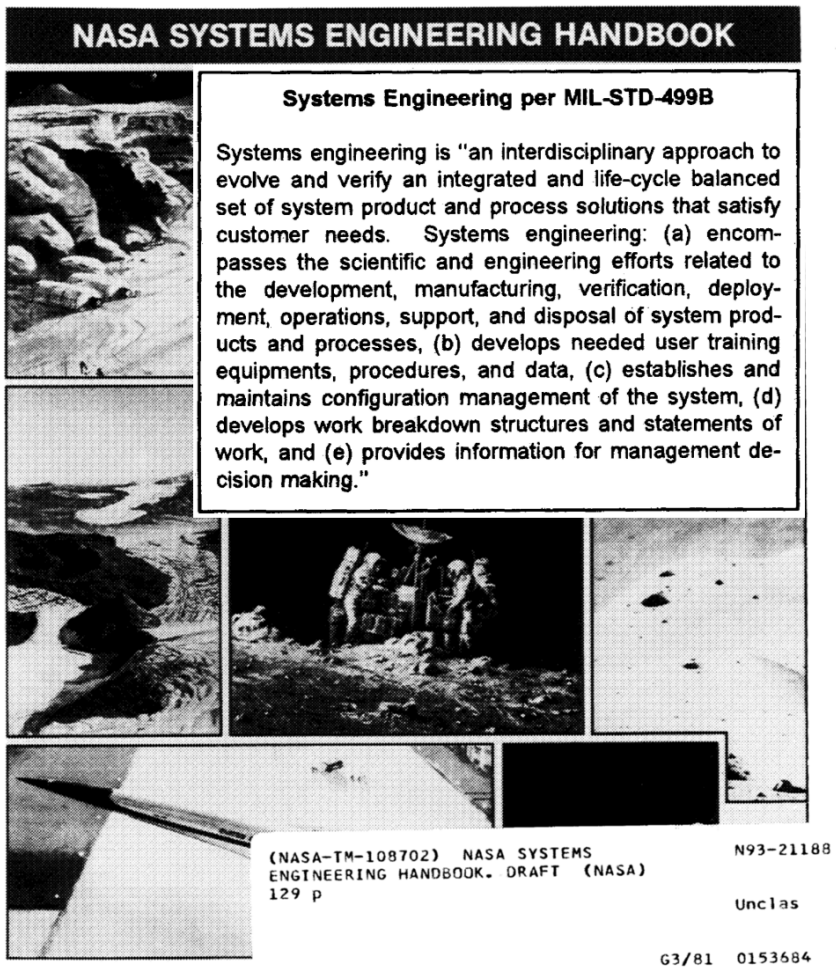
Développement logiciel et spécification

Projet de génie logiciel L3

Table des matières

Exemple de cycle de développement spatial



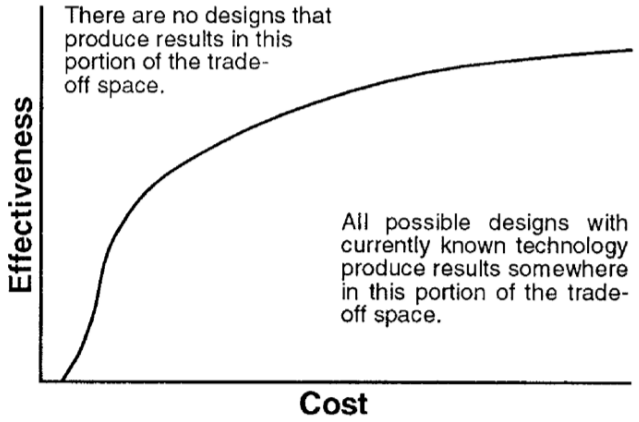


Figure 1 — The Enveloping Surface of Non-dominated Designs.

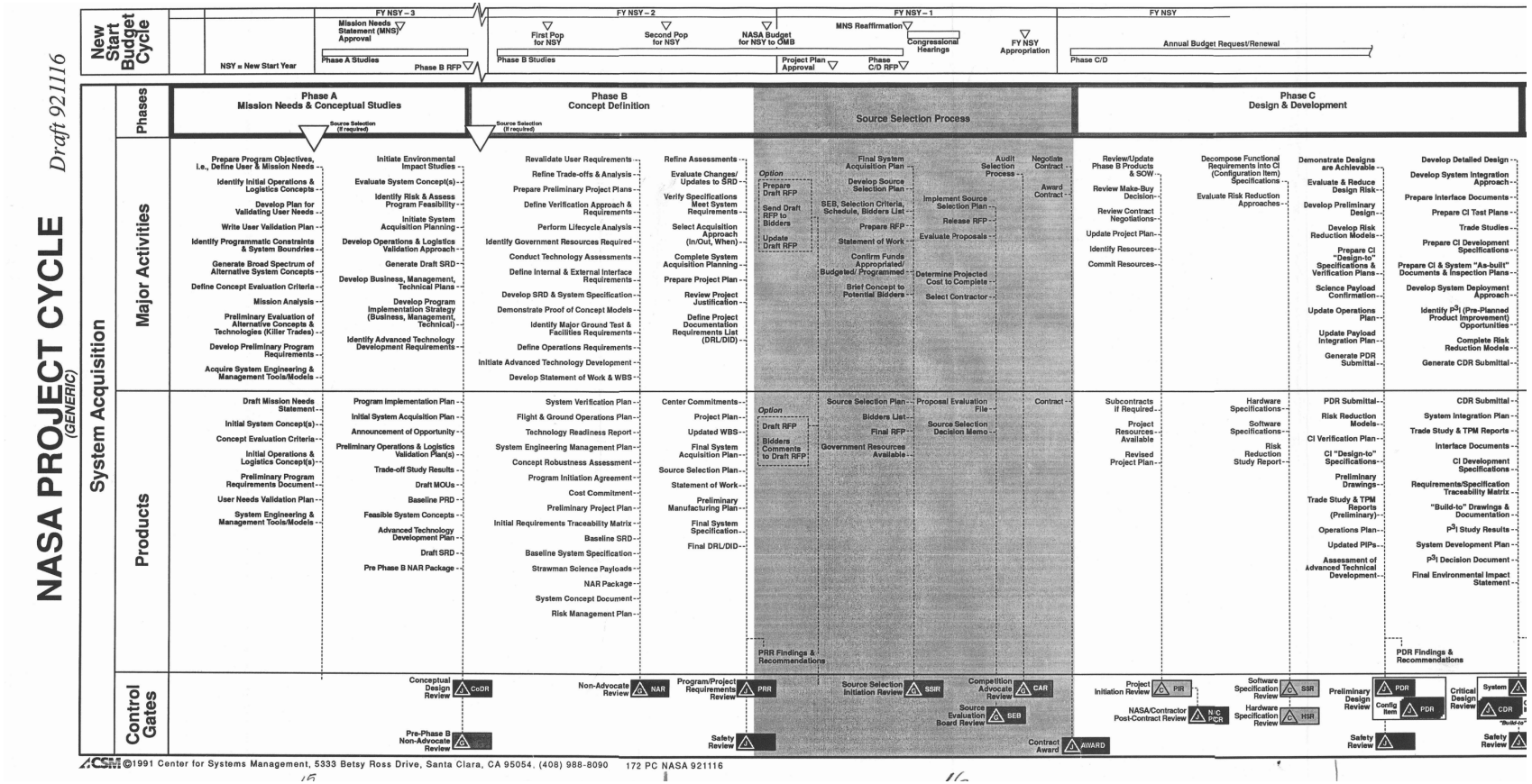
The System Engineer's Dilemma

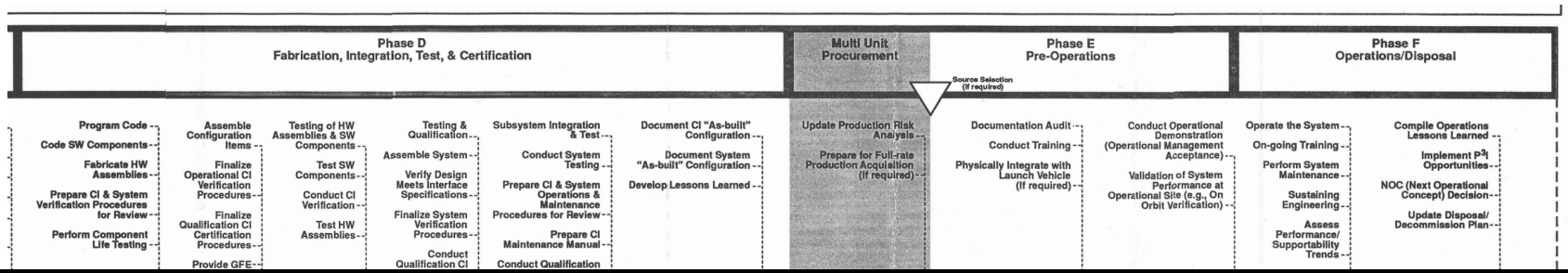
- To reduce cost at constant risk, performance must be reduced.
- To reduce risk at constant cost, performance must be reduced.
- To reduce cost at constant performance, higher risks must be accepted.
- To reduce risk at constant performance, higher costs must be accepted.

In this context, time in the schedule is often a critical resource, so that *schedule* behaves like a kind of *cost*.

NASA Systems Engineering Handbook, R. Shishko, R. G. Chamberlain, Sept '92

Exemple de cycle de développement spatial





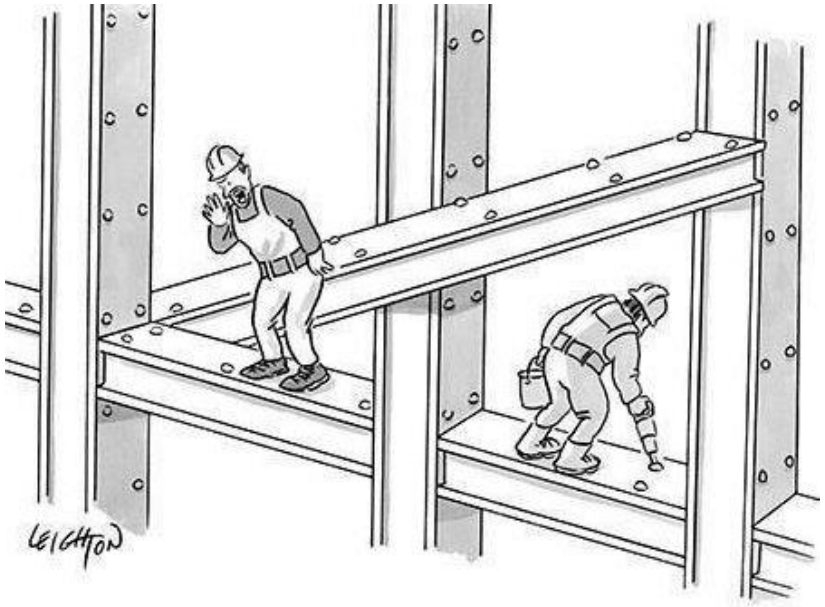
Crises du logiciel

- Conférence de l'OTAN à Garmish, Allemagne (1968)
  - L'informatique ne répond pas aux attentes qu'elle suscite
  - Coûte cher et désorganise les entreprises
- Etude du Standish Group (1995) sur 8 380 projets
  - Succès : 16%
  - Problématique : 53%
    (budget ou délais non respectés, fonctionalités manquantes)
  - Echec : 31%

Pourquoi faire du génie logiciel ?

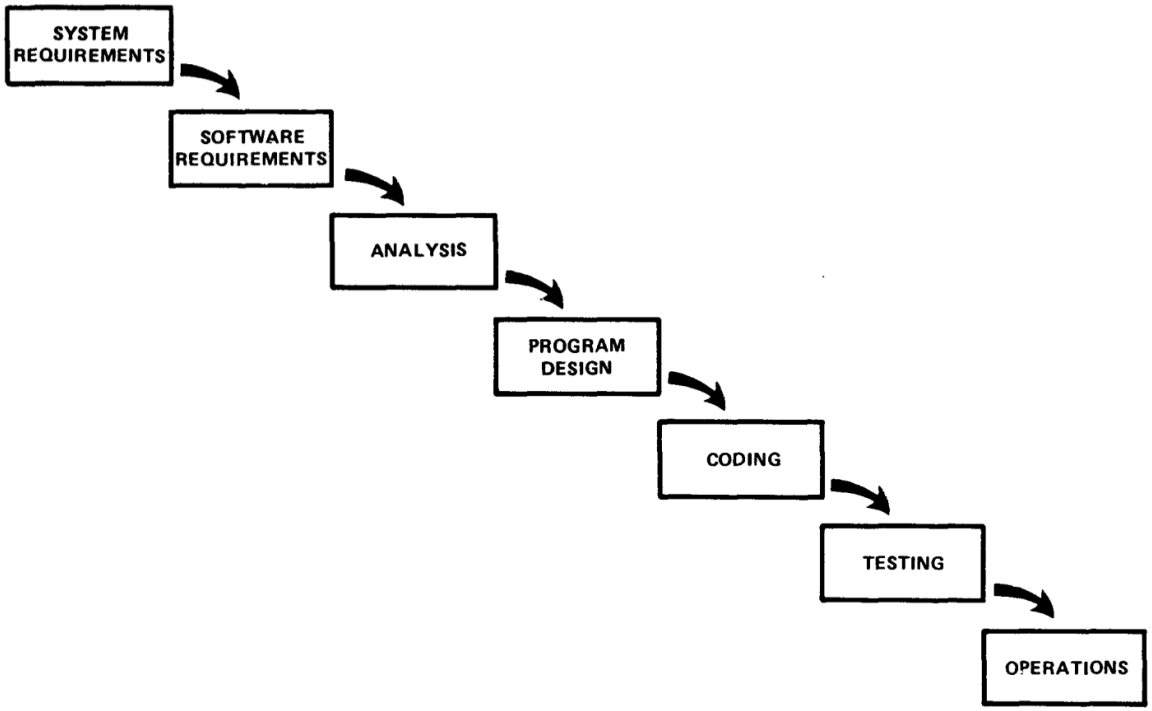
- Coordonner des **projets à grande échelle**
- **Communiquer** avec le client
- Produire un livrable **conforme**
- Respecter les **délais et coûts**
- Maintenir un produit sur le **long terme**

Quelques méthodologies du développement logiciel



- Développement en cascade
- Développement du cycle en V
- Développement Agile
- ...

Développement en cascade (Royce, 1970)



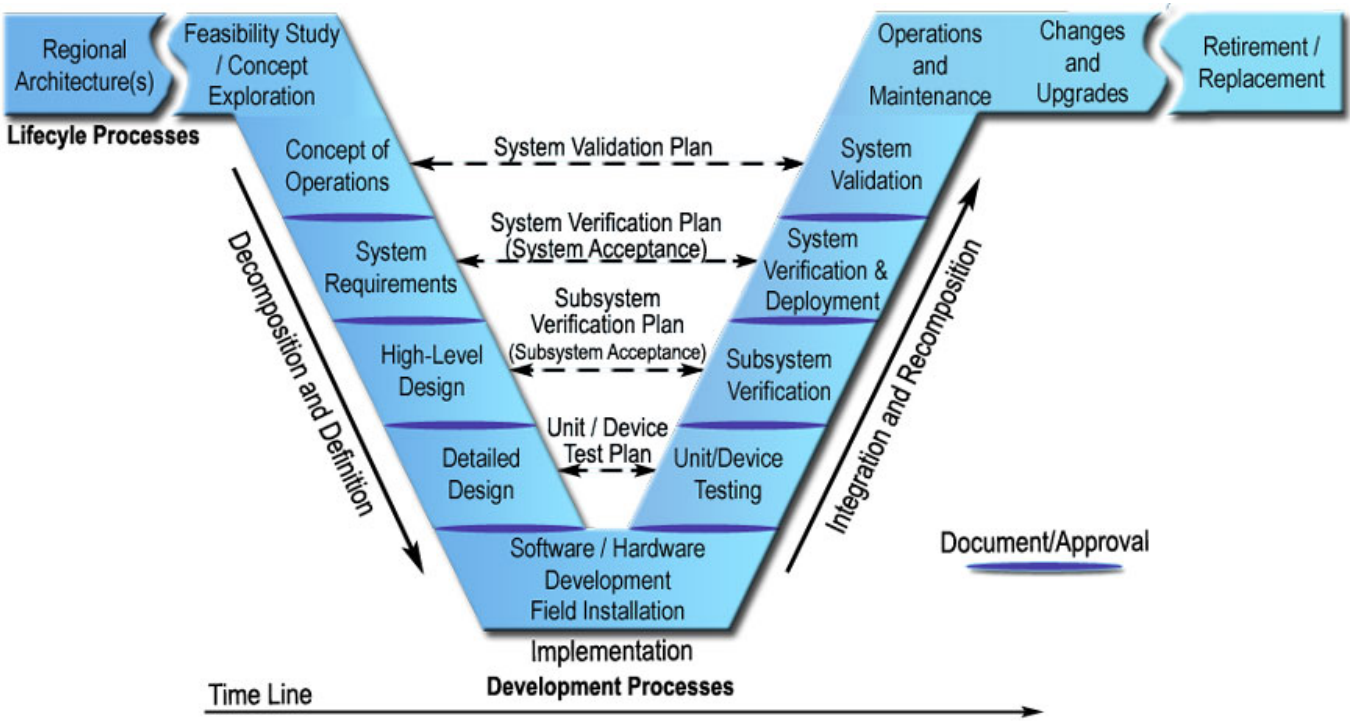
Trois phases principales : **spécification**, **conception** et **implémentation**.

Résumé et critique du développement en cascade

Spécification	Modèle conceptuel décrivant les besoins et fonctionalités exigées par le client (utilisateur)
Conception	Modèle physique décrivant l'approche algorithmique abordée
Implémentation	Concrétisation finale pour mise en production

Approche qui fonctionne en plan fixe pour des tâches **indépendantes**, **ne se chevauchant pas** et **connues à l'avance**.
Parfait pour construire une maison mais...

Développement en V (NASA, Forsberg et al., 1991)



Relation entre les phases gauche et droite par **tests de vérification et de validation**.

Tests et vérification du système

Par ordre croissant de concrétisation, le développement en V intègre trois catégories de test :

Qualification La fonction est-elle conforme aux exigences des spécifications des besoins du client ?

Intégration La fonction execute-t-elle son but demandé, sa fonctionnalité ?

Unitaire La fonction implémentée risque-t-elle de déclencher une erreur technique ? (crash, exceptions. . .)

Comparaison entre développement en cascade et en V

Critère	**En cascade**	**En V**
Spécification des besoins	Début	Début
Coût	Faible	Elevé
Garantie de succès	Faible	Elevé
Simplicité	Simple	Intermédiaire
Flexibilité	Rigide	Plutôt
Réutilisabilité	Limité	Moyen
Implication du client	Début uniquement	Début

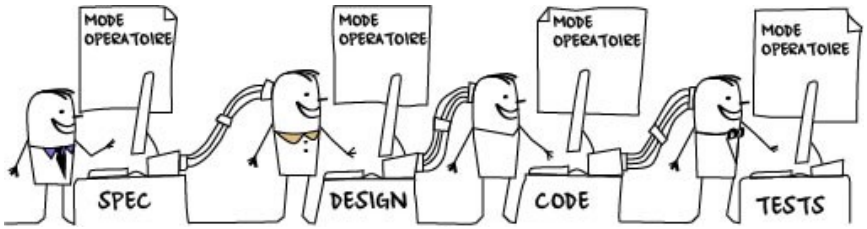
Développement Agile (Agile Manifesto, 2001)

Valeurs fondamentales

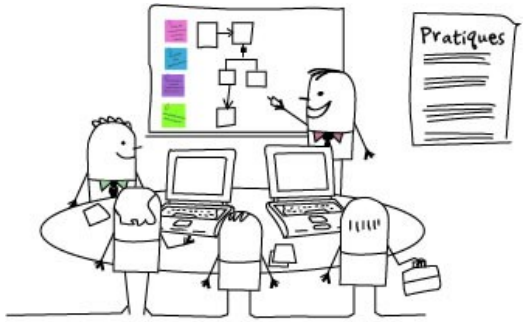
1. **Intéraction entre les individus**, processus, outils
2. **Logiciel fonctionnel** comme documentation compréhensible
3. Collaboration et **implication du client**
4. Réactivité rapide au **changement de plan**

- Répond aux limitations de l'approche (trop) **prédictive** du développement en V ;
- Adapté à la réalisation auto-gérée d'**objectifs précis** avec accent sur l'**innovation** et où la **prédictibilité des tâches est difficile**.

Développement Agile (Agile Manifesto, 2001)



Avant



Après

Limitations.

1. Adapté au développement et **non à la gestion de projets** ;
2. Nécessité de **combiner** avec d'autres méthodes pour fiabilité ;
3. Empêche la **maturation de concepts à long terme** ;
4. Indifférentiation des **responsabilités**.

Parties prenantes d'un projet

1. **Maitrîse d'ouvrage**
   - Responsable de l'expression du besoin ;
   - Généralement étranger à l'informatique.-
2. **Maîtrise d'oeuvre**
   - Responsable de la concrétisation logicielle ;
   - Généralement étranger aux besoins fonctionnels ;
   - Assure les bons choix techniques, en adéquation avec les besoins.

Etude de faisabilité

1. Faisabilité **économique**
2. Faisabilité **technique**
   - Compétences techniques des équipes de développement
   - Disponibilité des ressources
   - Maturité technologique suffisante
3. Faisabilité **légale**

Prototypage évolutif

Maquette

- Aspect extérieur et ergonomique du système
- Instaure un dialogue développeur-utilisateur
- Souvent jetable

Prototype

- Esquisser les fonctionnalités du produit
- Valider la réalisabilité des besoins
- Vérifier la cohérence des spécifications

Structure du cahier des charges (phase 1)

1. **Présentation du projet** (contexte, motivation, objectifs, état-de-l'art existant, maquette)
2. **Expression des besoins**
   1. **Fonctionnels** : Opérations et fonctions réalisées par le logiciel ;
   2. **Non fonctionnels** : Performance, système d'exploitation cible
3. **Contraintes** (budget, matériel, délais)
4. **Déroulement** (planification, méthodologie de qualité, responsabilités)

Formalismes liées à la spécification

Quelques exemples de moyens pour aborder la spécification :

Informel Langage naturel incluant **imprécisions et ambiguïtés**. Mélange des concepts et des interprétations ;

Semi-formel Langages graphiques de type *UML* ou *MERISE*, avec peu d'expressivité mais **faciles à utiliser** et à comprendre ;

Formel Langages logiques de type *Méthode B*, avec **fort prouvoir d'expressivité** et permettant de prouver la **correction de comportements**.
Exemple : Ligne de métro 14 de la RATP (projet METEOR)

Unified Modeling Language (UML)

Inspiré et développé par Booch et al., 1994 chez Rational Software dans le but de **standardiser** les systèmes notationels variés par approche *objet*. Désormais édité par l'**Object Management Group**.

Expression des besoins en UML

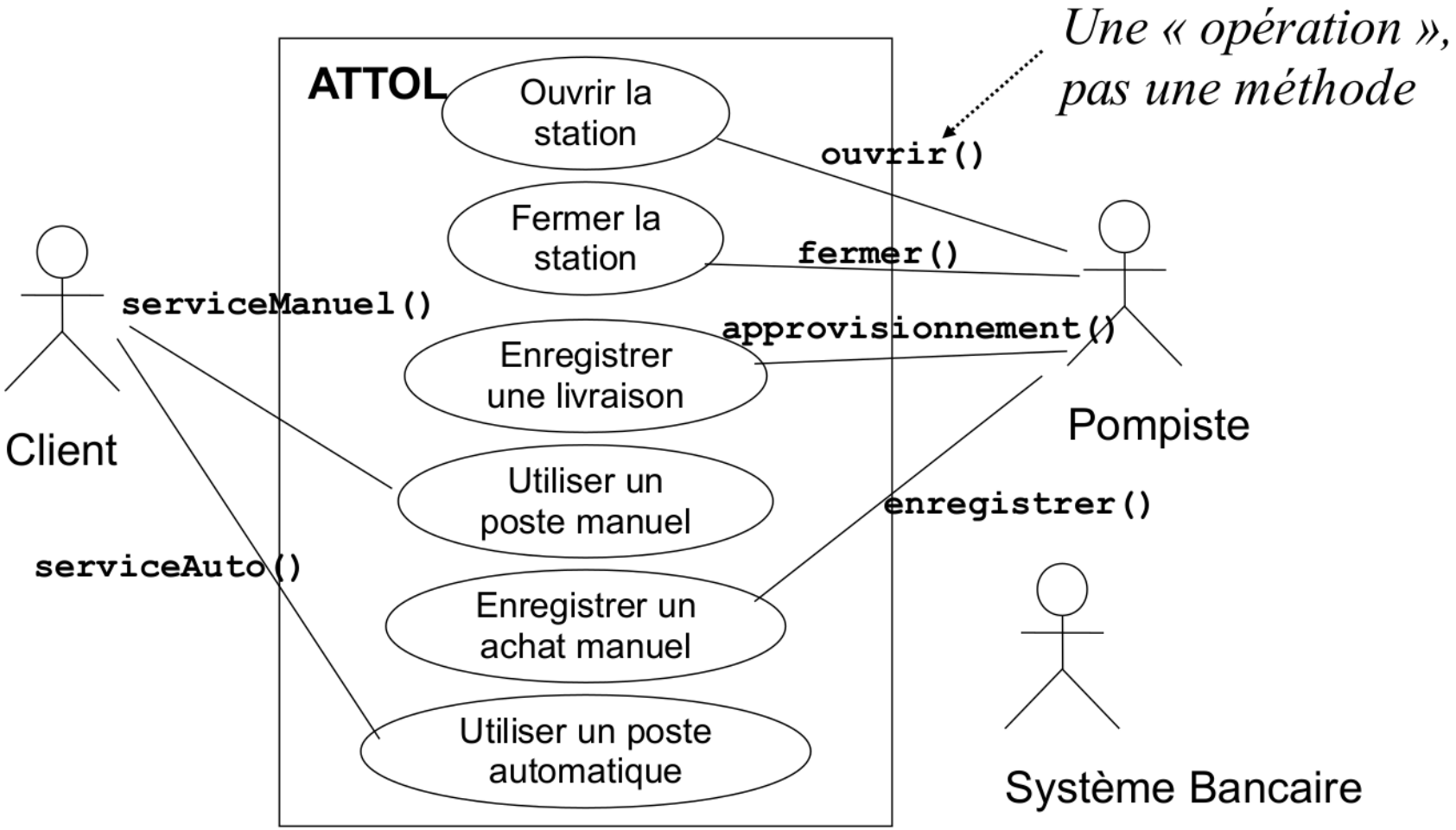
- Diagrammes de cas d'utilisation
- Diagrammes de séquences
- Diagrammes de classes

Diagrammes de cas d'utilisation

Définition. Fonctionalité du système, déclenchée par un *acteur* extérieur.

- Comportement du système
- Réactions et intéractions avec l'extérieur

Cas d'utilisation d'une station-service

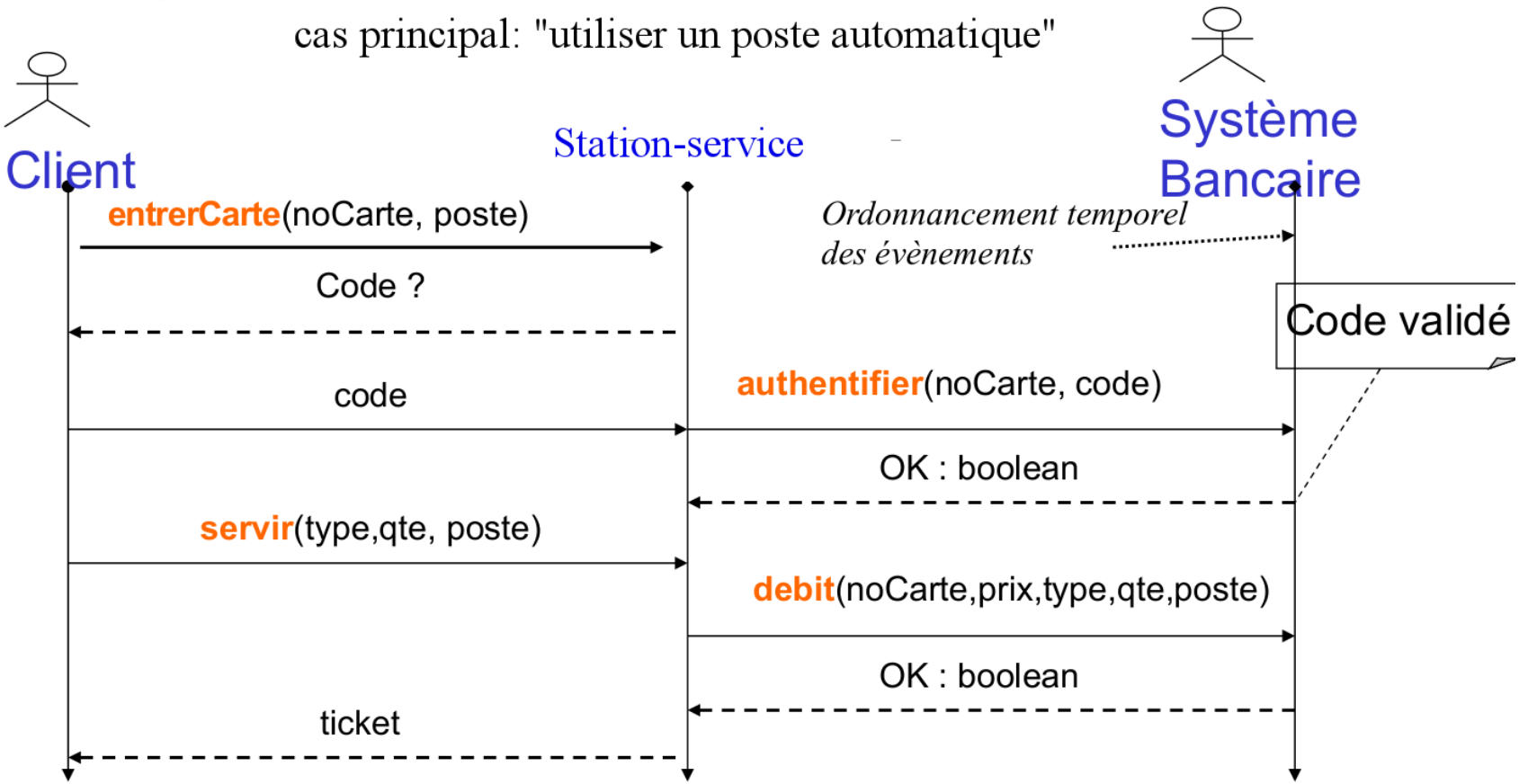


Diagrammes de séquences

En analyse,

- Pas d'échanges entre acteurs (impertinent)
- Pas de composants internes du système (conception...)
- Pas de boucles, ni tout autre *cadre*

Exemple de diagramme de séquence de la station-service

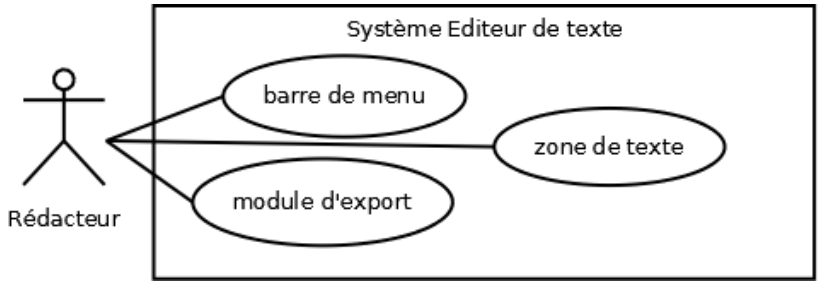


Exemple de synthèse d'un cas d'utilisation

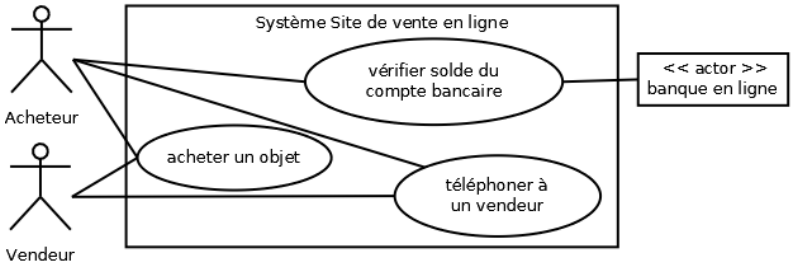
Utiliser un poste automatique	
Cas principal	Un client utilise un poste automatique après avoir authentifié sa carte bancaire et sélectionné un carburant. Le niveau de ce carburant reste au-dessus du niveau d'alerte .
Cas alternatif	Après utilisation d'un poste automatique par un client, le seuil d'alarme du carburant est franchi.
	Après utilisation d'un poste automatique par un client, le seuil minimal du carburant est franchi.
Cas d'exceptions	Echec d'authentification de la carte bancaire d'un client d'un poste automatique
Remarques	L'interface du système garantit qu'en cas de niveau de carburant inférieur au minimum, la sélection du carburant est impossible sur tous les postes. Ce niveau est tel que toute distribution commencée peut être achevée correctement

Erreurs courantes de modélisation

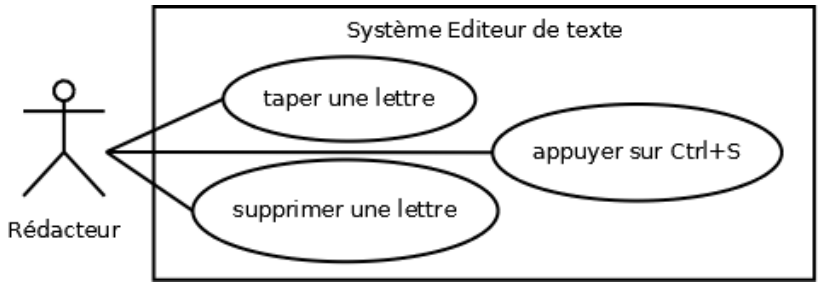
- Présence de détails d'interface du logiciel



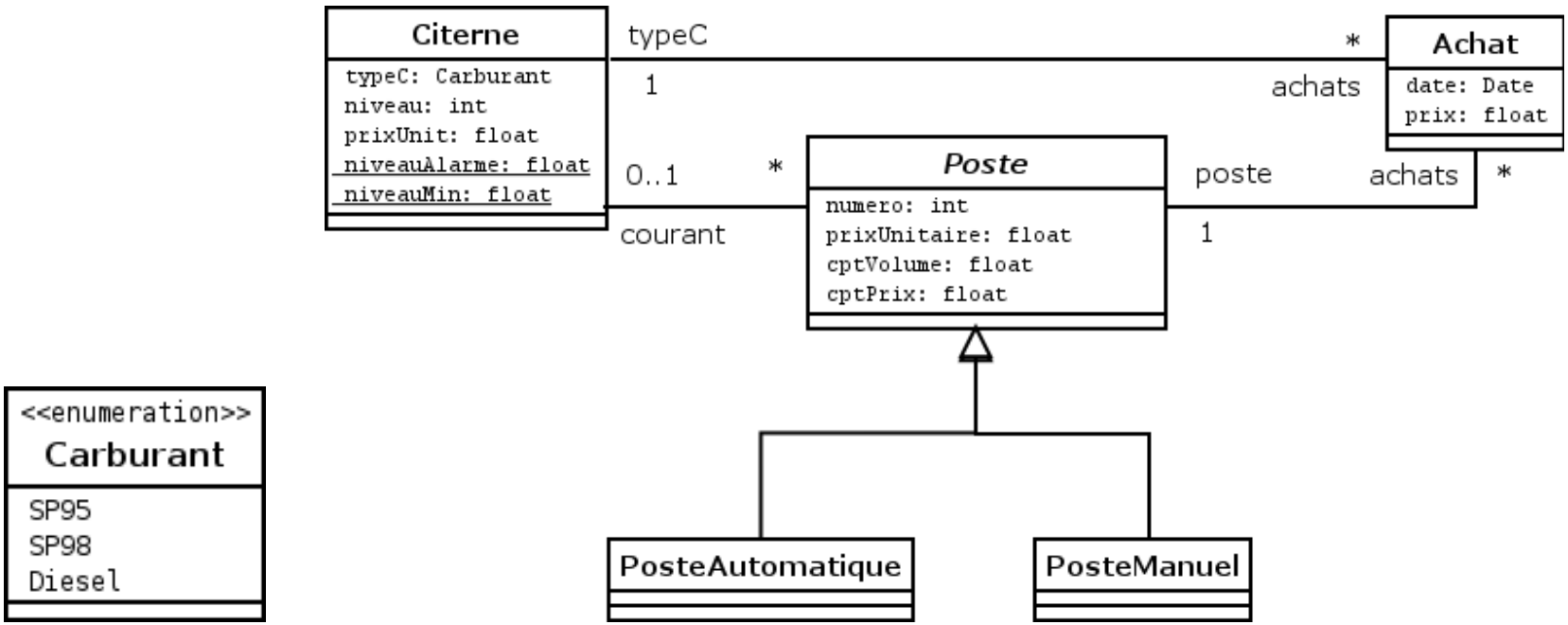
- Intéraction entre acteurs



- Non-sens vis-à-vis de l'utilisateur



Exemple de diagramme de classes de la station-service



Indispensable pour la conduite d'un projet

1. Déterminer les tâches
2. Identifier les compétences des maîtres d'oeuvre
3. Décider des ressources allouées à chaque tâche
4. Décider d'un planning de livrables

Estimer la durée d'un projet et ses incertitudes

Program Evaluation and Review Technique (PERT)

1. Identifier les tâches et estimer leur durée
2. Ordonner par dépendance
3. Exploiter

Exemple de préparation d'un bon couscous

Référence	Tâche	Durée
A	Couper les légumes	30 min
B	Couper la viande	20 min
C	Cuire le tout	1h

Quel **ordre** donner aux tâches ? Lesquelles peuvent-elles être faites en **parallèle** ?

Cinq conseils pour rater le projet

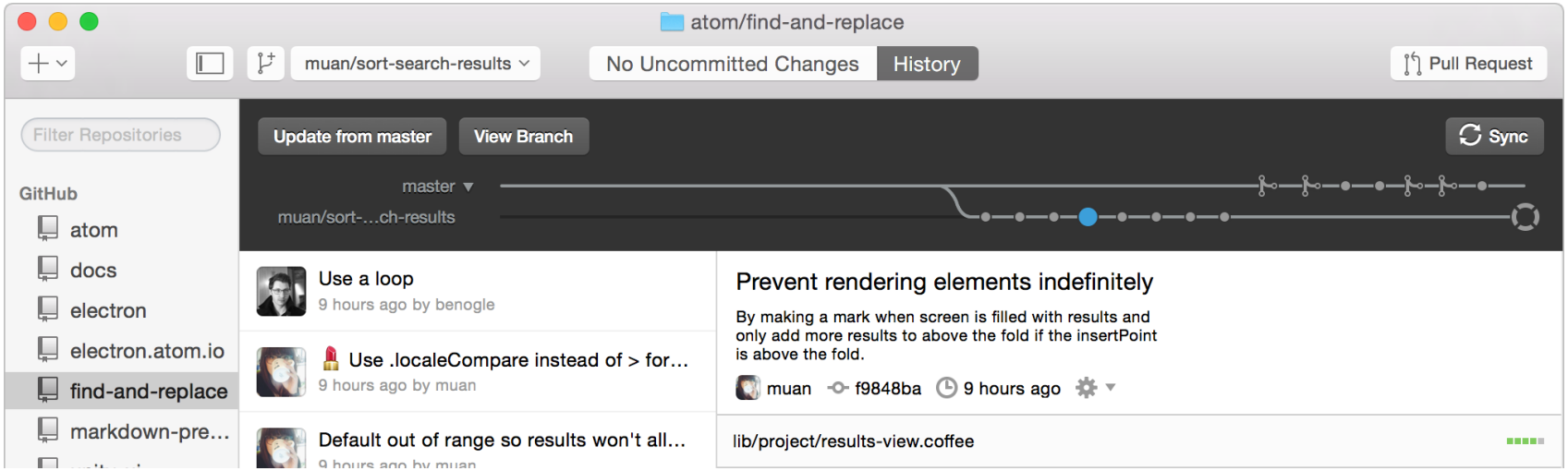
1. Ne pas se faire **comprendre**
2. Sous-estimer les **deadlines**
3. Rester **trop dans le détail ou dans le large**
4. Utiliser des **mauvais outils**
5. Être **passif** dans une situation

Gestionnaire de versionnement Git

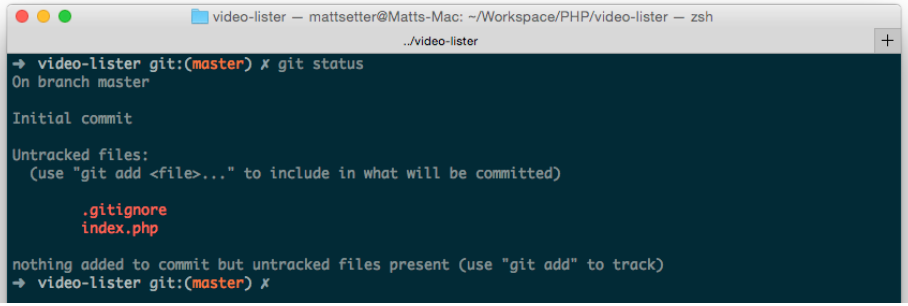
- Visibilité professionnelle sur GitHub
- Eviter les échanges de code type-email
- Collaborer entre plusieurs développeurs
- Revenir à une version précédente dans le temps

Gestionnaire de versionnement Git

Avec un outil graphique



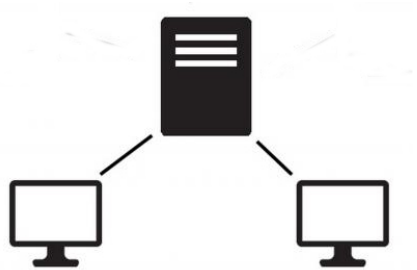
En ligne de commande



Gestionnaire de versionnement Git

Quelques opérations de base

Init	Initialisation d'un dépôt local
Clone	Récupération d'un dépôt distant
Add	Sélectionner les fichiers modifiés
Commit	Déclarer un changement
Push	Pousser les changements locaux vers le serveur distant
Pull	Récupérer les changements depuis le serveur distant



References

P. R. Reed, *Developing Applications with Java and UML*, Addison-Wesley