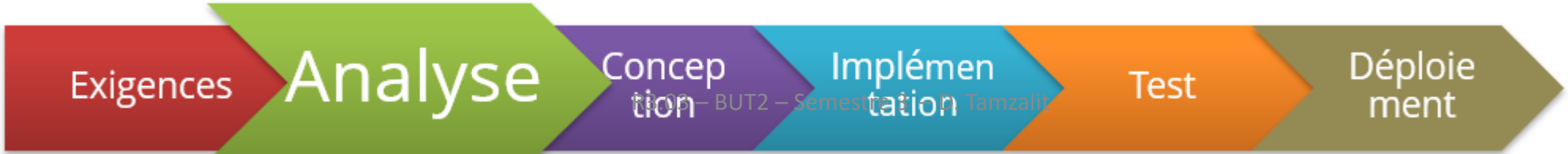
- But: déterminer ce dont le client a **besoin**

1. Comprendre le **domaine** d'application
   - Environnement spécifique au métier où le logiciel sera exploité.
   - Quel est le modèle d'affaires (vente de produits, fournir services…)
2. Construire un **modèle d'affaire**
   - Utiliser UML pour décrire le processus d'affaire du client (cas d'utilisation).
   - À n'importe quel moment, si le client pense que le coût n'en vaut pas la peine, on met immédiatement fin au développement
3. Définir les **exigences** du système
   - Découvrir quelles sont les **contraintes**



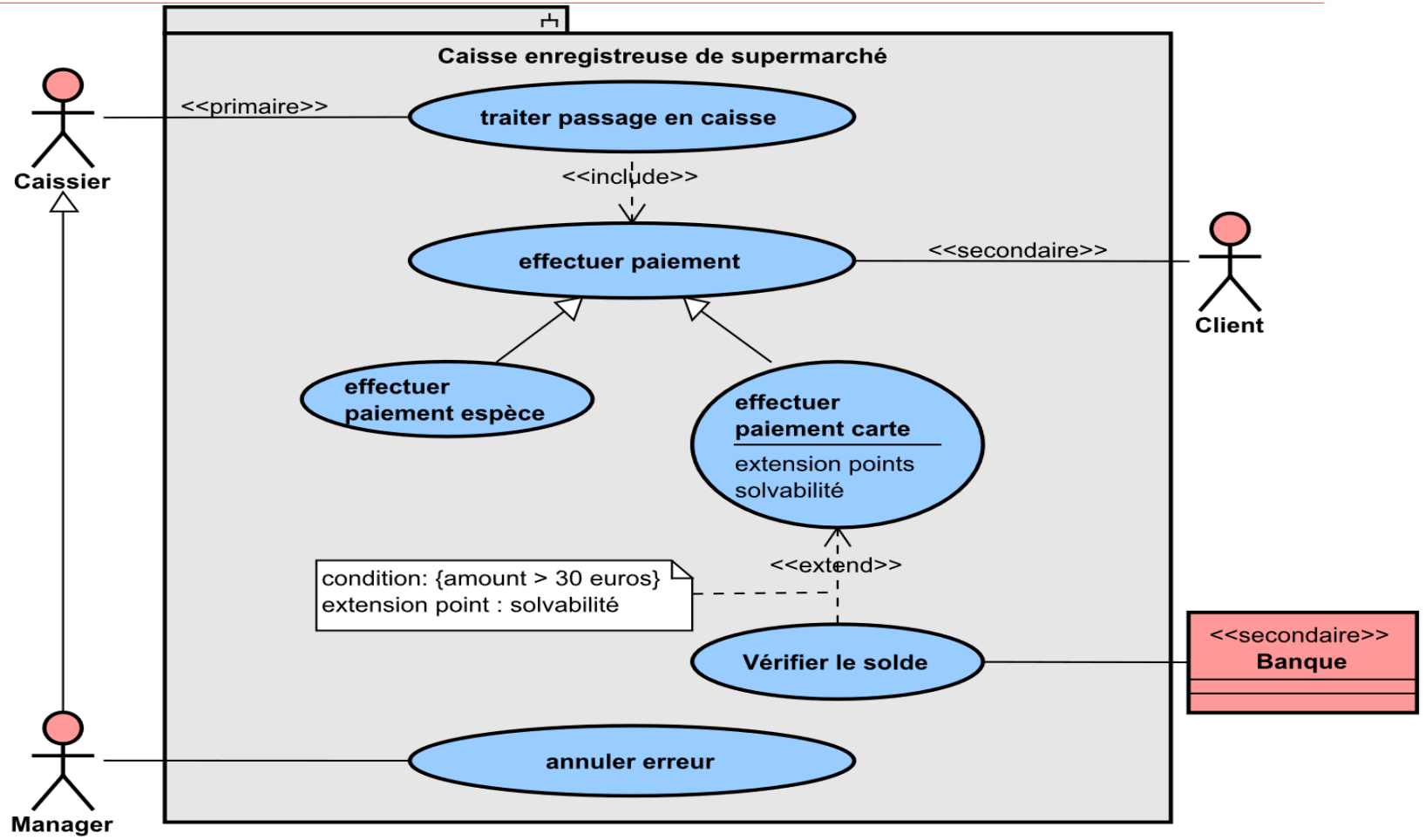
Workflow d'analyse

- But: analyser et **affiner** les exigences
- *Pourquoi ne pas le faire dans le workflow des exigences ?*
  - Exigences doivent être entièrement compréhensibles par le client
  - Exprimés en langage naturelle, donc imprécises/ambigües
  - Artéfacts d'analyse doivent être assez précis et complets pour l'équipe de développement
- Document (artefacts) de **spécifications**
  - Constitue un contrat, décrit ce que le logiciel doit faire
  - Précis et non ambigüe
- Spécifications doivent être complètes et correctes
  - Base pour les tests et la maintenance
  - Une fois approuvé, planification détaillée, cahier des charges, estimation du temps et du coût



Quels moyens pour l'analyse ? Diagrammes

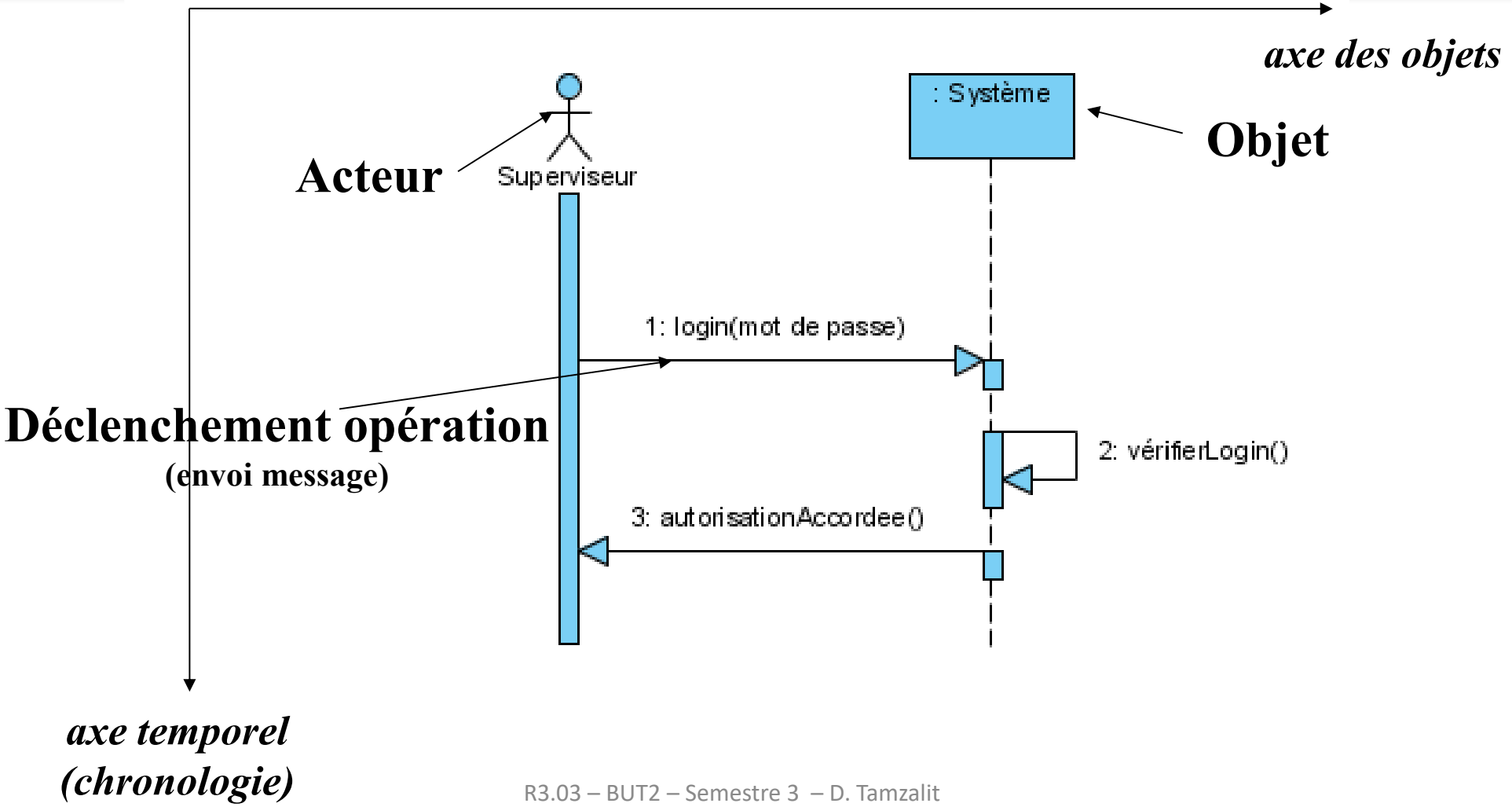
- Diagrammes de cas d'utilisation d'UML.
- Fonctionnement du système
- *Exemple* : cours de M. Mottu, en R2.10.
- Ne pas oublier les scénarios !



Diagrammes de séquences en phase d'analyse

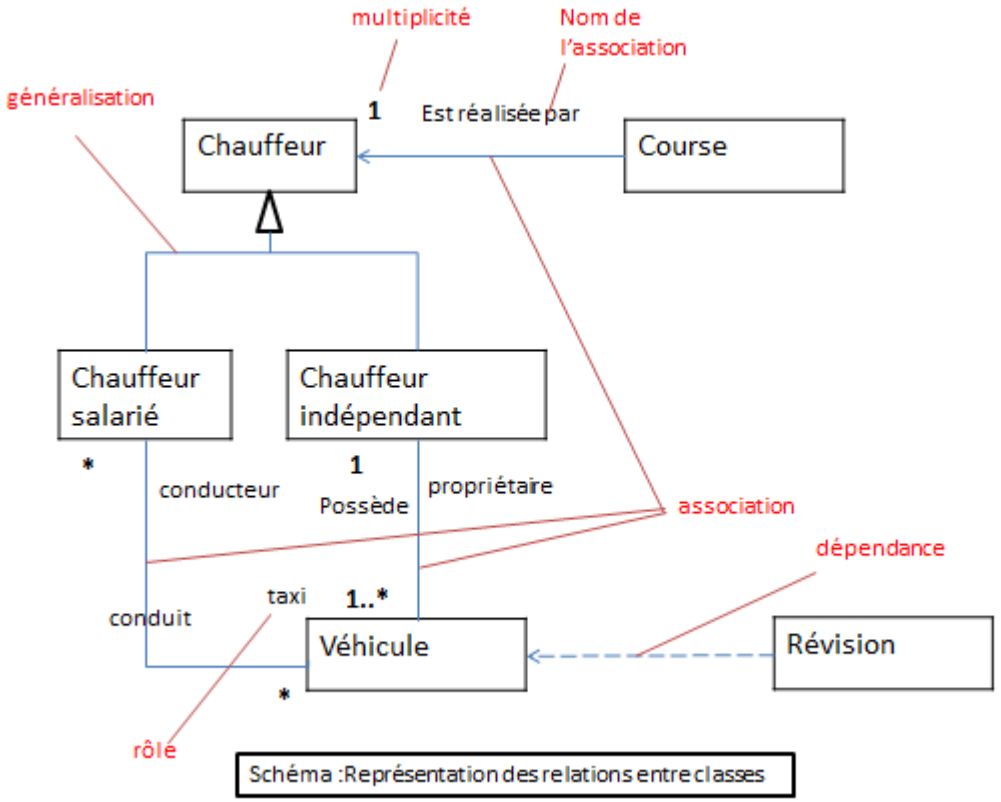
- Utilisation des diagrammes de séquences système (*boîte noire*).
- Généralement associé à un cas d'utilisation.
- Vue graphique d'un scénario (d'un cas d'utilisation) :
  - Met en évidence les interactions d'objets
  - Selon un point de vue temporel
- A deux dimensions :
  - verticale : représente le temps
  - horizontale : représente les différents objets.

Diagrammes de séquences en phase d'analyse



Quels moyens pour l'analyse ? Diagrammes

- Diagrammes de classes.
- Dimension statique du système : données et informations.
- Deux niveaux :
  - **Analyse**
  - Conception



https://fr.wikiversity.org/wiki/Mod%C3%A9lisation_UML/Le_diagramme_de_classes

Diagramme de classes d'analyse

- « On précise les entités métiers et leurs relations quelques méthodes-clefs pertinentes, certains types, etc. pour capturer le domaine métier » (confère ressource Intro Dev Objet en BUT1).
- Autre terminologie : conception architecturale, conception générale, analyse fonctionnelle
- Utilisation et utilité :
  - Outil de dialogues entre clients et utilisateurs
  - Comprendre les besoins métiers et les décrire
  - Extraire les modèles de l'expression des besoins
  - Uniquement les entités du domaine (métier)
  - Support pour la conception, l'implémentation et la maintenance
  - Description métier de ce que doit faire le système et comment il le fait sans la solution technique (trop tôt)

Diagramme de classes d'analyse

- Classes du domaine métier uniquement
- Use cases dans le langage du client mais diagramme d'analyse dans le langage du concepteur
- Objectif :
  - clarifier le domaine métier, se familiariser avec
  - pas de classes techniques
- Diagramme de classes d'analyse :
  - classes, attributs, associations
  - pas d 'opérations
- Démarche usuelle
  - diagramme de classes par use case
  - recouper les diagrammes de classes

Construction du DCA

- Recenser les objets métier dans :
  - les documents de définition des besoins
  - les documents d'analyse
- Noms ou groupes nominaux en classe, objet ou attribut.
- Doute entre attribut ou classe : préférer la classe puis affiner par la suite.
  - Exemple : adresse en attribut de Personne ou en classe associée à Personne ?
- Attention aux faux amis et doublons.
- Eviter les classes fonctionnelles qui font du traitement :
  - gestionnaire du planning, décodeur de message …

Construction du DCA

- Identifier les classes d'objets
  - puis garder les bonnes classes
  - Constitution du dictionnaire de données
- Identifier les associations
  - puis garder les bonnes associations
- Identifier les attributs
  - puis garder les bons attributs
- Raffiner au moyen de l'héritage
  - Généralisations et raffinages
- Itérer la modélisation
- Grouper les classes en modules/paquetages

Diagramme de classes d ’analyse

- Démarche
  - Dictionnaire des données/glossaire
  - Les classes (noms communs)
  - Les objets (noms propres ou nom référencés)
  - Les attributs (données simples qui qualifient une classe ou un objet)
  - Les éléments qui sont étrangers à notre problème ou qui n'y ont pas de rôle réel sont omis

IUT Nantes
Pôle Sciences et technologie
Nantes Université

Révisions

Visit **gosocrative.com** and enter room name TAMZALIT

Ressources

- Précédents cours DUT
- https://ca.insight.com/fr_CA/content-and-resources/2016/07152016-types-of-software-development-models.html
- https://membres-lig.imag.fr/dubousquet/docs/2.2_CyclesDeVie.pdf
- « GL – 2 Processus de développement, Cycles de vie », Lydie du Bousquet, IMAG.
- Support cours E. Syriani, Université de Montréal.