GÉNIE LOGICIEL - L2

Cours 1

Introduction au génie logiciel

Analyse des besoins

Laboratoire Informatique Image Interaction (L3I)
Université de La Rochelle - Pôle Sciences et Technologie - Avenue Michel Crépeau - 17042 LA ROCHELLE CEDEX 1 France
Tél : +33 (0)5 46 45 82 62 – Fax : 05.46.45.82.42 – Site internet : http://l3i.univ-larochelle.fr/

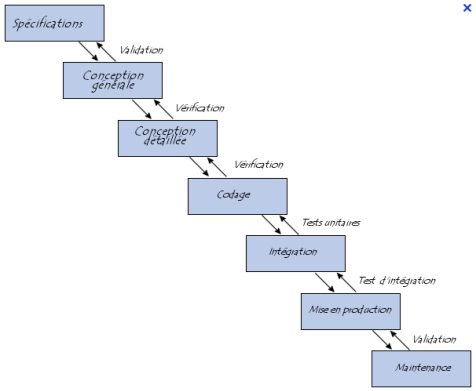
1ERE PARTIE

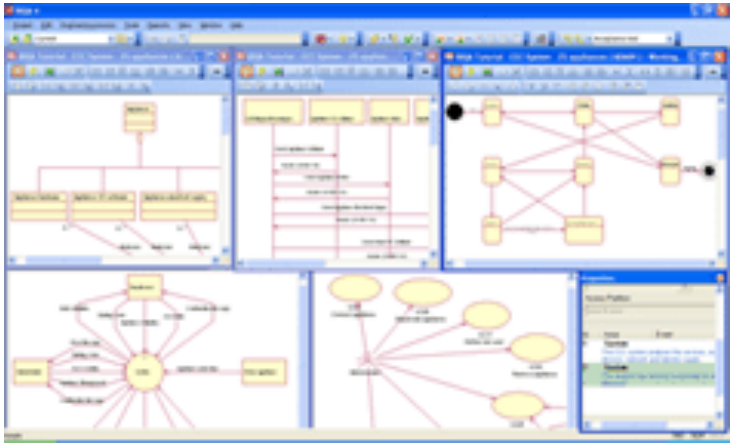
Introduction au génie logiciel

Laboratoire Informatique Image Interaction (L3I)
Université de La Rochelle - Pôle Sciences et Technologie - Avenue Michel Crépeau - 17042 LA ROCHELLE CEDEX 1 France
Tél : +33 (0)5 46 45 82 62 – Fax : 05.46.45.82.42 – Site internet : http://l3i.univ-larochelle.fr/

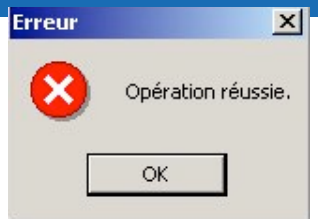
UNE INTRODUCTION AU GÉNIE LOGICIEL

1. Quelles sont les problématiques du développement logiciel?
2. La qualité d'un logiciel.
3. Le cycle en V

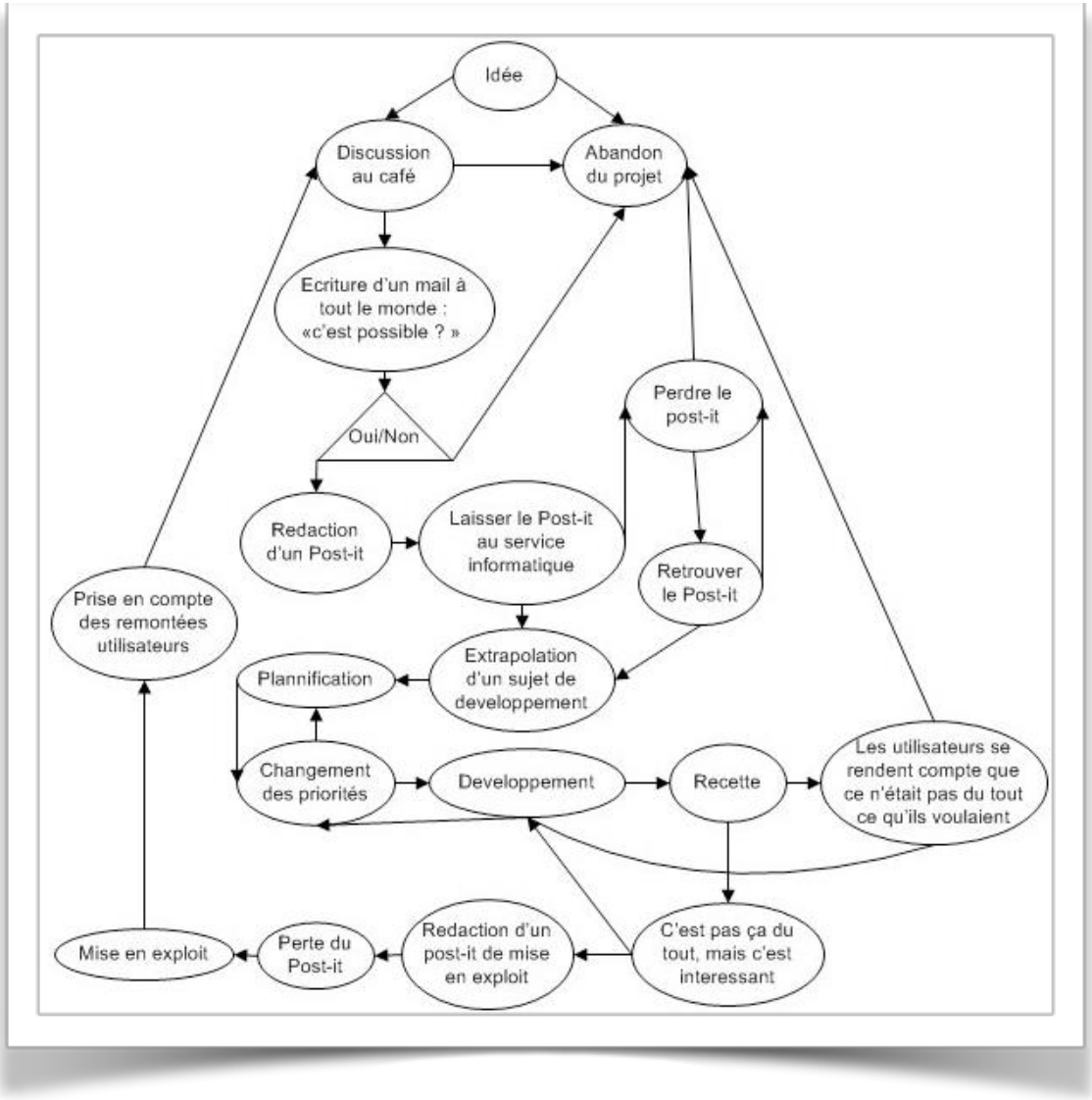




MÉTHODE LA-RACHE.COM

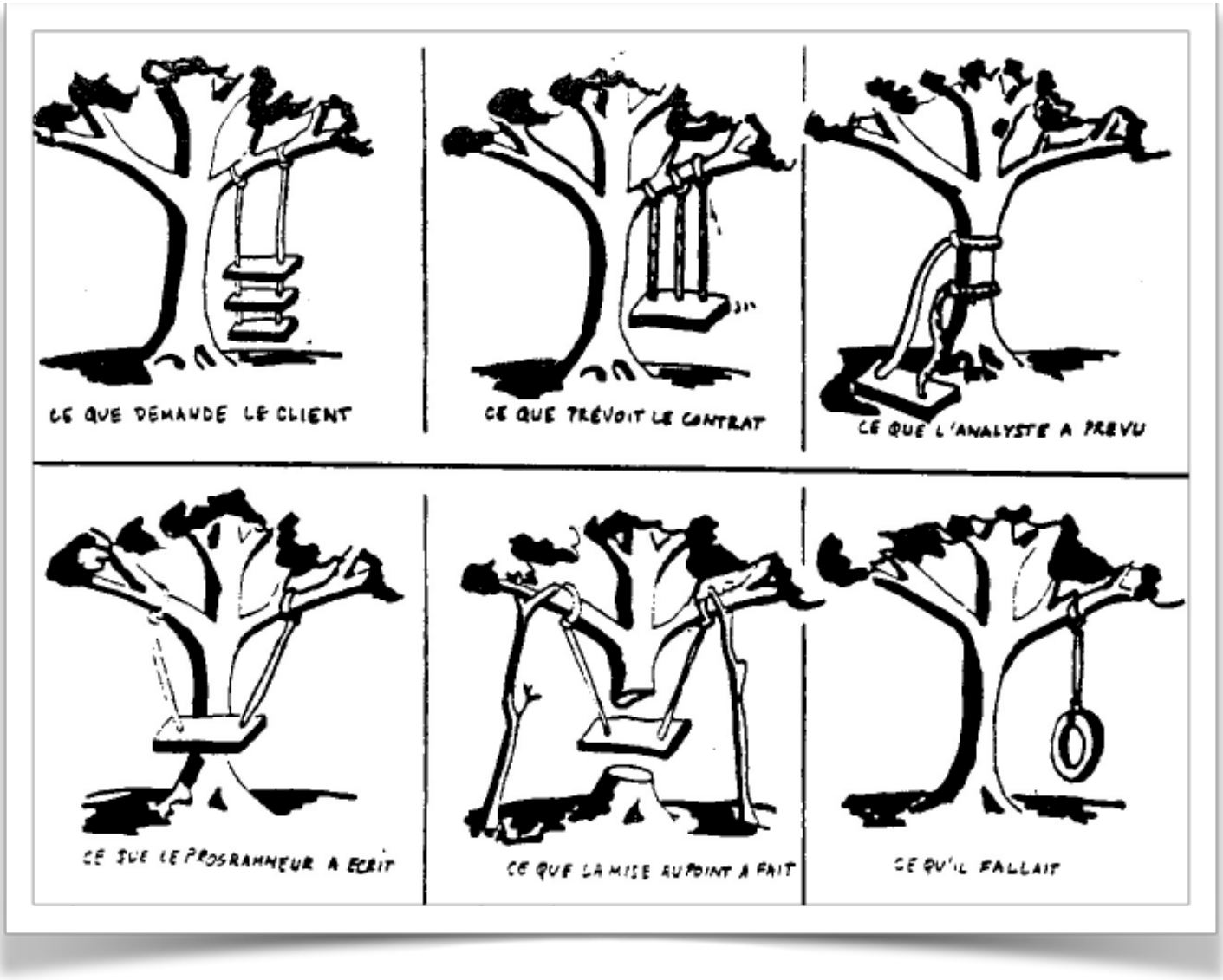


Quelles sont les problématiques ?



http://www.cafenware.com/la-rache/

LES DIFFÉRENTES ÉTAPES DU DÉVELOPPEMENT



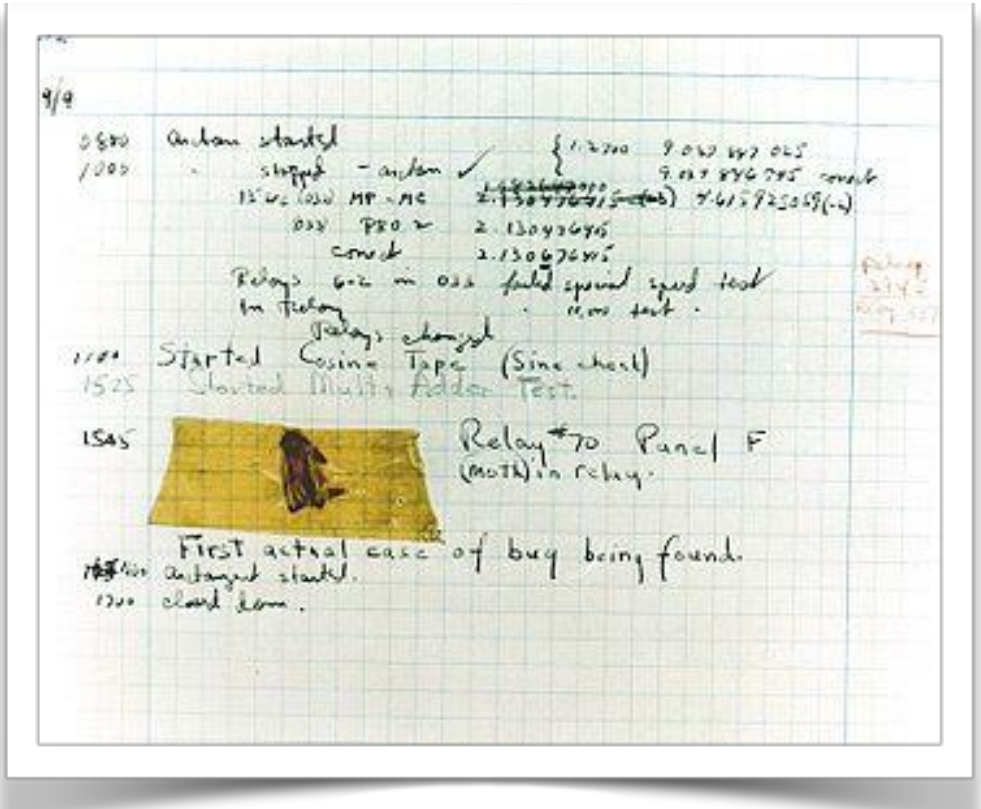
Quelles sont les problématiques ?

QUELQUES BUGS CÉLÈBRES … ET LEURS COÛTS

Le premier «cas avéré de bug» : 9 septembre 1945

Découvert par Grace Hopper

Dans le journal d'entretien, le corps de la mite trouvée entre deux relais.



Quelles sont les problématiques ?

QUELQUES BUGS CÉLÈBRES … ET LEURS COÛTS

Quelles sont les problématiques ?

Perte de Mars Climate Orbiter, le 23 septembre 1999, après 9 mois de voyage. Coût : 120 M$
Cause : confusion entre pieds et mètres.

Mission Vénus : passage à 5 000 000 de Km de la planète, au lieu de 5 000 Km prévus.
Cause : remplacement d'une virgule par un point (au format US des nombres).

Ebay : L'indisponibilité durant 22 heures du serveur web de EBay, site de vente aux enchères, a fait échouer plus de 2,3 millions d'enchères. Dans un secteur où la compétitivité dépend plus que jamais du zéro-défaut face au consommateur, EBay a annoncé qu'elle remboursera les frais d'enregistrement des enchères en cours le jeudi 10 juin 1999, soit entre 3 et 5 millions de dollars. A Wall Street, le titre a chuté de plus de 9%. (S&T Presse, 14 juin 1999)

ARIANE 501 : L'ÉCHEC DU 4 JUIN 1996

Quelles sont les problématiques ?

Conditions météorologiques acceptables

- Visibilité "correcte", pas de risque de foudre, champs électrique négligeable

Chronologie de lancement correcte

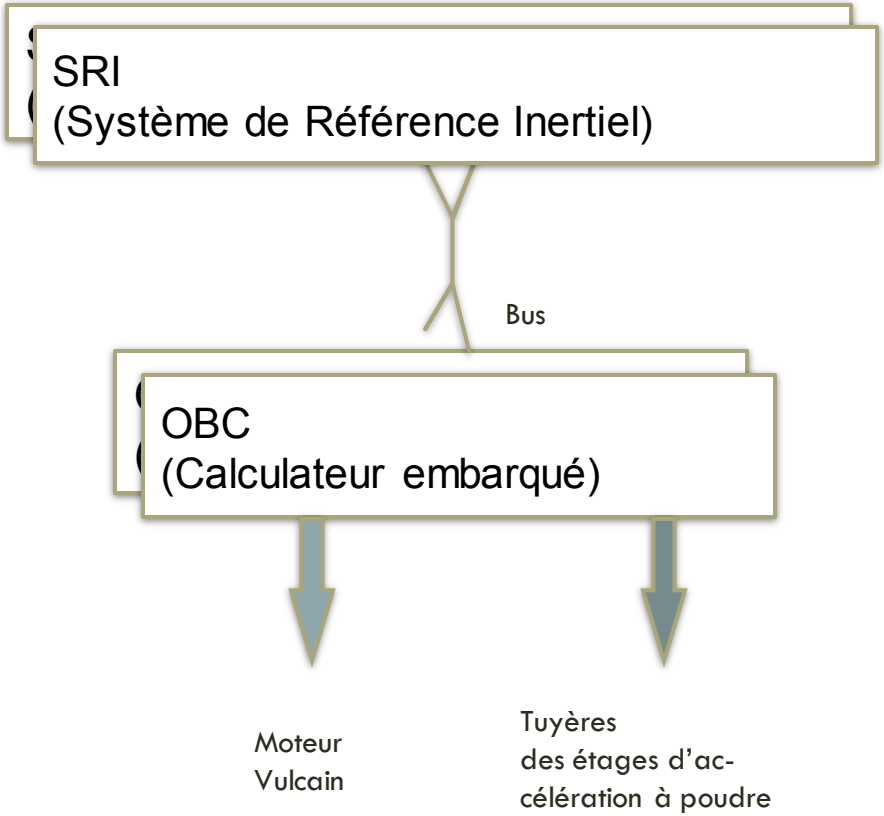
- Arrêt du remplissage de l'étage cryotechnique de quelques minutes suite à une dégradation temporaire de la visibilité
- Allumage (moteur principal Vulcain et moteurs à propergol) nominal
- Début de vol nominal

vers H0+40 secondes :

- déviation soudaine de la trajectoire
- Explosion de la fusée

ANALYSE PRÉLIMINAIRE DE L'ACCIDENT ET CONTEXTE

Quelles sont les problématiques ?



Réplication : SRI, OBC etc... doublés
Réutilisation
La conception des SRI est pratiquement la même que pour Ariane 4 (notamment en ce qui concerne le logiciel)

Ce qui est observé (extraits du rapport)

un comportement nominal du lanceur jusqu'à H0 + 36 secondes ;

la défaillance du système de référence inertielle actif, suivie immédiatement par celle du système de référence inertielle de secours ;

le braquage en butée des tuyères des deux moteurs à propergols solides puis, quelques instants après, du moteur Vulcain, provoquant le basculement brutal du lanceur ;

l'auto-destruction du lanceur déclenchée correctement par la rupture des liaisons entre les étages d'accélération à poudre et l'étage principal.

A priori : "On a pu remonter rapidement jusqu'à l'origine de cette défaillance dans le système de pilotage et, plus précisément, dans les systèmes de référence inertielle qui, à l'évidence, ont cessé de fonctionner presque simultanément à environ H0 + 36,7 s"

RAPPORT FINAL : ANALYSE DE L'ÉCHEC (1)

Quelles sont les problématiques ?

(1> Le lanceur a commencé à se désintégrer à environ H0 + 39 s sous l'effet de charges aérodynamiques élevées dues à un angle d'attaque de plus de 20°; qui a provoqué la séparation des étages d'accélération à poudre de l'étage principal, événement qui a déclenché à son tour le système d'auto-destruction du lanceur ;

(2> Cet angle d'attaque avait pour origine le braquage en butée des tuyères des moteurs à propergols solides et du moteur principal Vulcain ;

(3> Le braquage des tuyères a été commandé par le logiciel du calculateur de bord (OBC) agissant sur la base des données transmises par le système de référence inertielle actif (SRI2). A cet instant, une partie de ces données ne contenait pas des données de vol proprement dites mais affichait un profil de bit spécifique de la panne du calculateur du SRI 2 qui a été interprété comme étant des données de vol ;

(4> La raison pour laquelle le SRI 2 actif n'a pas transmis des données d'altitude correctes tient au fait que l'unité avait déclaré une panne due à une exception logiciel ;

RAPPORT FINAL : ANALYSE DE L'ÉCHEC (2)

Quelles sont les problématiques ?

(5> L'OBC n'a pas pu basculer sur le SRI 1 de secours car cette unité avait déjà cessé de fonctionner durant le précédent cycle de données (période de 72 ms) pour la même raison que le SRI 2 ;

(6> L'exception logiciel interne du SRI s'est produite pendant une conversion de données de représentation flottante à 64 bits en valeurs entières à 16 bits. Le nombre en représentation flottante qui a été converti avait une valeur qui était supérieure à ce que pouvait exprimer un nombre entier à 16 bits. Il en est résulté une erreur d'opérande (division par 0). Les instructions de conversion de données (en code Ada) n'étaient pas protégées contre le déclenchement d'une erreur d'opérande bien que d'autres conversions de variables comparables présentes à la même place dans le code aient été protégées ;

(7> L'erreur s'est produite dans une partie du logiciel qui n'assure que l'alignement de la plate-forme inertielle à composants liés. Ce module de logiciel calcule des résultats significatifs avant le décollage seulement. Dès que le lanceur décolle, cette fonction n'est plus d'aucune utilité ;

RAPPORT FINAL : ANALYSE DE L'ÉCHEC (3)

Quelles sont les problématiques ?

(8> La fonction d'alignement est active pendant 50 secondes après le démarrage du mode vol des SRI qui se produit à H0 - 3 secondes pour Ariane 5. En conséquence, lorsque le décollage a eu lieu, cette fonction se poursuit pendant environ 40 secondes de vol. Cette séquence est une exigence Ariane 4 mais n'est pas demandé sur Ariane 5 ;

(9> L'erreur d'opérande s'est produite sous l'effet d'une valeur élevée non prévue d'un résultat de la fonction d'alignement interne appelé BH (Biais Horizontal) et lié à la vitesse horizontale détectée par la plate-forme. Le calcul de cette valeur sert d'indicateur pour la précision de l'alignement en fonction du temps ;

(10> La valeur BH était nettement plus élevée que la valeur escomptée car la première partie de la trajectoire d'Ariane 5 diffère de celle d'Ariane 4, ce qui se traduit par des valeurs de vitesse horizontale considérablement supérieures.

Ce scénario a été reproduit après coup avec les données du vol en séance de simulation

CONCLUSIONS À TIRER

Quelles sont les problématiques ?

Le problème a pour origine une **séquence compliquée** de circonstances (décisions, oublis...)

Attention à la **réutilisation** et aux **conditions** de réutilisation

Le logiciel embarqué d'Ariane 5 constitue plusieurs centaines de milliers de lignes de code

QUELQUES IDÉES REÇUES …

Une idée grossière du logiciel à réaliser est suffisante pour commencer d'écrire un programme (il est assez tôt de se préoccuper des détails plus tard).

- *Faux : une idée imprécise du logiciel à réaliser est la cause principale d'échecs*

Une fois que le programme est écrit et fonctionne, le travail est terminé.

- *Faux : la maintenance du logiciel représente un travail important : le coût de la maintenance représente plus du 50 % du coût total d'un logiciel*

Si les spécifications du logiciel à réaliser changent continuellement, cela ne pose pas de problème, puisque le logiciel est un produit souple

- *Faux : des changements de spécifications peuvent se révéler coûteux et prolonger les délais*

Si la réalisation du logiciel prend du retard par rapport aux délais prévus, il suffit d'ajouter plus de programmeurs afin de finir dans les délais.

- *Faux : si l'on ajoute de gens à un projet, il faut compter une période de familiarisation. Le temps passé à communiquer à l'intérieur du groupe augmente également, lorsque la taille du groupe augmente*

LES CARACTÉRISTIQUES DU LOGICIEL

Quelles sont les problématiques ?

Un logiciel est un produit avec des caractéristiques particulières

Le processus de développement se poursuit après la livraison du logiciel, pour la **maintenance**

Après son développement un produit logiciel est mis en exploitation et entre dans une phase de maintenance qui peut remettre en cause les fonctions du système et impliquer des modifications et/ou un re-développement

Un logiciel est un produit (manufacturé) comme une voiture ou une machine à outils

Procédé de fabrication : processus de développement

LA MAINTENANCE DU LOGICIEL

La correction des défauts : maintenance corrective

Adaptation du logiciel à un nouvel environnement : maintenance adaptative

Amélioration des caractéristiques et suivi de l'évolution des besoins : maintenance perfective

Quelles sont les problématiques ?

TEMPS NÉCESSAIRE POUR LA CORRECTION DES BUGS

Source : «l'architecte et le chaos»

J. Printz - Professeur émérite - CNAM - Chaire de génie logiciel

On dispose également de statistiques précises sur les durées moyennes de corrections des erreurs en intégration. Voici celles de Hewlett Packard[17] :

Temps moyen de correction des défauts (soit environ 6h/défaut)
25% des défauts sont diagnostiqués et corrigés en 2h
50% des défauts sont diagnostiqués et corrigés en 5h
20% des défauts sont diagnostiqués et corrigés en 10h
4% des défauts sont diagnostiqués et corrigés en 20h
1% des défauts sont diagnostiqués et corrigés en 50h

Temps moyen de correction des défauts

THE CHAOS REPORT

http://www.projectsmart.co.uk/docs/chaos-report.pdf

Traduction : http://www.fabrice-aimetti.fr/dotclear/index.php?post/2010/02/11/The-Chaos-Report-1994

Constat du coût important du développement du logiciel

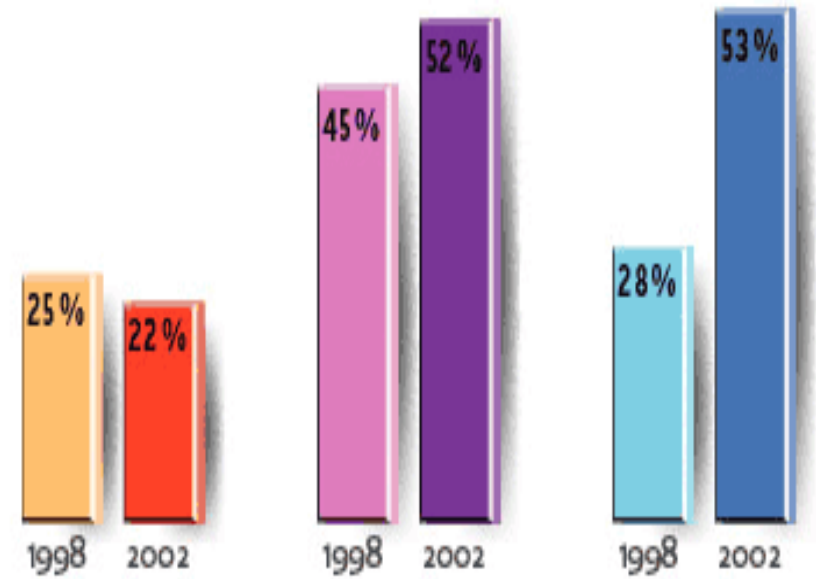
Constat des dépassements importants et des échecs

The survey participants were also asked about the factors that cause projects to be challenged.

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

UN PEU PLUS TARD....

Quelles sont les problématiques ?

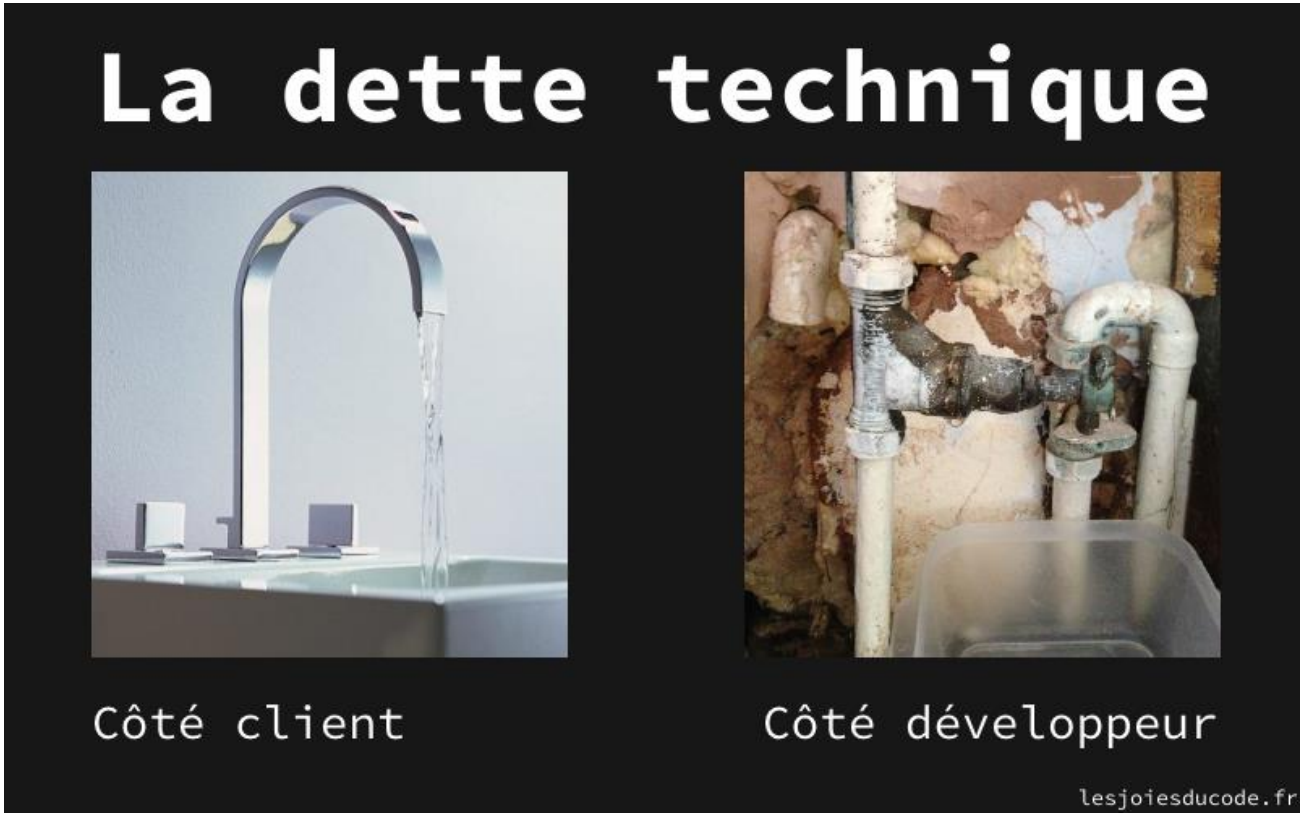


Étude réalisée par The Standish Group International (completée en 2002)

Quelques sources de bug

- Fuite mémoire
- Erreur de segmentation
- Dépassement de capacité
- Dépassement de tampon
- Dépassement de pile
- Situation de compétition
- Interblocage

LA DETTE TECHNIQUE



MODÈLES DE CYCLE DE VIE

Définition :

- Modèles généraux de développement décrivant les enchaînements et les interactions entre activités.

Objectif :

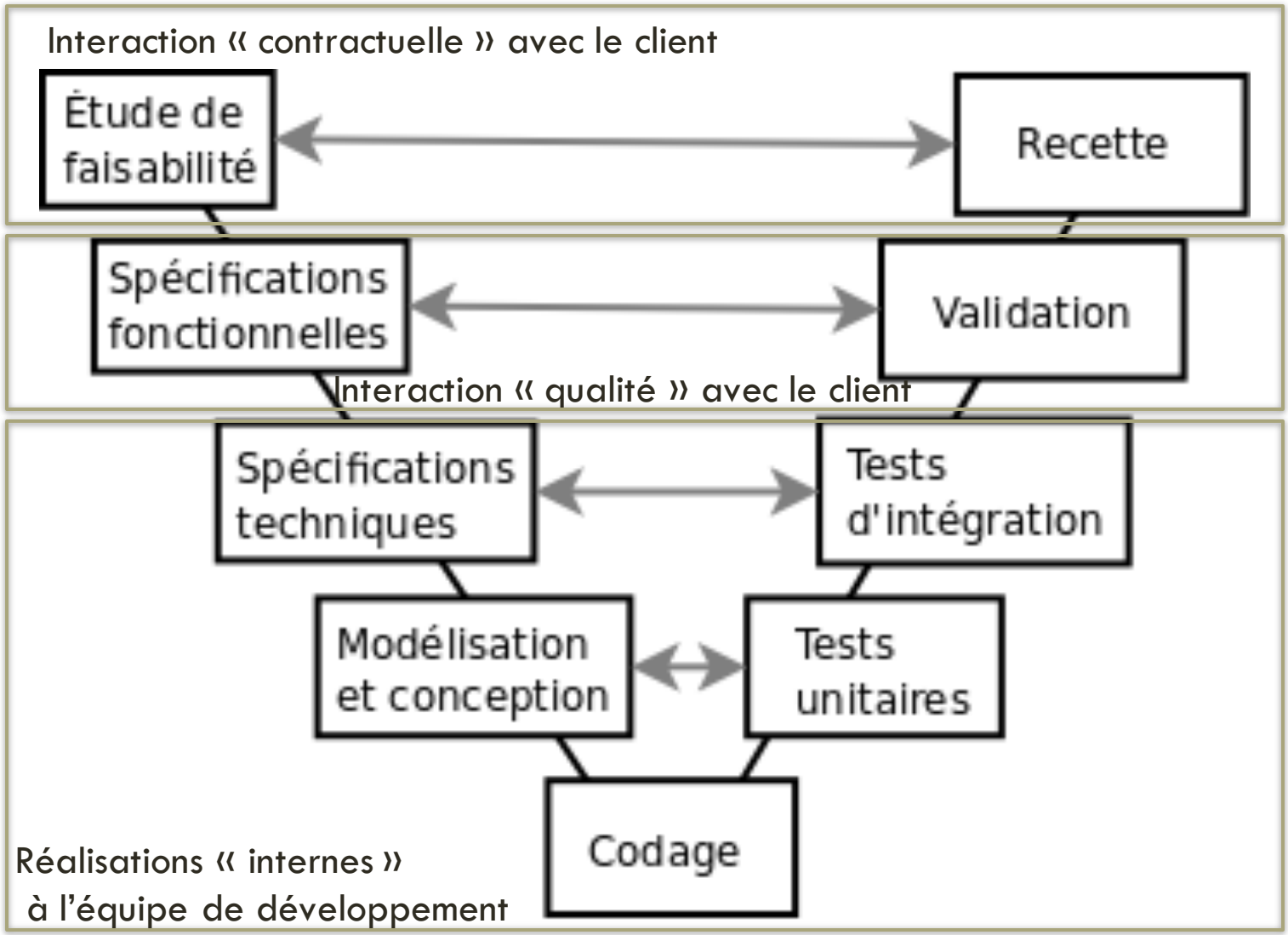
- Obtenir des processus de développement rationnels, reproductibles et contrôlables. Utilisés pour mieux maîtriser le processus développement en permettant de prendre en compte, en plus des aspects techniques, l'organisation et les aspects humains

Remarques :

- Une étape, telle que la conception, peut faire intervenir plusieurs activités, comme celles de la spécification globale, du prototypage et de la validation

- une activité comme la documentation peut se dérouler pendant plusieurs étapes

LE CYCLE DE VIE DU LOGICIEL



LA SPÉCIFICATION FONCTIONNELLE

Un document encore plus précis rédigé par les développeurs (itérations avec le client)

- Etape 1 : Rédiger les exigences
- Etape 2 : Rédiger la spécification fonctionnelle
  - Les entrées/sorties de chaque fonctionnalité
  - L'arborescence détaillée du site
  - Les contraintes techniques

FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

Document complet de description du système

Sert de support

- À la contractualisation
- A l'étape de conception architecturale et détaillée

PLAN-TYPE D'UN DOSSIER DE SPECIFICATIONS LOGICIELLES

Sujet

FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

TABLE DES MATIERES

FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

1 INTRODUCTION

1.1 But du document

Ce paragraphe est très court. Il s'agit là de présenter la phase de spécifications des besoins du logiciel d'un point de vue méthodologique. Il dépend très peu de l'application elle même.

1.2 Contexte de l'application

Ce paragraphe présente l'application dans son contexte.

1.3 Documentation

1.3.1 Documents applicables

Ce paragraphe liste tous les documents applicables à la présente phase du projet (plan d'assurance qualité, normes, etc.).

1.3.2 Documents de référence

Ce paragraphe liste tous les documents qui servent à l'élaboration du présent document. Il peut faire foi en cas de litige entre un client et un fournisseur.

2 SPECIFICATIONS GENERALES

2.1 Présentation du logiciel

2.1.1 Environnement du logiciel

Ce paragraphe décrit sommairement l'environnement d'exécution du logiciel.

Exemple :

Contexte du système	Description des interactions entre les entités externes et le système.
Architecture physique	Identification des principaux éléments physiques du système (diagrammes de déploiement).

2.1.2 Description générale du logiciel

Ce paragraphe décrit en quelques lignes les fonctionnalités du logiciel. Il s'agit de donner un premier aperçu sans rentrer dans le détail. Donner un schéma représentant la structure générale du projet et un autre représentant l'utilisation du programme.

Exemple :

Objectifs du système	Définition des principales exigences fonctionnelles.

2.2 Contraintes opérationnelles

2.2.1 Performances

Ce paragraphe précise les temps de réponses attendus.

FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

2.2.2 Installation

Ce paragraphe précise sous quelle forme est livré le logiciel: installation automatique, source et/ou exécutable, éventuellement méthodes de création d'exécutables à partir des sources.

2.3 Contraintes de réalisation

2.3.1 Contraintes structurelles

Ce paragraphe décrit les choix déjà faits ou obligés qui pourront affecter la structure du logiciel.

2.3.2 Contraintes d'évolution

Ce paragraphe précise si le logiciel est développé en prévoyant de le faire évoluer pour le réutiliser dans des applications similaires, ou si des évolutions sont d'ores et déjà prévues.

2.3.3 Contraintes de développement

Ce paragraphe décrit le matériel, le langage, les outils et les méthodes de développement.

2.3.4 Contraintes de qualité

Ce paragraphe liste les critères de qualité retenus pour déterminer si le logiciel est de qualité.

3 SPECIFICATIONS FONCTIONNELLES ET/OU OBJET

Ce paragraphe peut être élaboré suivant deux plans différents selon la méthodologie en vigueur : spécifications fonctionnelles ou spécifications "objet". On peut aussi donner les deux types de spécification (fonctionnelle et objet). Cela permet d'être plus complet en approchant les besoins de l'utilisateur suivant deux points de vue complémentaires.

SPECIFICATIONS OBJET

3.1 Définition des acteurs

Définition des acteurs	Définition de chaque acteur, hiérarchies d'acteurs éventuellement construites, diagrammes d'interaction les concernant.

3.2 Analyse du domaine

Analyse du domaine	Identification des interactions entre acteurs et cas d'utilisation. Identification des objets métier.

3.3 Cas d'utilisation

Les cas d'utilisation pourront être organisés par fonctionnalités ou exigences.

Modèle par cas d'utilisation	La description des cas d'utilisation se basera sur le modèle fourni en annexe. Elle pourra être organisée par paquetage. Pour chaque cas d'utilisation, on inclura les diagrammes utilisés : diagrammes de séquence, diagrammes de collaboration, diagrammes partiels de classes (si besoin).

FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

3.3.x Cas d'utilisation x

3.4 Paquetages

Architecture générale du système	On détaillera la hiérarchie des packages le cas échéant, ainsi que les modèles de conception appliqués.
Description des paquetages	Rôles et contenu principal de chaque package.
Interactions entre paquetages	Diagrammes de packages, diagrammes de séquence ou de collaboration entre packages.

SPECIFICATIONS FONCTIONNELLES

Ce paragraphe peut être élaboré avec l'aide d'une méthode de spécification de logiciel.
L'exemple de plan qui suit s'appuie sur les principes de décomposition fonctionnelle de SADT.

Structure du logiciel

Il s'agit de découper le logiciel en modules regroupant des fonctionnalités à forte cohérence.
Schéma représentant les relations entre les modules.

3.x Module x

Présentation du module et rapide résumé des fonctionnalités qu'il regroupe.

3.x.y Fonction y

3.x.y.1 Présentation de la fonction

Ce paragraphe présente la fonction d'un point de vue synthétique.

3.x.y.2 Entrées

Ce paragraphe liste les informations en entrées de la fonction.

3.x.y.3 Traitements

Ce paragraphe liste les traitements effectués.

3.x.y.4 Sorties

Ce paragraphe liste les informations produites par la fonction.

4 SPECIFICATIONS D'INTERFACE

Les interfaces du logiciel peuvent être de plusieurs types : Homme/Logiciel, Logiciel/Logiciel, / logiciel/Matériel. Ce paragraphe permet de les spécifier.

4.1 Interface Homme Machine

Ce paragraphe a pour objet de décrire l'interface Homme/Machine, selon les points suivants :

4.1.1 Prototypage de l'interface

Ce paragraphe contient une description exhaustive de l'interface homme machine et de chacun de ses composants, y compris leurs fonctions et leurs restrictions.

AUTRE FORMAT TYPE DE SPECIFICATION FONCTIONNELLE

4.1.2 Enchaînement des fenêtres

Ce paragraphe décrit l'enchaînement des fenêtres dans un contexte d'utilisation normale et anormale (messages d'erreurs, etc.).

4.1.3 Contraintes d'utilisation de l'interface

Ce paragraphe contient les restrictions dues aux contraintes d'utilisation éventuelles.

4.2 Interfaces Logiciel/Logiciel

Les interfaces Logiciel/Logiciel peuvent être de deux types:

- *relation d'utilisation entre deux modules logiciels,*
- *communication entre deux logiciels qui s'exécutent en parallèle*

4.2.1 Interfaces d'utilisation

4.2.1.x Interface x

Ce paragraphe décrit les primitives de l'interface avec leur type, leurs arguments, leur valeur retournée, leur conditions d'utilisation et/ou les classes d'interface et de communication.

4.2.2 Interfaces de communication

4.2.2.x Interface x

4.2.2.x.1. But et informations échangées

Ce paragraphe décrit le but de l'échange d'informations, la signification, le sens, la taille et la fréquence des informations échangées.

4.2.2.x.2. Procédures d'échange

Ce paragraphe décrit la manière dont on va échanger les informations (synchronisation, protocole, format des données, condition de déclenchement, temps de réponse).

4.3 Interface Logiciel/Matériel

Ce paragraphe décrit les instructions du matériel utilisés si cela est nécessaire, les signaux envoyés par le matériel, les retour d'erreurs.

5 TRACABILITE

Ce paragraphe décrit les éléments et la matrice de traçabilité entre les exigences utilisateur et les éléments décrits dans la spécification fonctionnelle ou objet.

Exigences / cas d'utilisatio	Matrice de traçabilité entre les exigences utilisateur et les éléments d'analyse identifiés lors des activités A, B et C.

PARTIE 2

Analyse des besoins

Les exigences

Laboratoire Informatique Image Interaction (L3I)
Université de La Rochelle - Pôle Sciences et Technologie - Avenue Michel Crépeau - 17042 LA ROCHELLE CEDEX 1 France
Tél : +33 (0)5 46 45 82 62 – Fax : 05.46.45.82.42 – Site internet : http://l3i.univ-larochelle.fr/

CAPTURER LES BESOINS

→ Première étape de la construction d'un système

Exprimer les besoins/les exigences

- Exprimer les besoins des utilisateurs (non-informaticiens)
- Définir les grandes fonctionnalités du système

Synthétiser les besoins : le diagramme de cas d'utilisation

- Synthétiser la vision utilisateur du système
- Regrouper les fonctionnalités
- Emerge des entretiens avec les utilisateurs

LES EXIGENCES

« Quelque chose que le produit doit faire ou une qualité qu'il doit avoir » (cf Suzanne et Jim Robertson)

« Une description du comportement attendu du système sur les opérations, les propriétés du système, etc.. »

« une exigence spécifie comment un but doit être satisfait par le système ».

Une exigence est donc une **assertion prescriptive que doit satisfaire le système,** par exemple : « *les contraintes d'un participant invité à une réunion doivent être demandées à celui-ci dans un délai fixé* ».

Elle peut adresser différents aspects du système que l'on veut contraindre et exprimer, par exemple :

–une **contrainte**: *le système doit inclure un vérificateur orthographique* ;

– une **fonction requise** : *le système doit identifier les « clients les plus anciens » ;*

– une **qualité attendue** : *la carte de vœux aux « clients les plus anciens » doit être imprimée en couleur.*

FORMULATION DES EXIGENCES

Une exigence est formulée en termes de **phénomènes de l'environnement organisationnel et dans le vocabulaire des acteurs** impliqués.

Une exigence est une **expression en langage naturel** qui, souvent, commence par « le système doit... ». Par exemple, « le système doit garder trace des clients fidèles ».

Une exigence peut se **décomposer en sous-exigences**. La formulation des exigences peut donc être **hiérarchique**.

Par exemple, dans le cas de réservations de chambres d'hôtels, l'exigence R1 suivante :

- *R1 : «Le système doit prendre les demandes de réservations faites par les clients de manière automatique » se décompose en :*
  - *R1.1 : Générer une réservation coïncidant avec la demande,*
  - *R1.2 : Offrir le passage sur la liste d'attente,*
  - *R1.3 : Proposer une réservation alternative et similaire.*

EXIGENCES FONCTIONNELLES ET NON-FONCTIONNELLES

Les **exigences fonctionnelles** définissent les services à fournir par le système en terme de fonctionnalités prévues.

Les **exigences non-fonctionnelles** contraignent la manière dont le système doit satisfaire les exigences fonctionnelles, ou la manière de le développer. Elles se répartissent en diverses sous-catégories :

- **les exigences de qualité de service** (sûreté, sécurité, utilisabilité, performance, précision des représentations logicielles par rapport à leur contrepartie dans l'environnement, etc.) ;
- **les contraintes juridictionnelles** (obligations légales, réglementations, normes imposées);
- **les contraintes architecturales** (répartition géographique des composants logiciels eu égard à la structure de l'organisation, à la répartition des données à utiliser ou des appareillages à contrôler, contraintes induites par la plate-forme d'implémentation, interopérabilité, etc.) ;
- **les exigences de développement** (coûts, délais, réutilisabilité, adaptabilité et extensibilité, etc.).

PROPRIÉTÉS DES EXIGENCES

les exigences ne sont pas des propriétés du domaine ;

les exigences ne sont pas des spécifications de logiciel ;

les exigences sont des formulations du problème, non des formulations d'une solution ;

les exigences ne sont pas la simple traduction du cahier des charges initial ;

les exigences doivent être « précises » ce qui ne veut pas dire «formelles».

→ ***Nécessité de d'organiser les exigences***

LE DIAGRAMME DES EXIGENCES - SYSML

Qu'est ce que SysML

SysML est un langage de modélisation graphique développé par l'OMG, INCOSE et AP233.

SysML est un profil d'UML 2.0 fournissant aux ingénieurs un langage de modélisation allant bien au delà des problématiques de l'informatique.

La généralisation des concepts utilisés en UML enrichis de quelques notions donnent SysML. Comme vous pouvez le constater sur le schéma ci-dessous. Nous retrouvons des diagrammes de structure (diagrammes statiques) et des diagrammes de fonctionnement (diagrammes dynamiques) comme en UML.

Comme UML, SysML est un langage et non une méthode, nous allons donc reprendre le processus 2TUP pour utiliser SysML.

SysML est fait pour :

- Spécifier les systèmes.
- Analyser la structure et le fonctionnement des systèmes.
- Décrire les systèmes et concevoir des systèmes composés de sous systèmes.
- Vérifier et valider la faisabilité d'un système avant sa réalisation.

SysML intègre :

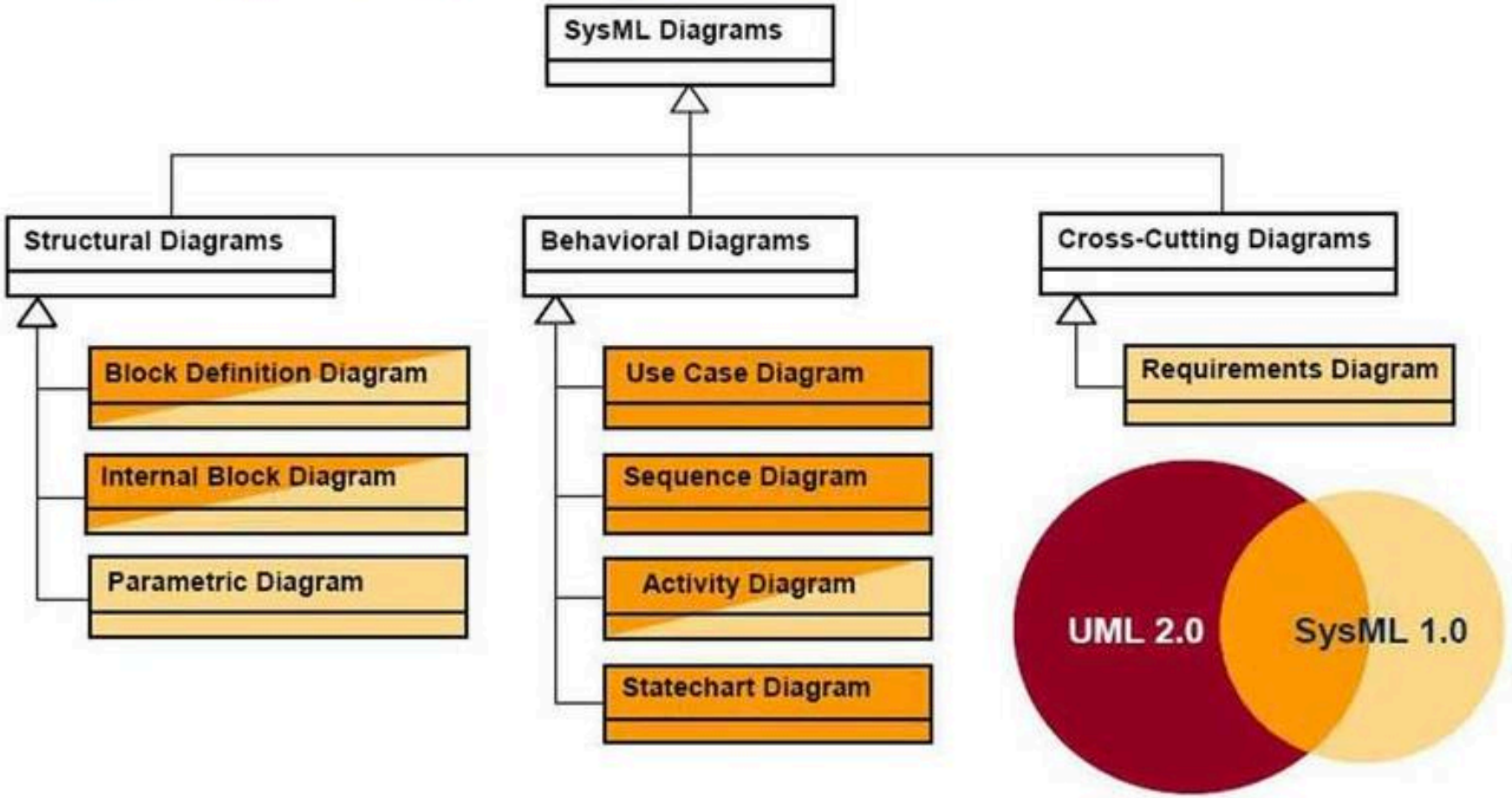
- Les composants physiques de toutes technologies.
- Les programmes.
- Les données et les énergies.
- Les personnes.
- Les procédures et flux divers.

http://www.uml-sysml.org/sysml

LE DIAGRAMME DES EXIGENCES - SYSML

Architecture SysML vs UML

Ci-dessous la description extraite de la spécification officielle de SysML.



http://www.uml-sysml.org/sysml

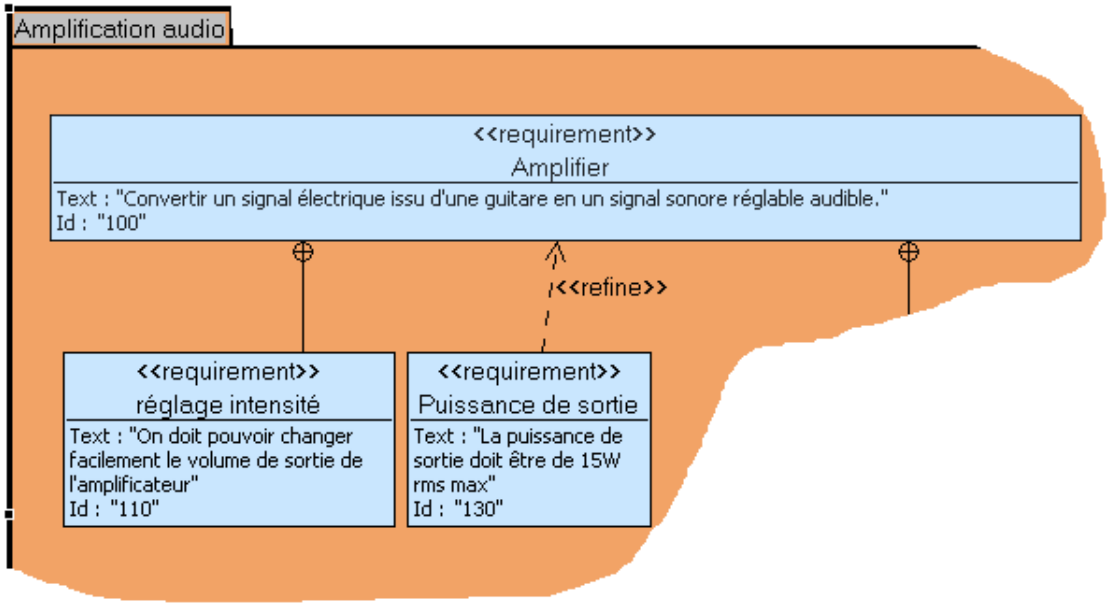
LE DIAGRAMME DES EXIGENCES - SYSML

Diagramme d'exigences (Requirement diagram)

Le diagramme d'exigence décrit graphiquement une capacité ou une contrainte qui doit être satisfaite par un système. C'est une interprétation du cahier des charges.

Un diagramme d'exigence, placé dans un cartouche de type ***req***, comporte :

- Des exigences décrites dans un cadre d'exigence (Requirement)
- Des liaisons
- Parfois le diagramme est inclus dans un package.



Exigence

Chaque cadre d'exigence dispose de trois informations au moins :

- un nom : il s'agit d'une chaîne de caractères décrivant l'exigence. Elle doit être courte précise et de préférence unique
- le texte de description: il décrit et précise l'exigence
- l'identifiant : il doit être unique et de préférence hiérarchisé. Un diagramme SysML pouvant être compilé pour générer un programme une duplicité d'identifiant générerait automatiquement une erreur.

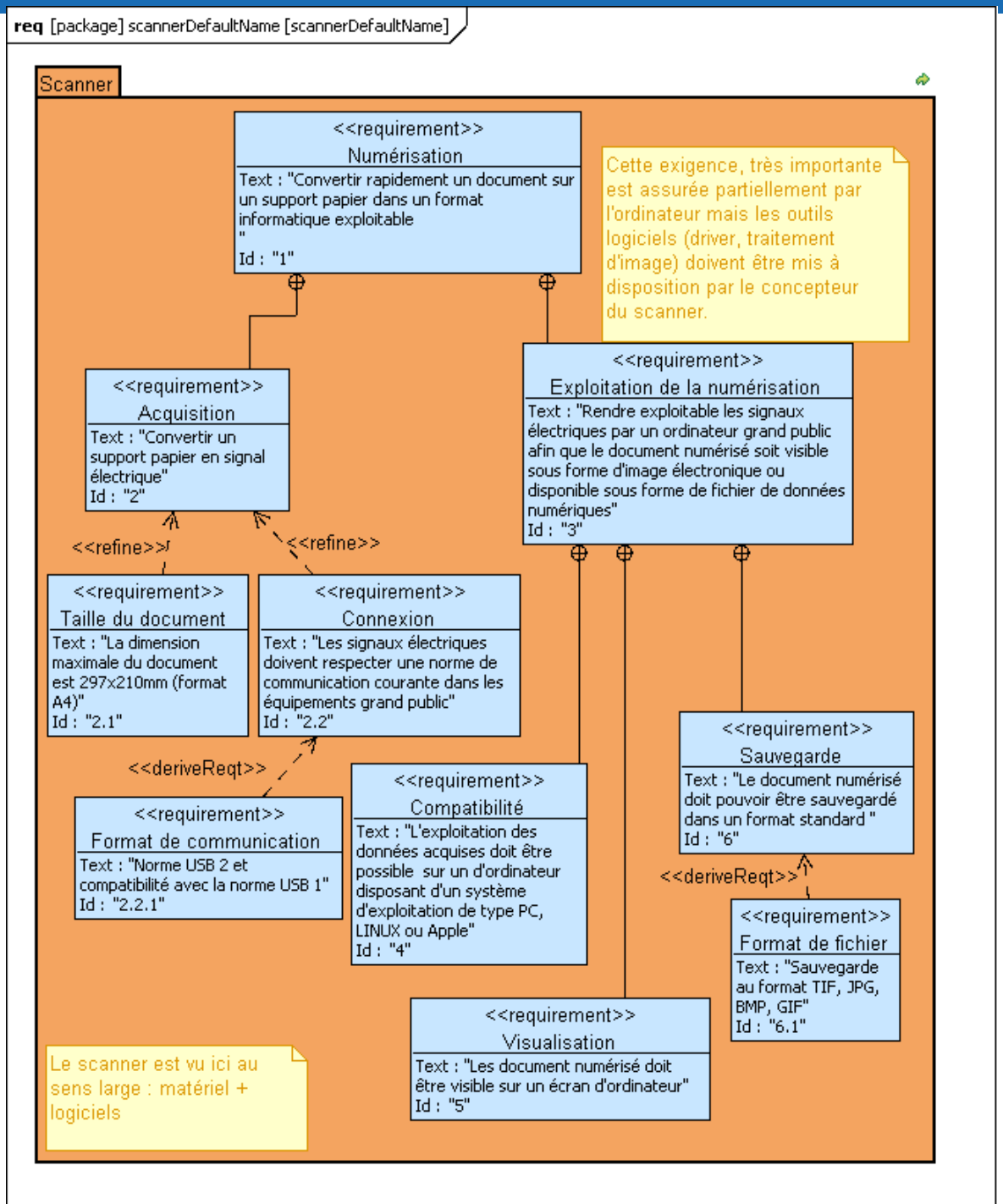
http://www.siloged.fr/cours/STI2D_sysml/index.html?Diagrammedexigence.html

LE DIAGRAMME DES EXIGENCES - LIAISONS

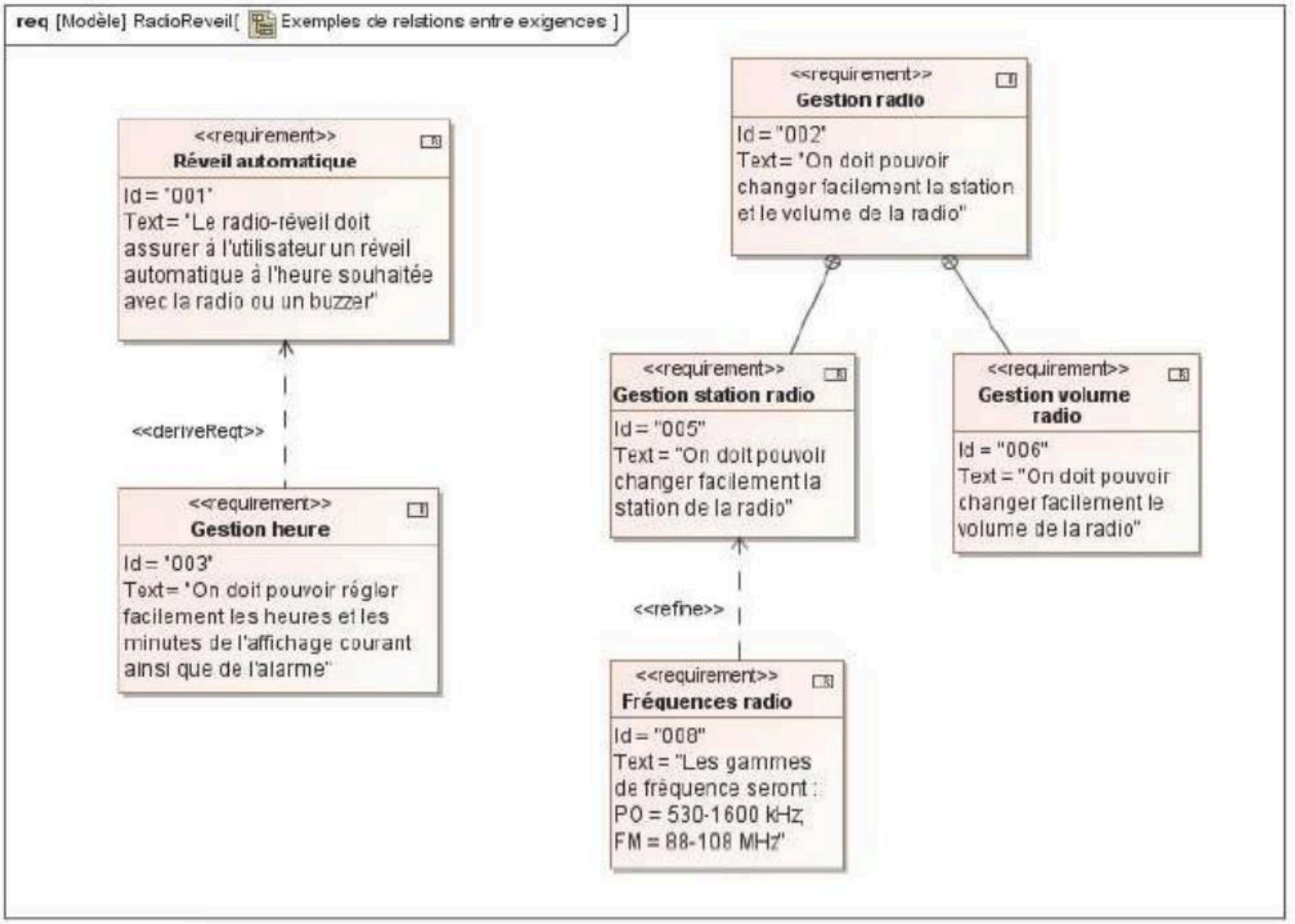
3 types de liaisons sont vraiment fondamentales :

- la **contenance** (ligne terminée par un cercle contenant une croix du côté du conteneur) permet de décomposer une exigence composite en plusieurs exigences unitaires, plus faciles ensuite à tracer vis-à-vis de l'architecture ou des tests ;
- le **raffinement** (« refine ») consiste en l'ajout de précisions, par exemple de données quantitatives ;
- la **dérivation** (« deriveReqt ») consiste à relier des exigences de niveaux différents, par exemple des exigences système à des exigences de niveau sous-système, etc. Elle implique généralement des choix d'architecture.

EXEMPLE


req [package] scannerDefaultName [scannerDefaultName]
Scanner
<<requirement>>
Numérisation
Text : "Convertir rapidement un document sur un support papier dans un format informatique exploitable
"
Id : "1"
Cette exigence, très importante est assurée partiellement par l'ordinateur mais les outils logiciels (driver, traitement d'image) doivent être mis à disposition par le concepteur du scanner.
<<requirement>>
Acquisition
Text : "Convertir un support papier en signal électrique"
Id : "2"
<<requirement>>
Exploitation de la numérisation
Text : "Rendre exploitable les signaux électriques par un ordinateur grand public afin que le document numérisé soit visible sous forme d'image électronique ou disponible sous forme de fichier de données numériques"
Id : "3"
<<refine>>
<<refine>>
<<requirement>>
Taille du document
Text : "La dimension maximale du document est 297x210mm (format A4)"
Id : "2.1"
<<requirement>>
Connexion
Text : "Les signaux électriques doivent respecter une norme de communication courante dans les équipements grand public"
Id : "2.2"
<<requirement>>
Sauvegarde
Text : "Le document numérisé doit pouvoir être sauvegardé dans un format standard "
Id : "6"
<<deriveReqt>>
<<deriveReqt>>
<<requirement>>
Format de communication
Text : "Norme USB 2 et compatibilité avec la norme USB 1"
Id : "2.2.1"
<<requirement>>
Compatibilité
Text : "L'exploitation des données acquises doit être possible sur un d'ordinateur disposant d'un système d'exploitation de type PC, LINUX ou Apple"
Id : "4"
<<requirement>>
Format de fichier
Text : "Sauvegarde au format TIF, JPG, BMP, GIF"
Id : "6.1"
<<requirement>>
Visualisation
Text : "Les document numérisé doit être visible sur un écran d'ordinateur"
Id : "5"
Le scanner est vu ici au sens large : matériel + logiciels

AUTRE EXEMPLE



https://cahier-de-prepa.fr/CoeffinPTSI/download?id=957

REQUIREMENT DIAGRAM - VISUAL PARADIGM

Requirement diagram

Requirement diagram lets you visualize system functions as well as the ways to test the functions. This chapter teaches you how to work with requirement diagram.

Drawing requirement diagrams

This page shows you how to create requirements in requirement diagram, specify requirements body, relate requirements and create test cases.

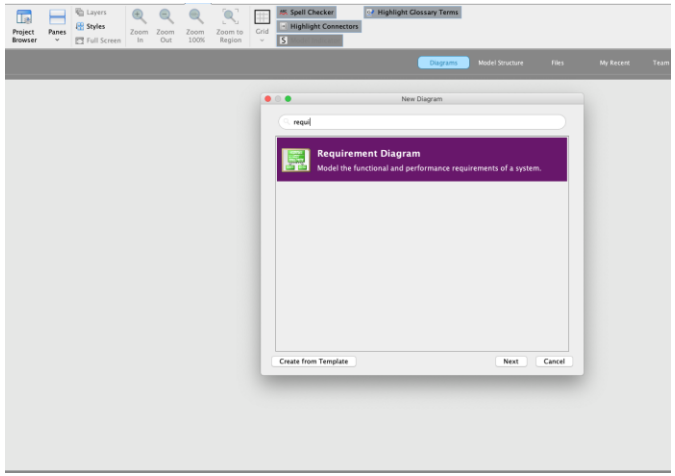
Customizing requirement types

You will see how to define your own requirement type in this page.

Modeling and documenting test cases

Make use of the test case element and its test plan editor to model the test case of requirements.

https://www.visual-paradigm.com/support/documents/vpuserguide/94/158_requirementd.html

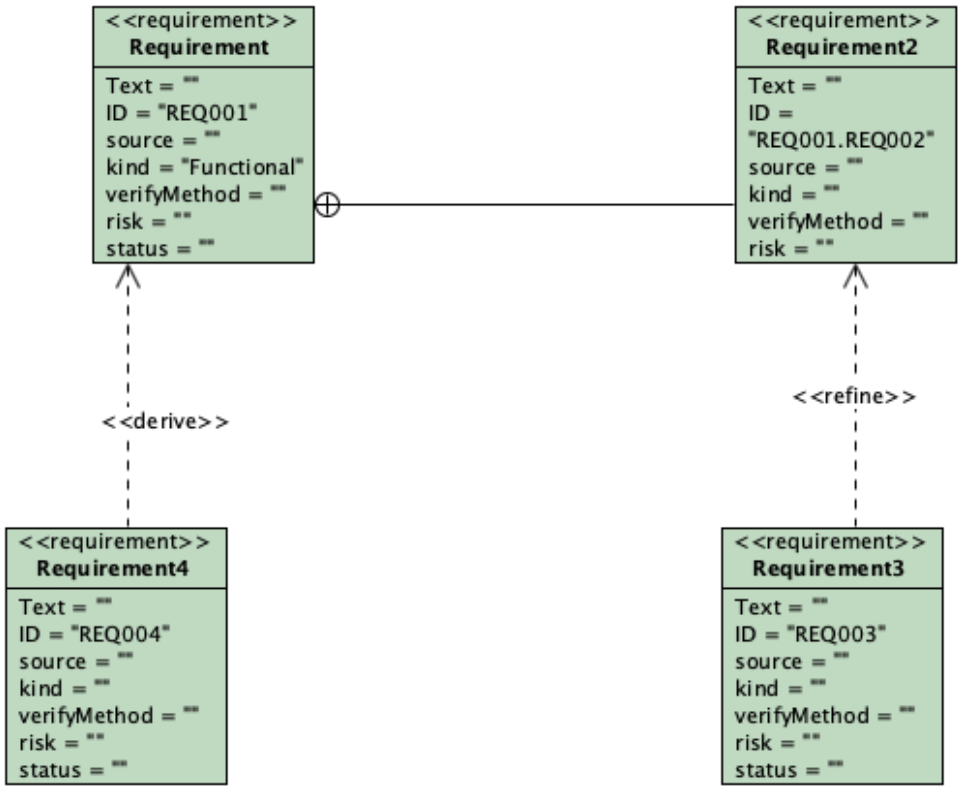


LES 3 TYPES D'EXIGENCES - VISUAL PARADIGM

Fonctionnelle

Performance

Interface



PARTIE 3

UML et l'analyse des besoins

Laboratoire Informatique Image Interaction (L3I)
Université de La Rochelle - Pôle Sciences et Technologie - Avenue Michel Crépeau - 17042 LA ROCHELLE CEDEX 1 France
Tél : +33 (0)5 46 45 82 62 – Fax : 05.46.45.82.42 – Site internet : http://l3i.univ-larochelle.fr/

LA MODÉLISATION UML

description abstraite d'un système ou d'un processus
représentation simplifiée qui permet de comprendre et de simuler

Modélisation fonctionnelle

Décompose les tâches en fonctions plus simples à réaliser

Exemple : MERISE

Modélisation objet

Décompose les systèmes en objets qui collaborent

Exemple : OMT (Object Modeling Technique)

UML est un langage universel (et non une méthode!) de modélisation objet pour :

la spécification,
la visualisation,
la construction et
la documentation de systèmes

langage servant à décrire des modèles d'un système (réel ou
concepts orienté-objets

UML: PRINCIPE

La notation U.M.L. consiste à proposer des graphes avec :

- Des symbolismes particuliers pour les nœuds, les arcs
- Des chaînes de caractères identifiants les nœuds ou les arcs (= noms), donnant des informations (= étiquettes) ou des mots clés

Les graphes s'accompagnent d'expressions (de contraintes par exemple)

- Textuelles
- Dans un langage de programmation quelconque
- Dans le langage O.C.L. (Object Constraint Language propre à U.M.L.)

Les éléments du modèle peuvent être regroupés dans des paquetages

A faire ….

comprendre et conceptualiser le problème (besoins, analyse)

résoudre le problème (conception)

donner une solution (implémentation)

documenter

UML: langage pour …

spécifier, visualiser et comprendre le problème

capturer, communiquer et utiliser des connaissances pour la résolution du problème

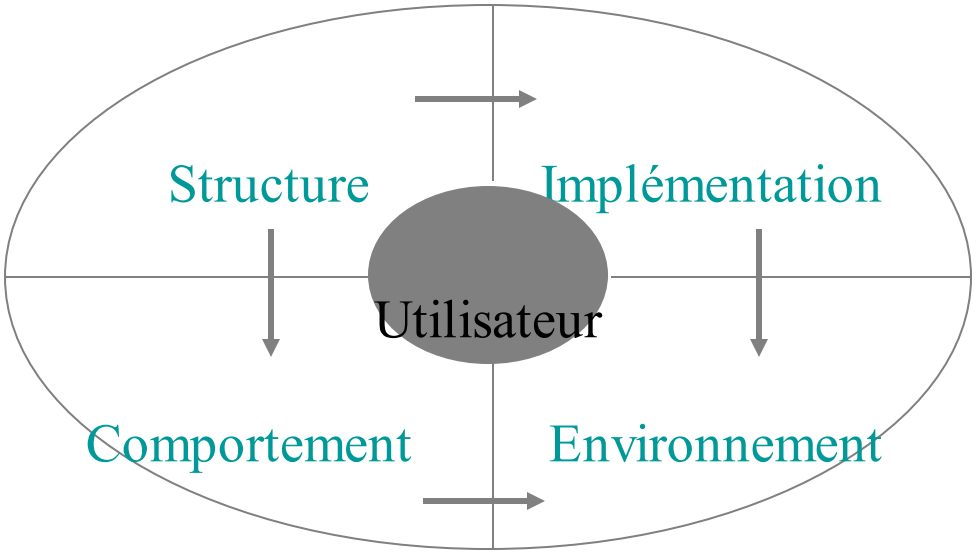
spécifier, visualiser, et construire la solution

documenter la solution

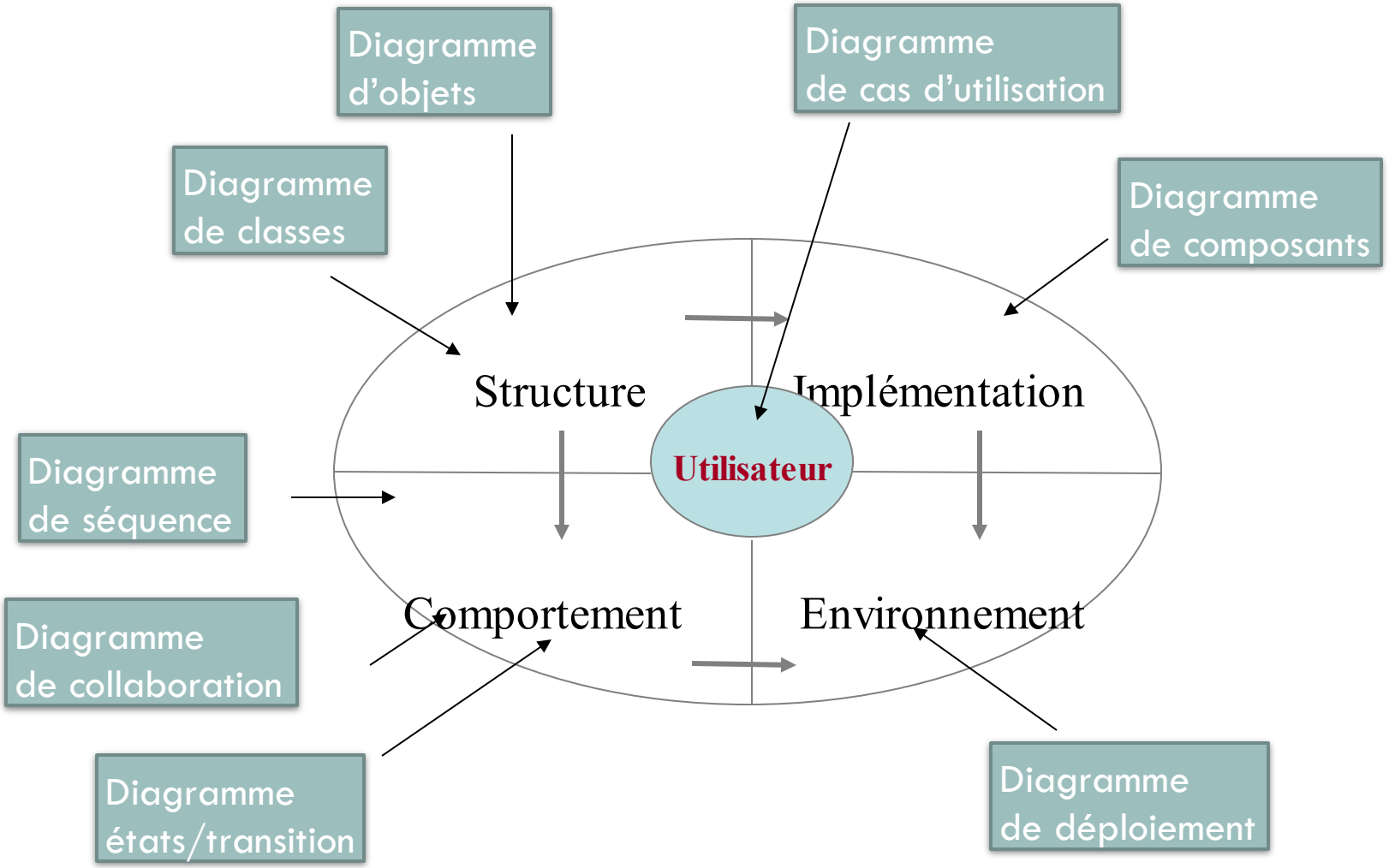
LES DIAGRAMMES UML

Les modèles du système sont visualisés par des **vues**

- Modéliser: créer différentes vues du système
- Chaque participant au développement voit le système différemment



LES DIAGRAMMES UML



MODÉLISATION OBJET

Un seul paradigme

- Même langage utilisé par utilisateurs, analystes, architectes, programmeurs

Modèles plus proches du monde réel

- Description plus précise des données et des processus
- Décomposition en partitions naturelles
- Plus faciles à comprendre et à maintenir

Ré-utilisation

- Architecture et code plus faciles à ré-utiliser

Stabilité

- Léger changement dans le domaine de l'application non ⇔ modifications massives dans le système

Les concepts:

Abstraction
Encapsulation
Modularité
Hiérarchie

ELEMENT DU DIAGRAMME DE CAS D'UTILISATION

Ce diagramme est construit en phase de définition des exigences fonctionnelles et enrichi pendant la phase d'analyse en utilisant d'autres modèles (entre-autres les modèles d'interaction).

Les objectifs sont :

- identifier les fonctionnalités du logiciel
- en définir le périmètre
- identifier les éléments externes en interaction directe

Eléments du modèle

- Acteurs
- Cas d'utilisation
- Relations entre acteurs et cas d'utilisation ou entre cas d'utilisation

Exemples issus du cours de C. Buche

ACTEUR

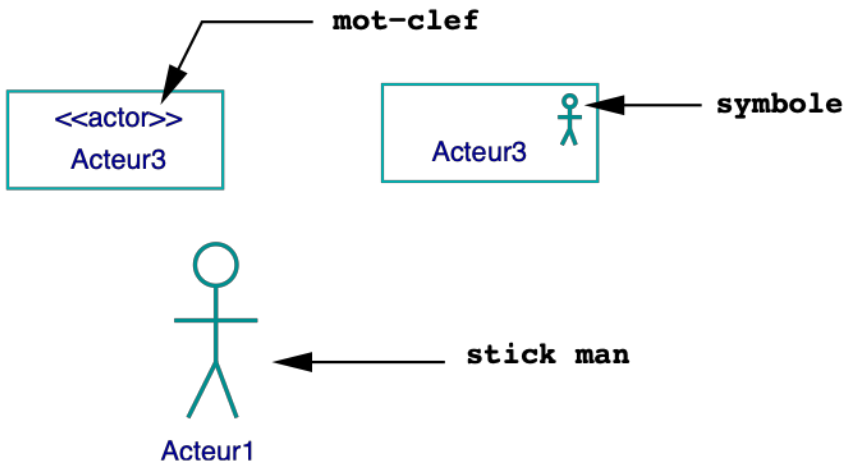
Rôle joué par une entité « externe » au système

Interagit avec le système au travers de messages

Il peut s'agir :

- d'utilisateurs humains
- de systèmes externes qui interagissent avec le système

2 représentations possibles



CAS D'UTILISATION

Ensemble de séquences d'actions réalisées par le système et qui produisent un résultat observable pour un acteur donné

Spécifie un comportement attendu

- Ce qui doit être fait
- Sans précision du « comment » cela doit être fait

Les cas d'utilisation correspondent aux exigences fonctionnelles

Un cas = une fonction métier

Méthodologie possible de construction:

- Pour chaque acteur, identifier les fonctions métiers
- Associer aux services attendus dans le cahier des charges
- Nommer les cas d'utilisation (**verbe à l'infinitif + complément**)

RELATIONS ENTRE ACTEUR ET CAS D'UTILISATION - ASSOCIATION

Chemin de communication entre un acteur et un cas d'utilisation

- Acteur : élément externe en communication directe avec le sujet (logiciel ou module du logiciel)
- Cas d'utilisation : ensemble fonctionnel cohérent

Indique la participation de l'acteur à la réalisation du cas.

Représentation de la frontière du système : cadre englobant les cas

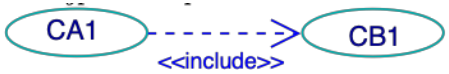
- acteurs à l'extérieur
- cas à l'intérieur

Nom du système : à l'intérieur du cadre, en haut

RELATIONS ENTRE CAS D'UTILISATION

Inclusion

- 1 réalisation de CA1 implique la réalisation de CA2



Extension

- Selon le contexte, la réalisation de CA1 peut entrainer la réalisation de CA2

Spécialisation

- Selon le contexte on réalise CA1 ou CA2

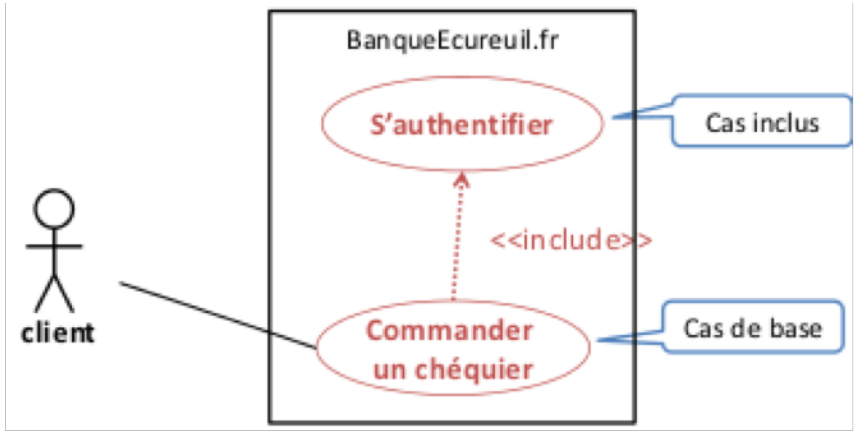
INCLUSION

Un cas A inclut un cas B si le comportement décrit par A inclut le comportement décrit par B

Le cas A dépend de B

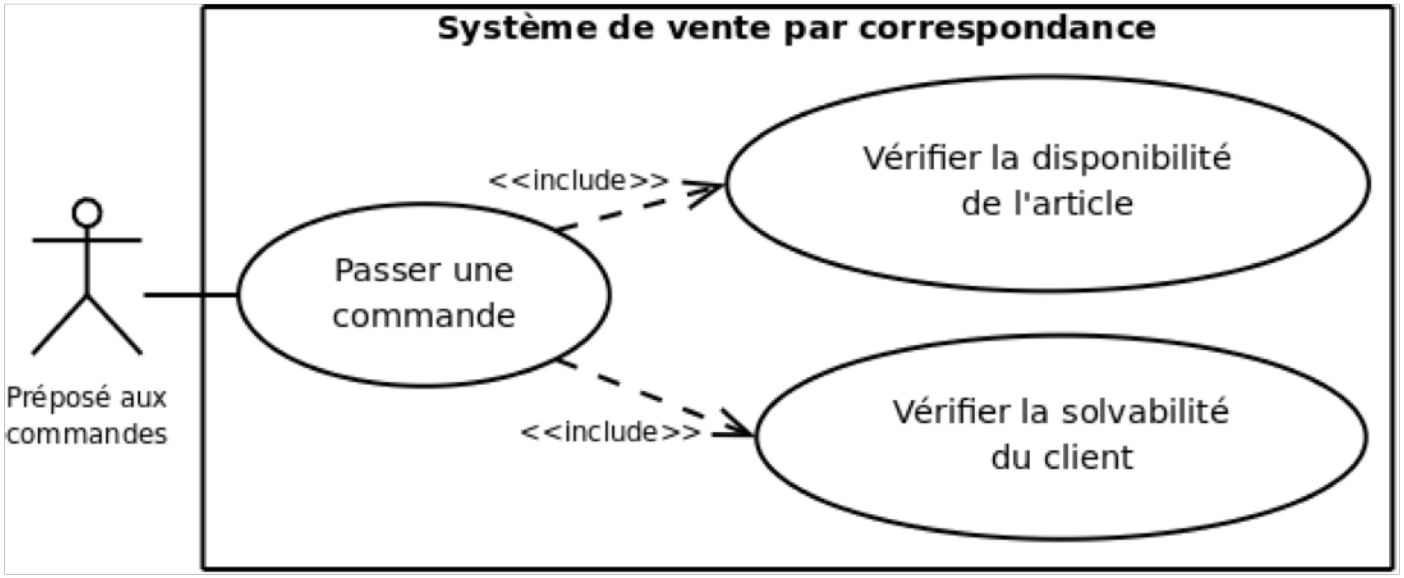
Lorsque A est initié, B l'est obligatoirement

Stéréotype << include >>



Permet de

- factoriser une partie de la description d'un cas d'utilisation qui pourrait être commune à d'autres cas
- décomposer un cas complexe en cas plus simples


Système de vente par correspondance
Vérifier la disponibilité de l'article
<<include>>
Passer une commande
Préposé aux commandes
<<include>>
Vérifier la solvabilité du client

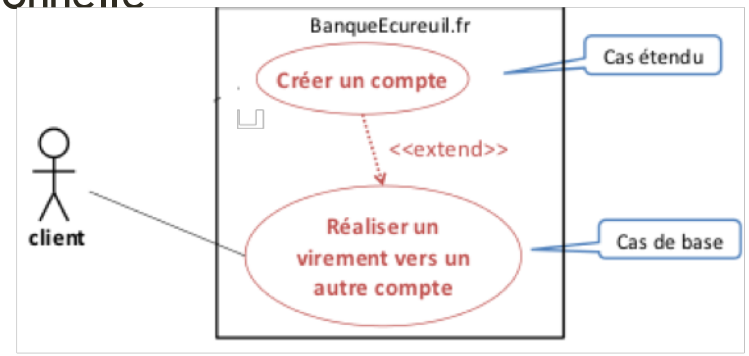
EXTENSION

Un cas A étend un cas B lorsque le cas A peut être appelé au cours de l'exécution du cas B.

Exécuter B peut éventuellement entrainer l'exécution de A

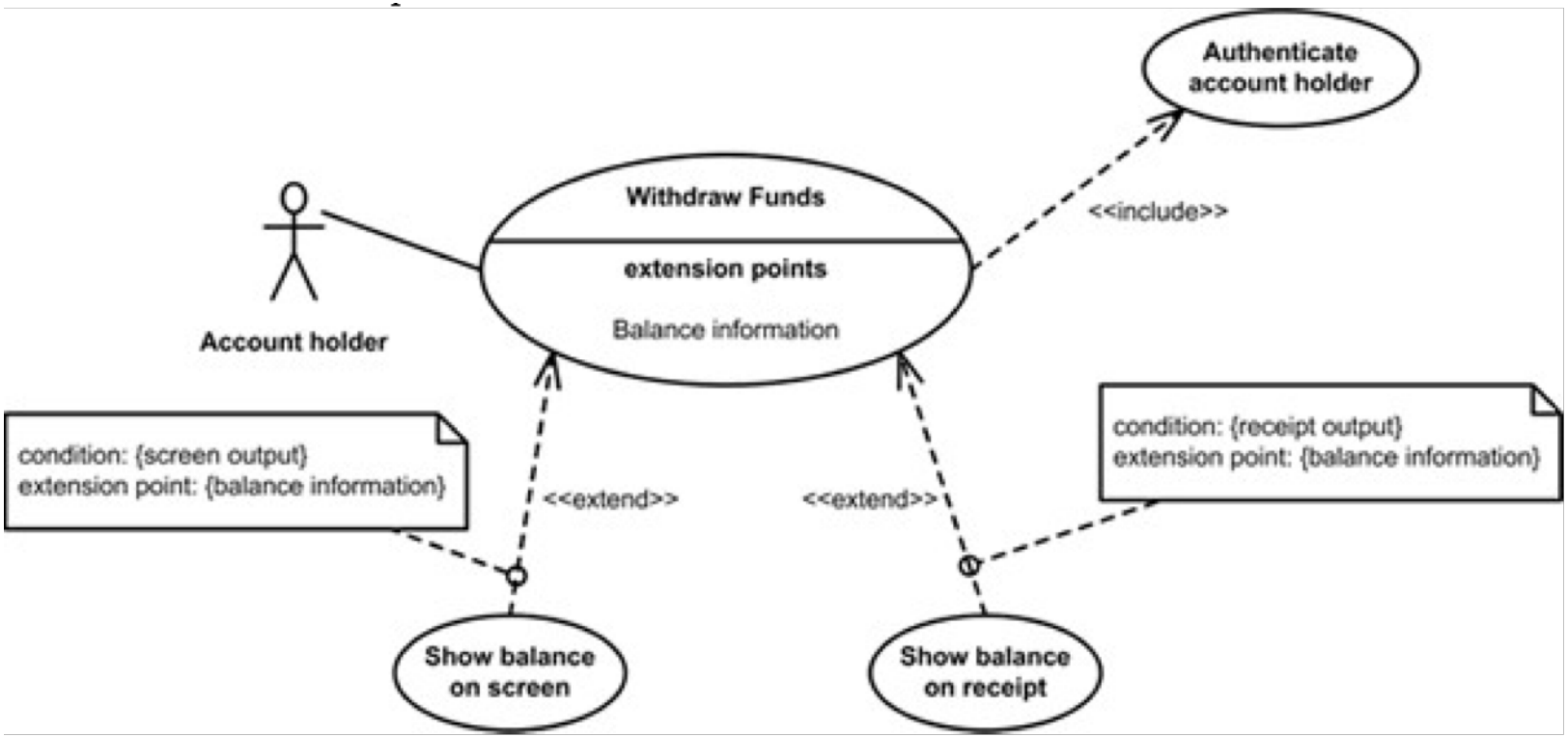
Contrairement à l'inclusion : l'extension est optionnelle

Stéréotype << étend >>



L'extension peut intervenir à un point précis du cas étendu

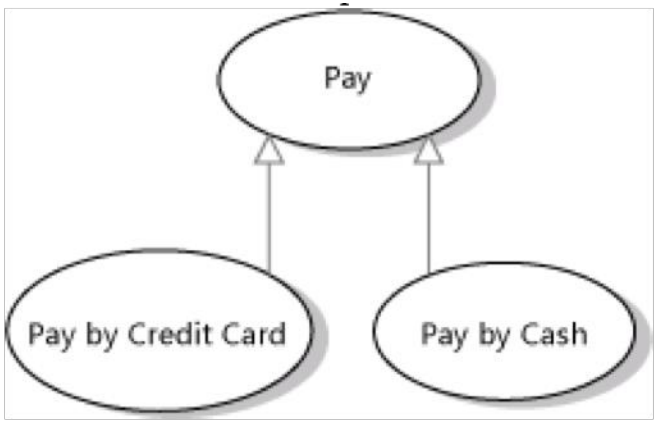
- C'est « le point d'extension »
- Il porte un nom qui figure dans un compartiment du cas étendu sous la rubrique « point d'extension »
- Il peut être associé à une contrainte (la condition est alors exprimée sous la forme d'une note)


Authenticate account holder
Withdraw Funds
extension points
Balance information
<<include>>
Account holder
condition: {screen output}
extension point: {balance information}
<<extend>>
<<extend>>
condition: {receipt output}
extension point: {balance information}
Show balance on screen
Show balance on receipt

SPÉCIALISATION

Un cas A est une généralisation d'un cas B si B est un cas particulier de A

Equivalent de l'héritage



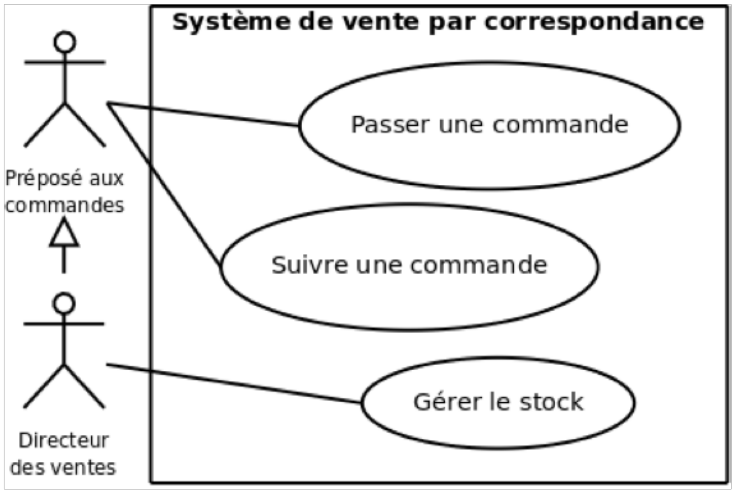
Les cas d'utilisation descendants héritent de la description de leur parent et peuvent comprendre des interactions spécifiques.

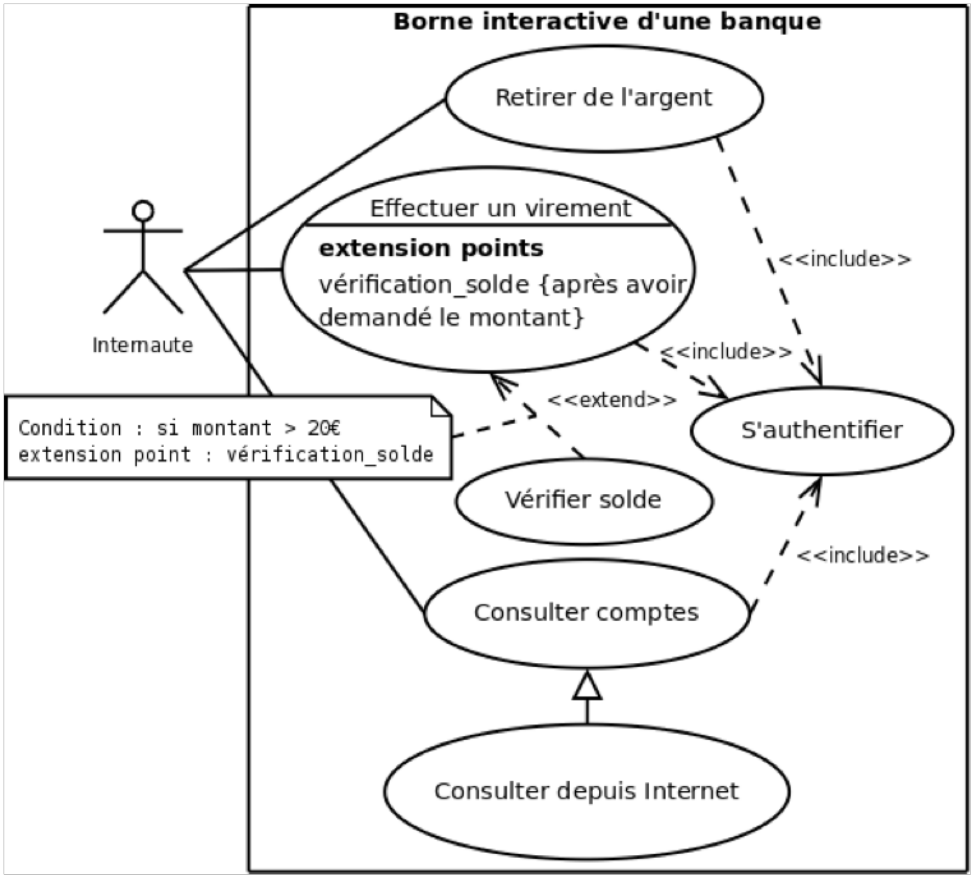
RELATIONS ENTRE ACTEURS : LA GÉNÉRALISATION

Un acteur A est une généralisation de l'acteur B si l'acteur A peut être « substitué » par l'acteur B

Tous les cas d'utilisation accessibles à A le sont aussi à B

Représenté par une relation d'héritage.




Borne interactive d'une banque
Retirer de l'argent
Effectuer un virement
extension points
vérification_solde {après avoir demandé le montant}
Internaute
<<include>>
<<include>>
<<extend>>
Condition : si montant > 20€
extension point : vérification_solde
S'authentifier
Vérifier solde
<<include>>
Consulter comptes
Consulter depuis Internet

ACTEUR PRINCIPAUX ET SECONDAIRES

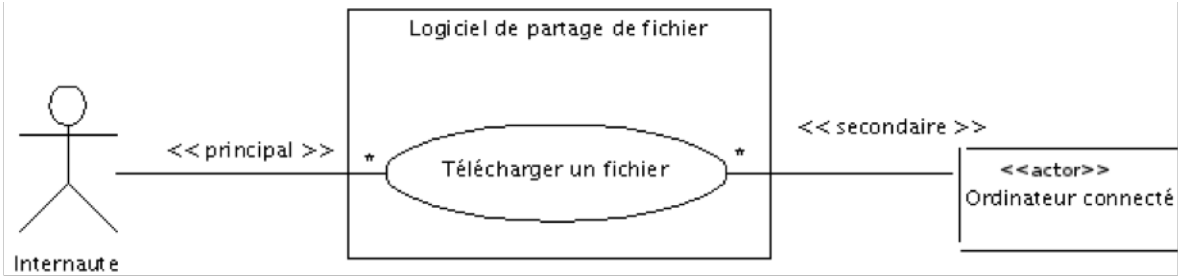
Acteur principal : lorsque le cas rend service à cet acteur

Un cas d'utilisation : au plus un acteur principal

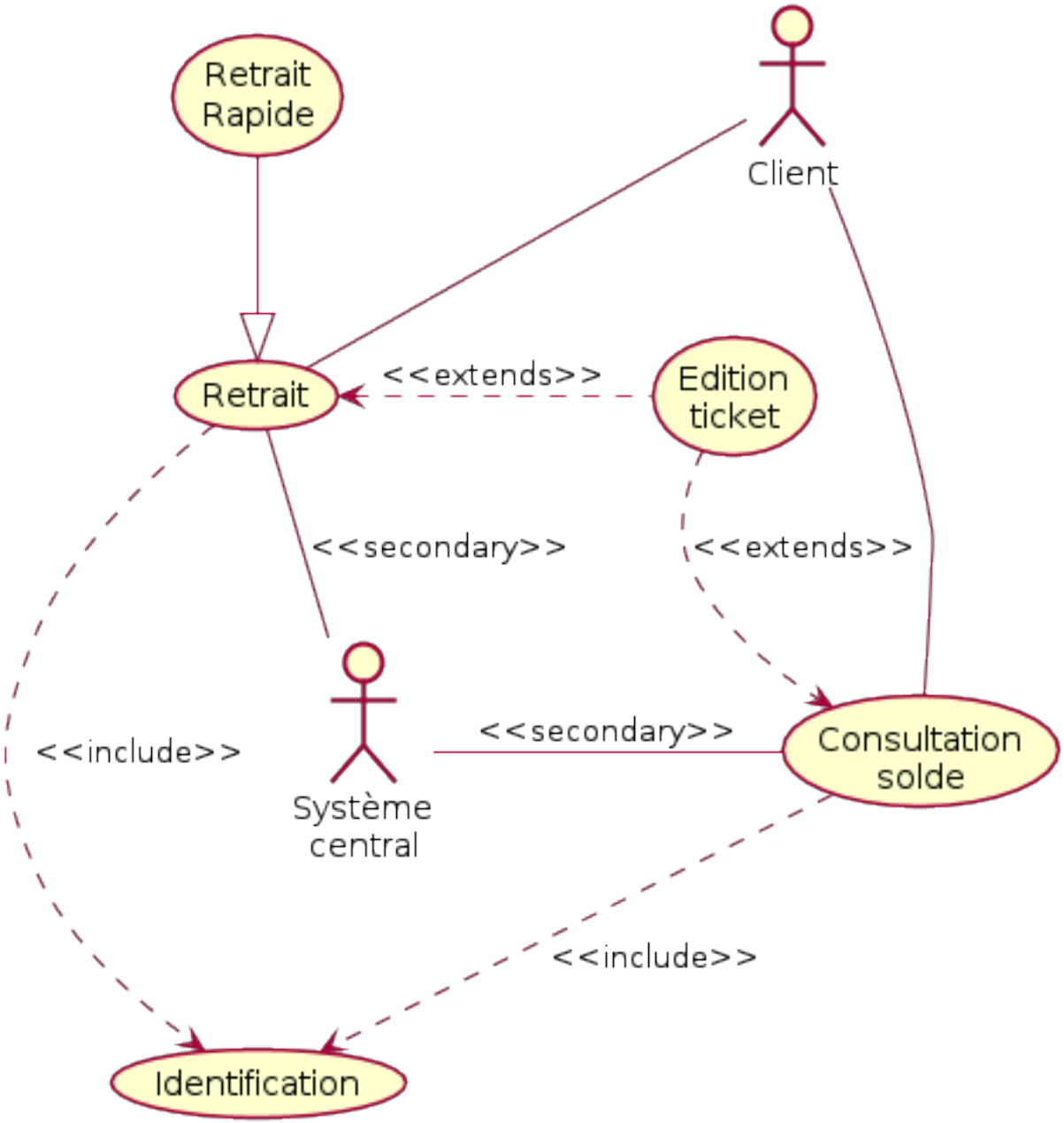
- Stéréotype << primary >>
- Il obtient un résultat observable du cas d'utilisation
- C'est lui qui initie le cas d'utilisation

Acteur secondaire : les autres acteurs

- Stéréotype << secondary >>
- Acteur sollicité pour des informations nécessaires au cas d'utilisation



EXEMPLE



https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-corr04.html

EXERCICE

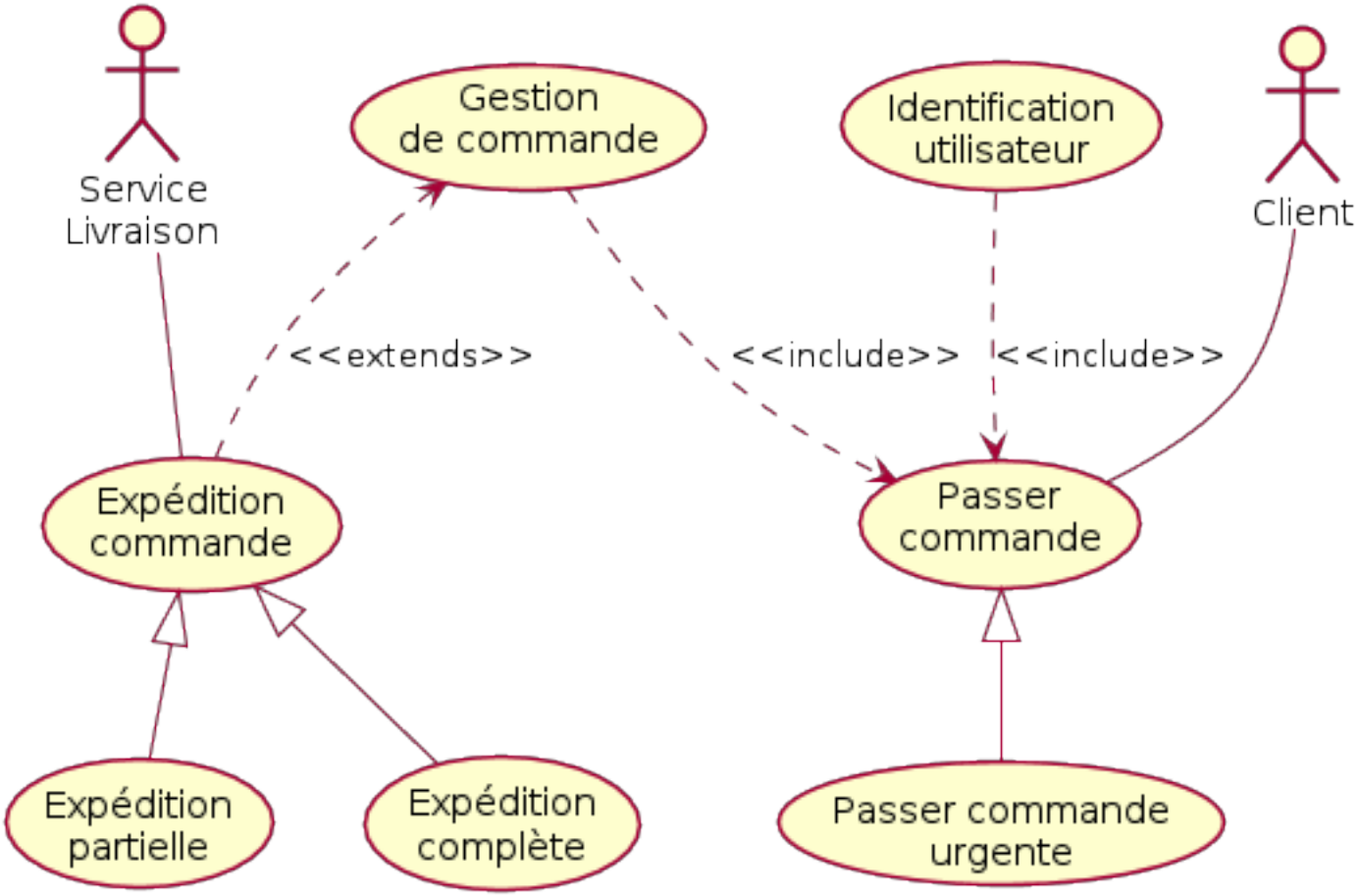
Organiser en diagramme de cas d'utilisation

Acteurs

- Service livraison
- Client

Cas d'utilisation

- Gestion de commande
- Identification utilisateur
- Expedition commande
- Passer commande
- Expedition partielle
- Expédition complète
- Passer commande urgente



https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-corr04.html