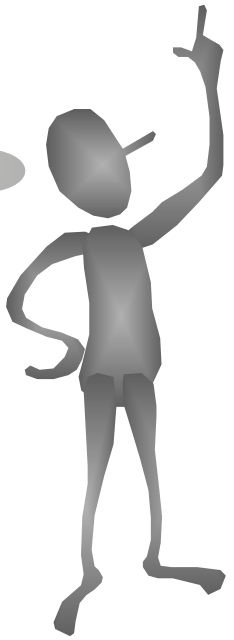
Génie logiciel



M. HAMRI

amine.hamri@univ-amu.fr

Objectifs du cours

- Définir le génie logiciel
- Montrer l'utilité du génie logiciel
- Recenser des méthodes et des outils utilisés en génie logiciel
- Acquérir la maîtrise de quelques techniques de génie logiciel

Bibliographie

- Le génie logiciel et ses applications, Ian Sommerville, 1988, ISBN 2-7296-0180-5
- Spécification et conception des systèmes. Une méthodologie, Jean-Paul Calvez, Masson, Paris, 1991, ISBN 2-225-82107-0
- Génie logiciel. Les méthodes SADT, SA, E-A, SA, Patrick Jaulent, 1990, ISBN 2-200-42002-1
- Génie Logiciel : principes, méthodes et techniques, Presses polytechniques et Universitaires Romandes, 1996, ISBN 2-88074-296-X
- Précis de génie logiciel, Marie-Claude Gaudel, Bruno Marre, Françoise Schlienger et Gilles Bernot, Masson, Paris, 1996, ISBN 2-225-85189-1
- Génie logiciel, Jacques Printz, Que sais-je ?, 1995, ISBN 2-13-046894-2
- Le génie logiciel orienté objet, Ivar Jacobson, Addison-Wesley, 1993, ISBN 2-87908-042-8
- Conception et programmation orientées objets, Bertrand Meyer, Eyrolles 2000, ISBN 2-212-09111-7
- Bases de la programmation et du génie logiciel, Guy Pierra, Dunod, 1991, 2-04-020722-8
- Méthodologie de la programmation en C - Bibliothèque standard - API POSIX, Jean-Pierre Braquelaire, Masson, 1998, ISBN 2-225-83269-2
- Eléments de génie logiciel. Outils de développement en C, Jacques Chauché, 1990, 2-7298-9058-0
- Qualité du logiciel et système qualité - L'industrialisation par la certification, Jean-Pierre Martin, 1992, ISBN 2-225-82801-6
- L'assurance qualité - La nouvelle version de la norme ISO 9001 en pratique, Claude Jambart, Economica, Paris, 2001, ISBN 2-7178-4199-7
- Maîtriser la qualité - Tout sur la certification et la qualité totale - les nouvelles normes ISO 9001 version décembre 2000, Daniel Boéri et Mélina Cuguen, Maxima, Paris, 2001, ISBN 2-84001-271-5

Plan du cours

- Les problèmes du développement logiciel
- Le cycle de vie
- Méthodes et modèles
- La spécification
- La conception
- La programmation
- Le test

Le génie logiciel

- Expression née en 1968 (conférence de l'OTAN)
- Définition : Méthodes, techniques et outils de conception et réalisation des logiciels de qualité industrielle
- Objectifs : Maîtrise des coûts, des délais, de la qualité
- Moyens : Etude des méthodes et techniques suivies pour le développement et la maintenance des logiciels ➔ capitalisation du savoir-faire, modularité / réutilisabilité, automatisation partielle, standardisation et normalisation, mesures, modèles, assurance qualité
- Difficultés : Secret industriel qui nuit à la diffusion du savoir-faire (y compris en présence de problèmes), spécificités liées aux domaines d'application, impression de facilité, etc.

Deux aspects complémentaires du génie logiciel

- Transformation d'un problème informel en une solution logicielle

- Maintenance du logiciel jusqu'à la fin de son cycle de vie (mise à jour, correction des erreurs résiduelles, évolutions, etc.)

➔ **au-delà de la simple programmation**

Un programmeur écrit un programme alors qu'un ingénieur GL écrit (conçoit, programme, documente, valide) un composant appelé à être inséré au sein d'un système

Produit logiciel = programmes + documents

Types de logiciel

- Logiciel système
- Logiciel temps réel
- Logiciel de gestion
- Logiciel scientifique
- Logiciel combinatoire
- Logiciel critique

- Logiciel / Progiciel

Les problèmes du développement logiciel

Données économiques sur le coût des logiciels

- Volume, effort, délai, durée de vie
- Evolution de la demande
- Evolution des coûts
- Nature du risque logiciel

Volume, effort, délai, durée de vie

- Quelques coûts
  - Compilateur C — 10 HA
  - Compilateur ADA — 120-150 HA
  - AGL (Entreprise, Esat, etc.) — 150-200 HA
  - SGBD relationnel (Oracle, etc.) — 300-500 HA
  - Système industriel de GPAO — 400 HA
  - Système temps réel de la navette spatiale — >1000 HA
    2200 KLS
    6 ans
  - Système d'exploitation (MVS, VMS, etc.) — 2500-5000 HA
    5000-15000 KLS
    15-20 ans

L'évolution de la demande

- Le logiciel est utilisé presque partout :
  - montre 2 K instructions
  - TV 100 K instructions
  - automobile, téléphone mobile 150 K instructions
  - central téléphonique 1 M instructions
  - système de combat du porte-avions Charles de Gaulle 8 M instructions
- Tout ce qui était analogique devient digital et est candidat à être traité par ordinateur : GED (Gestion électronique de documents) et multimédia, images de synthèse, etc.
- Les services demandés à l'informatique sont en augmentation constante

L'évolution de la demande

- L'explosion du logiciel continue :
  - En 1995, le développement de Microsoft Exchange Server a coûté

    1000 HA pour 7 000 KLS,

    soit une productivité de 30 lignes par homme et par jour.

  - De 1965 à 1995, le volume de chaque logiciel a été multiplié par 100, alors que la productivité n'augmentait que d'un facteur 3.

- Le progrès exponentiel de l'informatique devrait se poursuivre encore longtemps

L'évolution des coûts

- Inversion complète des coûts matériel / logiciel
- La part des coûts logiciels de maintenance est de plus en plus grande
- Le coût humain devient le facteur de coût principal

La nature du risque logiciel

- La sûreté de fonctionnement devient essentielle
  - Risques humains
    - commandes de vol des avions
    - contrôle aérien, ferroviaire, nucléaire
    - etc.
  - Risques économiques
    - crach boursier d'octobre 87
    - effondrement du réseau de télécommunication AT&T sur le côté est des Etats-Unis d'Amérique
  - Risques sociaux
    - le système Socrate de la SNCF
    - la confidentialité des informations
    - les virus et les "chevaux de Troie"

Le constat

- Présence de nombreuses erreurs
  - Cas du système de la navette spatiale
    - logiciel embarqué 0.11 par KLS (1 toutes les 10000 lignes)
    - logiciel au sol 0.40 par KLS (1 toutes les 2500 lignes)
  - Logiciel grand public $\approx$ 5-10 par KLS
- Evolutivité problématique
  - Cas des grands systèmes

Des exemples de bogues

- 2 jours sans courant pour la station Mir, du 14 au 16 novembre 1997.
  Cause : plantage d'un ordinateur qui contrôlait l'orientation des panneaux solaires.
- Mission Vénus : passage à 5 000 000 km de la planète au lieu des 5 000 km prévus.
  Cause : remplacement d'une virgule par un point.
- Perte de satellites dans les années 70.
  Cause : +I au lieu de +1 dans une instruction du programme source FORTRAN (le langage admet sans le signaler des identificateurs non déclarés, leur valeur est alors aléatoire).
- Refus illégal de prestations sociales à des ayants droits.
  Cause : le cahier des charges n'avait pas prévu tous les cas.
- Inondation de la vallée du Colorado en 1983.
  Cause : mauvaise modélisation du temps d'ouverture du barrage.
- Echec du 1er lancement de la fusée Ariane V qui a explosé en vol.
  Cause : reprise du logiciel de plate-forme inertielle d'Ariane IV sans nouvelle validation. Le logiciel a considéré que l'inclinaison d'Ariane V (+ importante que celle d'Ariane IV car ses moteurs étaient plus puissants) n'était pas conforme au plan de tir et a provoqué l'ordre d'auto-destruction. Coût du programme d'étude d'Ariane V : 38 milliards de francs.

Et le bogue de l'an 2000

La lutte contre le bogue de l'an 2000 a coûté, seulement pour la France, des milliards de dollars. Des dysfonctionnements ont quand même été constatés, par exemple :

- Défaillances des systèmes informatiques de détection des vents de surface, dans plusieurs aéroports américains.
- Données fournies par des satellites de renseignement du Pentagone perdues à jamais.
- Bébé danois enregistré comme ayant 100 ans.
- 48 incidents répertoriés en France par le ministère de l'économie.

Cause : Codage de la donnée "année" sur 2 caractères pour gagner de la place.

La mauvaise maîtrise du développement de logiciel

- De nombreux projets informatiques n'ont jamais abouti :
  - système de réservation de places de United Airlines
  - système d'exploitation Rhapsody
- Ou n'ont pas réalisé les fonctionnalité attendues :
  - Windows 3
  - Windows 2000 / ME
- Ou ont été des catastrophes économiques :
  - système de réservation de places Socrate de la SNCF
  - système d'information de la Bibliothèque Nationale de France : dépassement de 40% du budget initial (280MF)
  - informatisation des caisses de sécurité sociale, carte Vitale

Les causes

- Erreurs humaines
  - activité logico-mathématique
  - généralisation / construction des abstractions
  - modélisation des systèmes
  - communication (comprendre et être compris)
  - traduction
- Complexité du problème
- Multiples versions
- Durée de vie sur plusieurs années
- Instabilité des spécifications

La qualité logicielle

- Un système logiciel peut être observé selon ses qualités :
  - externes

    détectables par les utilisateurs du produit et les clients

    ex : validité, robustesse, performance, à coût optimal, extensibilité, réutilisabilité, compatibilité, efficacité, portabilité, vérifiabilité, intégrité, facilité d'utilisation, facilité d'apprentissage, ergonomie, …
  - internes

    perceptibles par des informaticiens (concepteurs, implémenteurs)

    déterminants pour l'obtention des qualités externes

    ex : modularité, lisibilité, maintenabilité, ...
- Les facteurs qualité ne sont pas nécessairement compatibles 2 à 2
  ➔ compromis à trouver

Le problème du génie logiciel

- L'objectif est de construire des logiciels
  - ergonomiques
  - fiables
  - évolutifs
  - économiques

  ➔ satisfaisant les critères CQFD
  (Coût / Qualité / Fonctionnalités / Délais de réalisation)

- Le problème central est la présence d'erreurs et leur détection

Les solutions

Objectifs du développement logiciel

- Satisfaire les besoins du demandeur (client)
- Respecter les délais et les coûts
- Satisfaire des critères de qualité

Maîtrise des constituants de la réalisation
Interprétation correcte du domaine du client
Développement en équipe de plusieurs personnes
(cas d'applications complexes)

à défaut d'assurer la qualité d'un logiciel, assurons celle des méthodes de développement et de maintenance

Utilisation d'une méthodologie de développement
Mise en place d'une démarche qualité

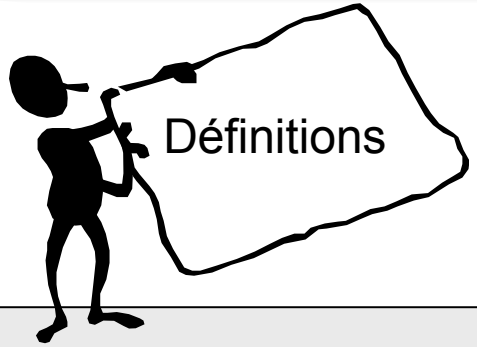
Conclusions

- **Il est impossible de créer un logiciel sans défaut.**
- La correction d'erreurs peut entraîner d'autres erreurs
- Le coût du développement d'un logiciel est très élevé. Il peut être estimé, très globalement, à :
  75 (développement) + 2250 (maintenance) € par instruction.
- L'effort de développement est proportionnel au nombre d'instructions
- La productivité obéit à une loi empirique, la loi des rendements décroissants :
  $MH = 3,5\ KISL^{1,2}$
  et n'augmente pas avec l'augmentation de l'investissement en hommes.
- Les moyens de sortir de cette crise du logiciel sont en partie connus
  ➔ méthodologie de développement + processus d'assurance qualité
- Les moyens mis en œuvre pour le développement d'un logiciel définissent son niveau de maturité. Plus ce niveau est élevé, moins il y a de bogues par ligne.

Modèle de développement :

le cycle de vie

Le développement d'un logiciel

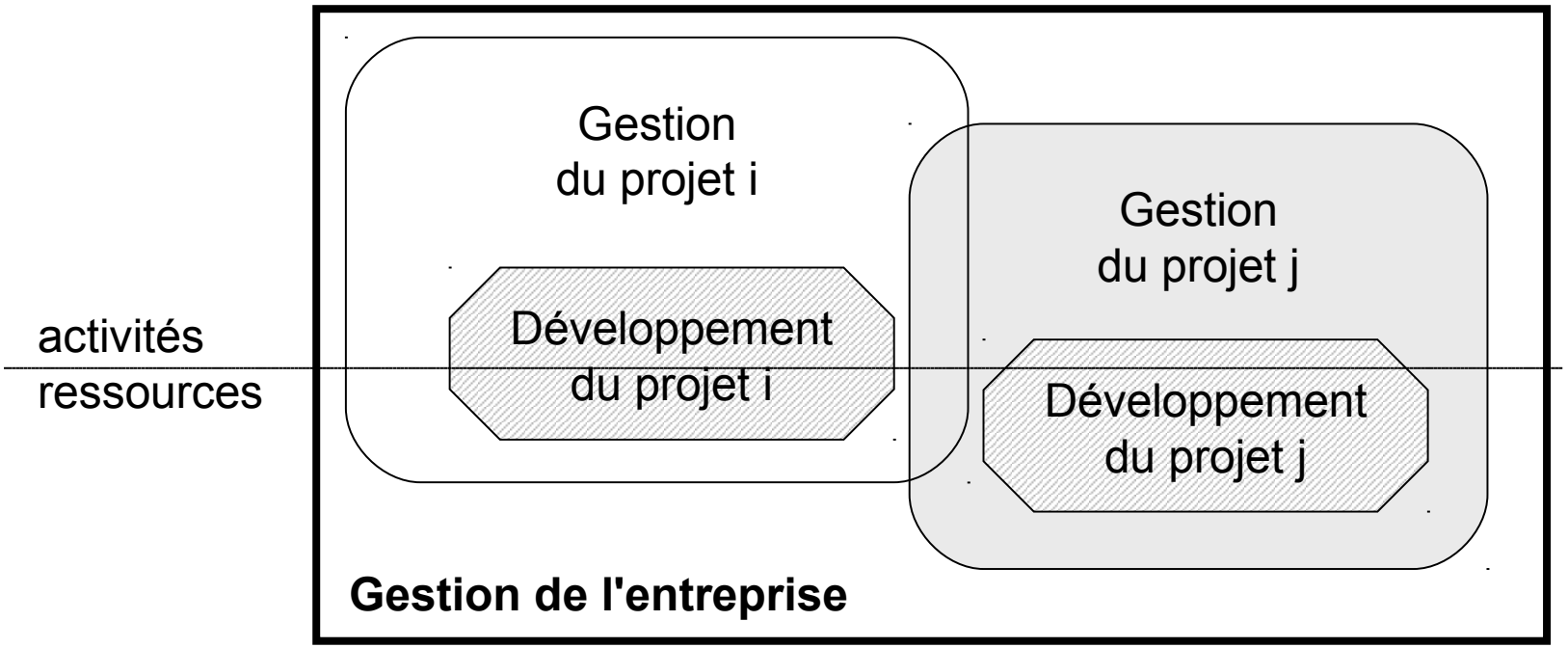


Le développement désigne l'ensemble des activités qui permettent de passer d'un cahier des charges au produit industriel répondant au besoin.

Le cahier des charges est un document décrivant les besoins du client.

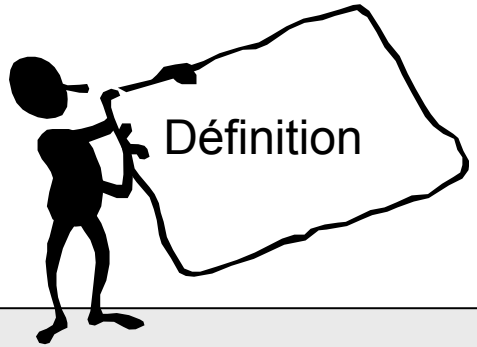
Le contexte d'un développement

L'activité de développement doit être replacée dans le contexte de l'entreprise :



La gestion de projet nécessite de modéliser le processus de développement lui-même ➔ **cycle de vie**

Le cycle de vie d'un logiciel



Le cycle de vie décompose le processus de développement selon une série d'activités couplées entre elles, partant du besoin initial pour aboutir au produit opérationnel.

Les modèles de cycle de vie

- Plusieurs modèles de cycle de vie ont été élaborés. Tous ont en commun au minimum les 5 phases essentielles de tout développement :
  - spécification ← cahier des charges
  - conception
  - réalisation
  - test
  - exploitation
- Ces modèles permettent de structurer le travail et facilitent le contrôle.

V&V

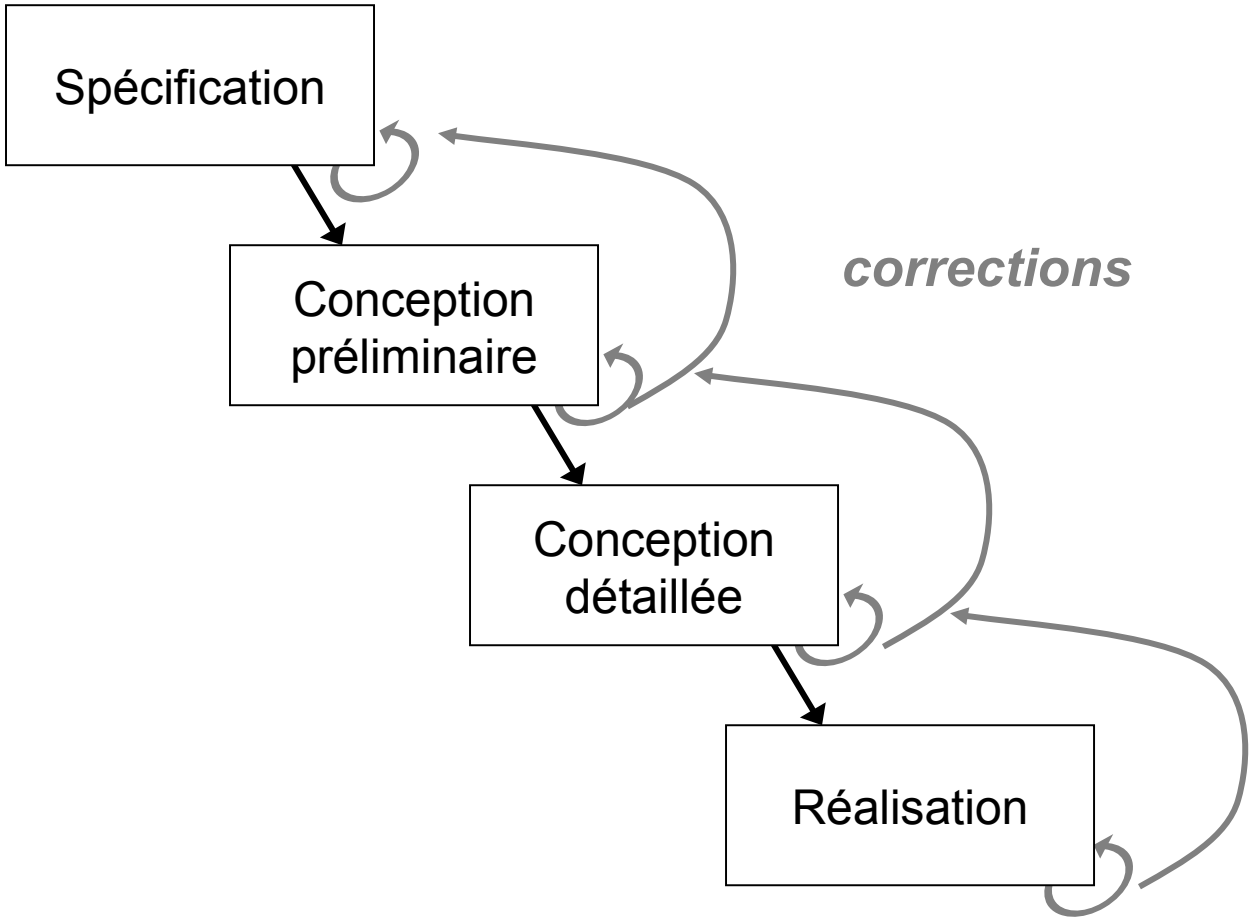
- **Validation** externe

  Conformité des spécifications aux besoins des utilisateurs

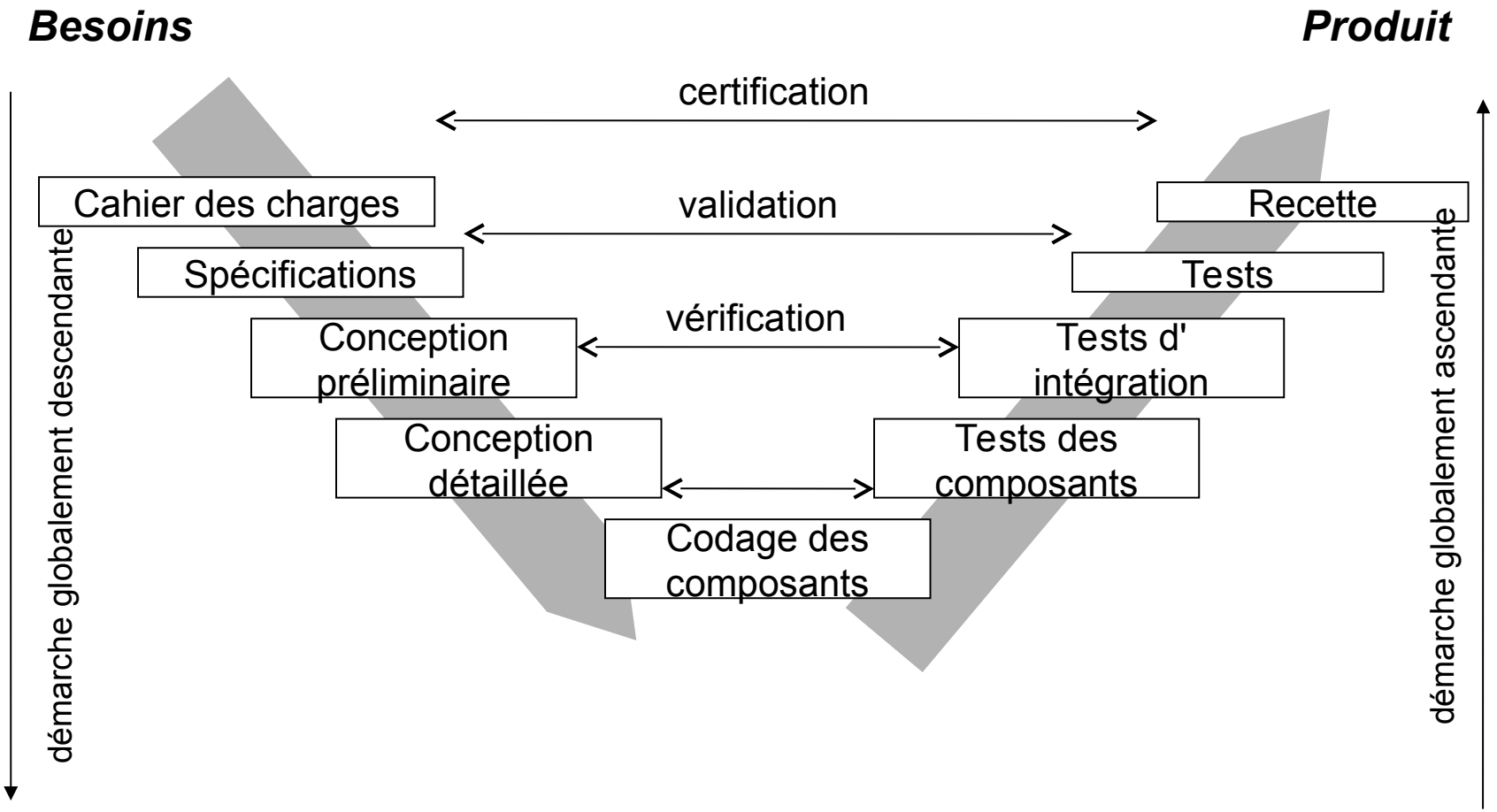
- **Vérification** (validation interne)

  Conformité d'une réalisation logicielle ou d'une spécification détaillée à une spécification plus abstraite

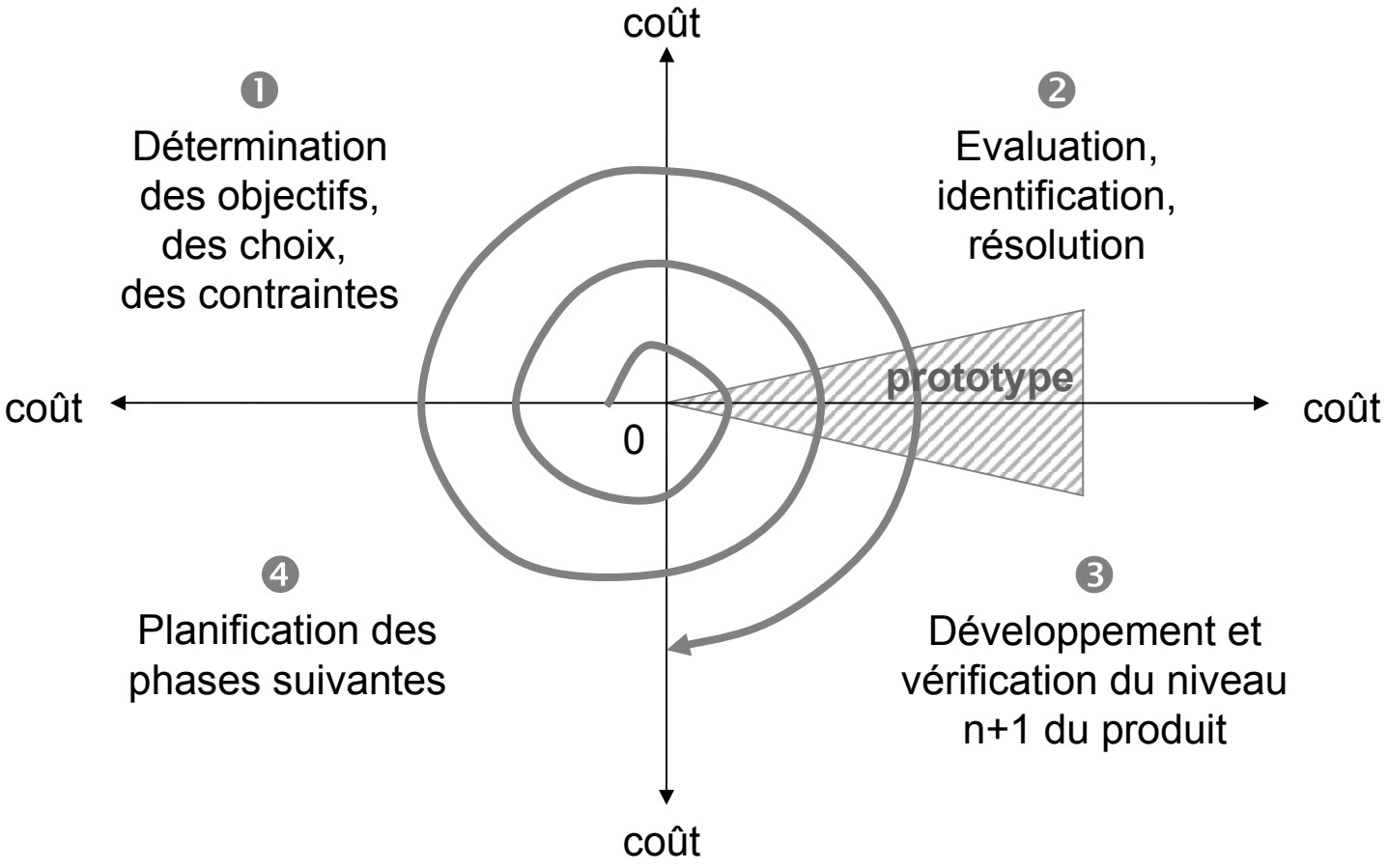
Le modèle de la cascade



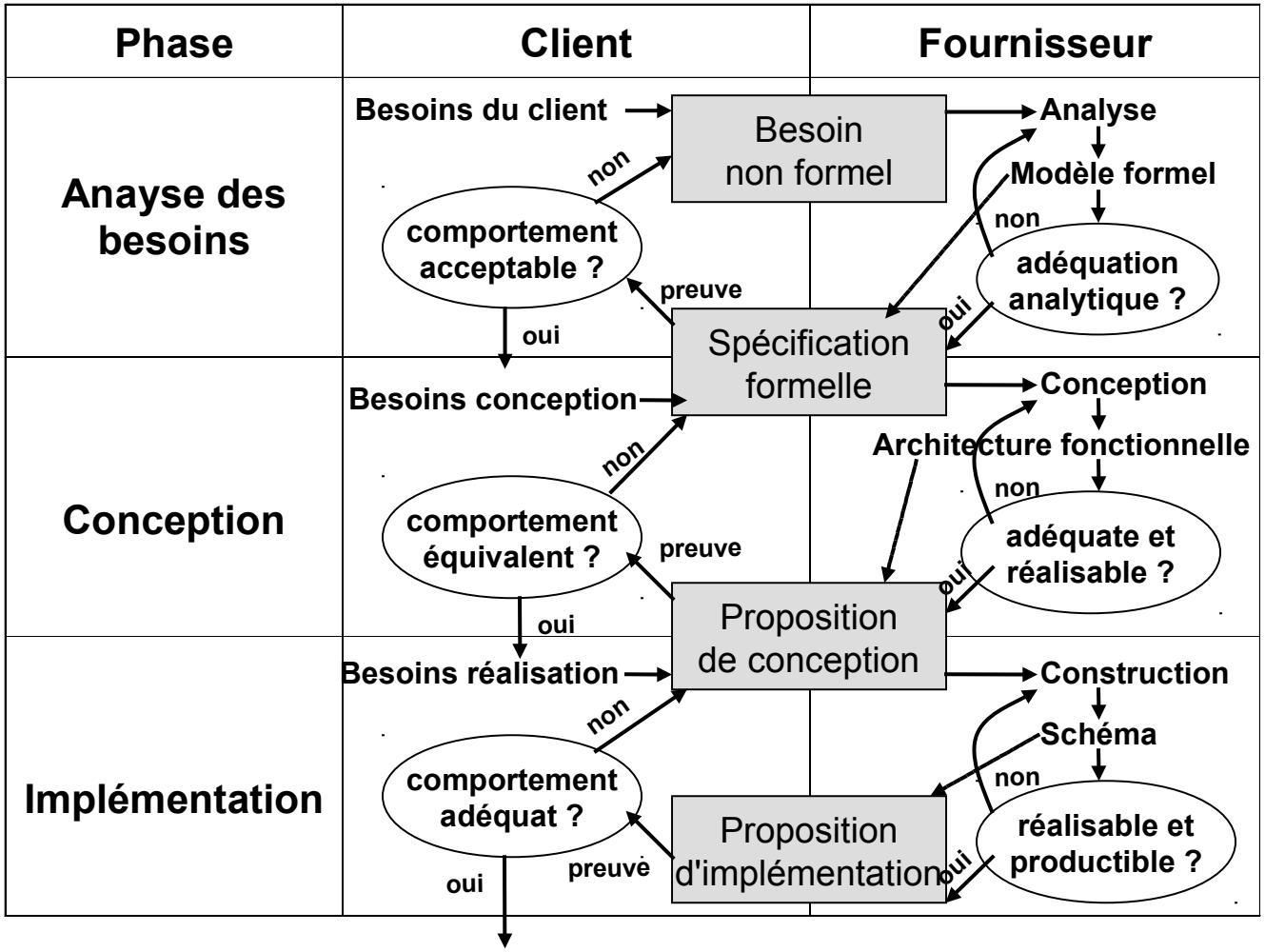
Le modèle en V



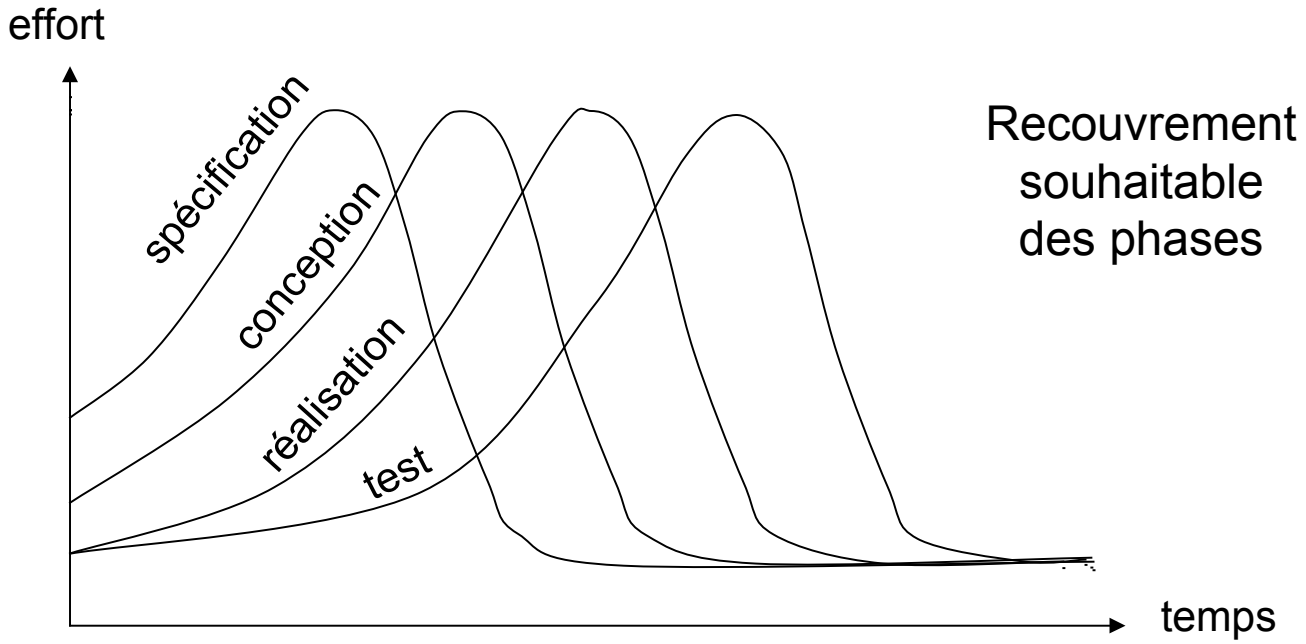
Le modèle en spirale



Le modèle contractuel

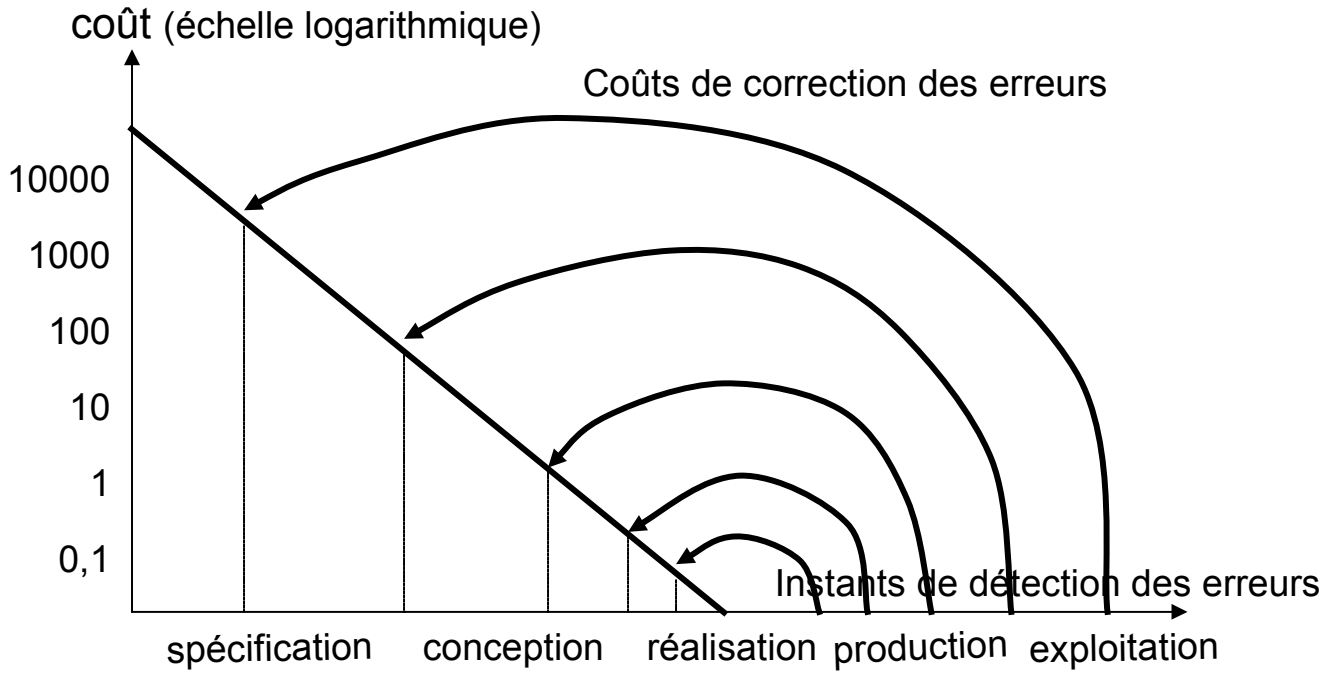


Recouvrement des phases



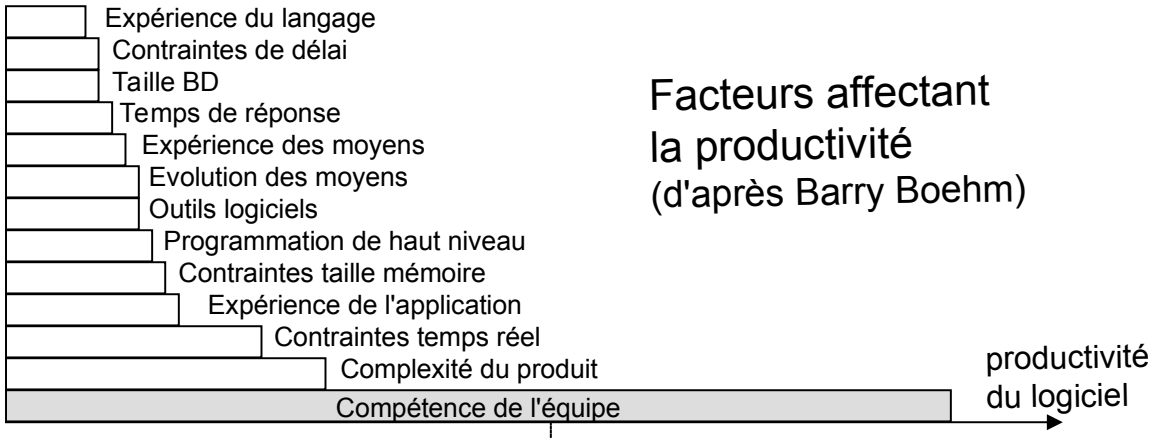
- Toutes les phases débutent presque simultanément, mais avec un effort qui varie en fonction de la progression.
- Les spécifications sont rarement achevées car des évolutions apparaissent côté demandeur

Coût de correction des erreurs



➔ Plus les erreurs sont détectées tard dans le cycle de vie, plus elles sont difficiles à cerner et à corriger : le coût de correction d'une erreur est à chaque étape multiplié par 10.

Facteurs de productivité



- Faible contribution des facteurs d'expérience
- Le facteur essentiel est la compétence de l'équipe qui est intimement liée aux méthodes utilisées

Répartition de l'effort entre les phases

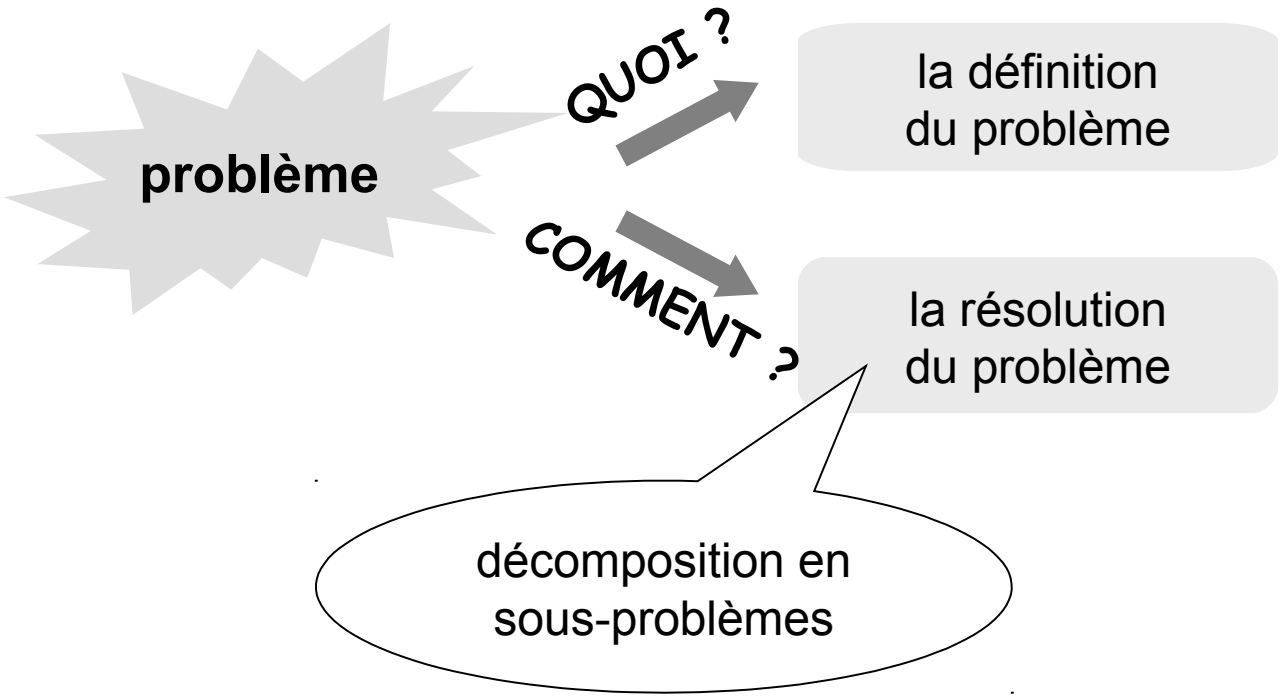
La répartition de l'effort dépend de la nature du problème et de la stratégie de développement souhaitée.

Pour réduire au minimum possible le coût des erreurs et pour que la réalisation soit efficace, l'effort doit être porté sur les deux premières phases : spécification, puis conception.

Les activités de spécification et de conception peuvent être jugées comme improductives. La tendance naturelle est de penser qu'on est plus efficace et plus productif quand on est affairé sur son programme que quand on élabore au préalable des documents de spécification et de conception.

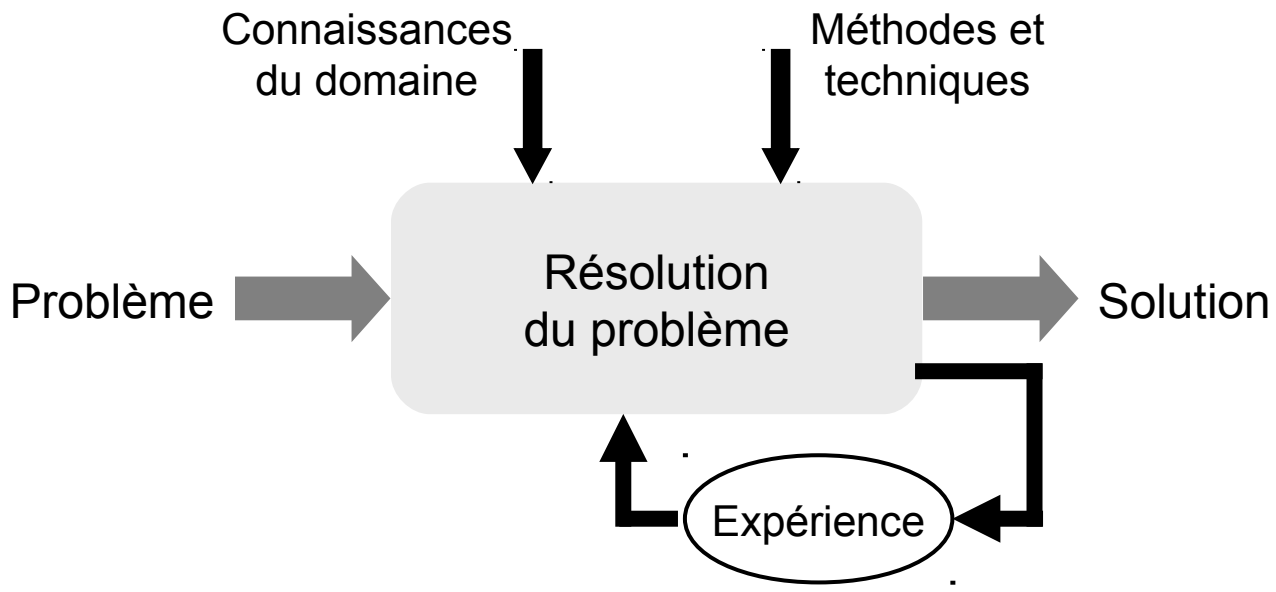

Méthodes et modèles

Le problème



Résoudre un problème consiste à analyser la situation dans laquelle il existe, déduire des décisions, puis décider d'une solution.

Le processus de résolution de problème

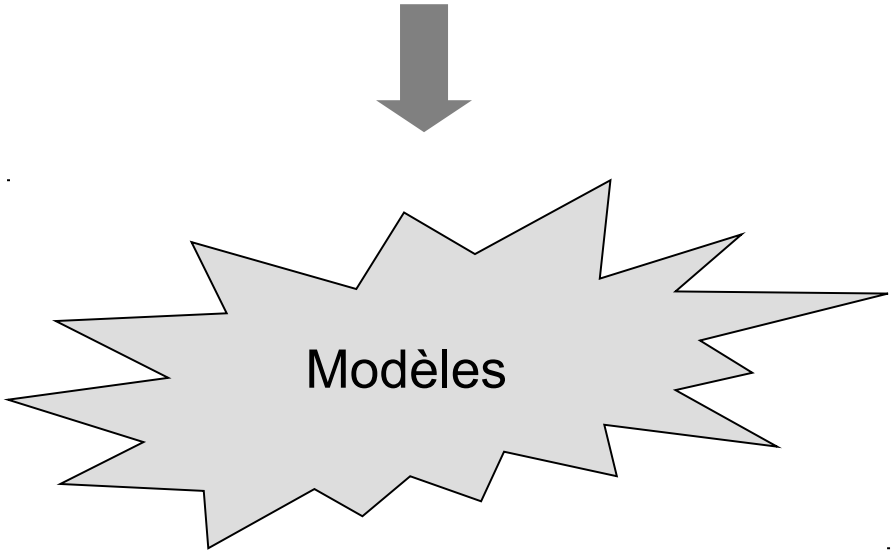


Le processus de résolution de problème

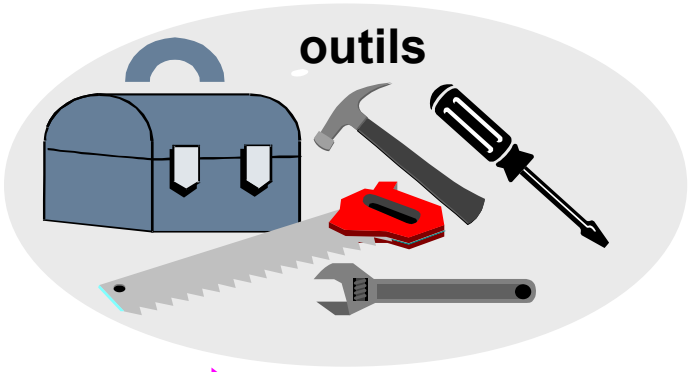
- C’est un processus de prise de décisions
- Démarche normale de résolution de problème :
  - formuler le problème
  - analyser le problème pour faire apparaître suffisamment de détails
  - rechercher les solutions potentielles
  - déduire la solution la plus appropriée en évaluant et en comparant des solutions entre elles
  - décrire en détail la solution retenue.

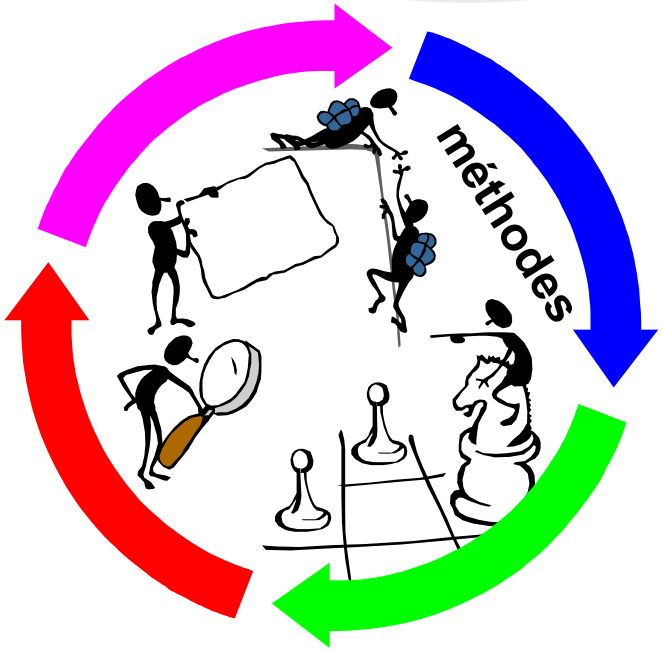
Le processus de résolution de problème

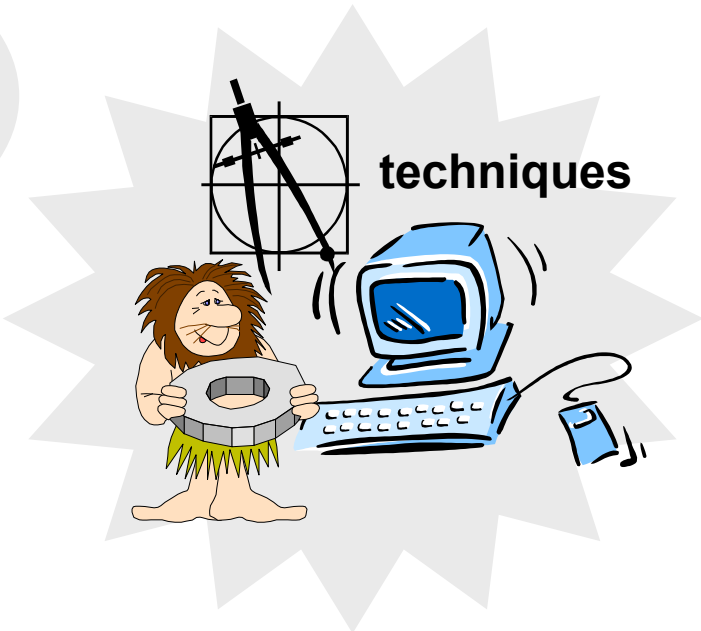
Les solutions sont-elles correctes et de qualité ?

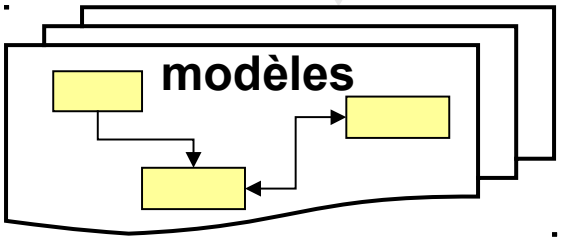


Qu'est-ce qu'une méthodologie de développement ?

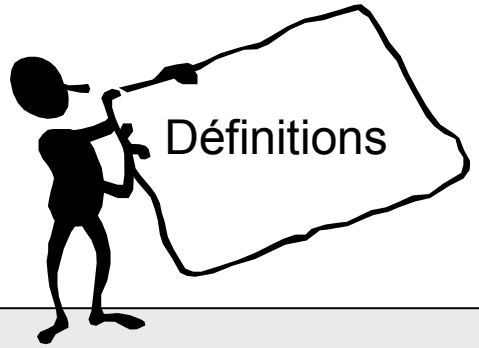








Méthode et méthodologie



Une méthode est un ensemble de règles qui conduisent à une solution.

Une méthodologie est une agrégation de méthodes, guides, outils, techniques de différents domaines, permettant de déduire la manière de résoudre un problème.

Utilité d'une méthodologie de développement

Une méthodologie ne couvre pas nécessairement tout le cycle de développement.

Objectifs :

- garantir l'obtention d'un résultat
- faciliter l'évaluation a priori de la faisabilité, du coût et du temps de développement
- augmenter la productivité des concepteurs et la qualité du résultat
- accroître la réflexion à un niveau conceptuel
- permettre l 'organisation et la conduite de projets

Les 3 vues d'un système

Tout système est observable selon 3 vues complémentaires :

- structurelle

  Description des composants (données, fonctions) et des relations entre ces composants

- comportementale

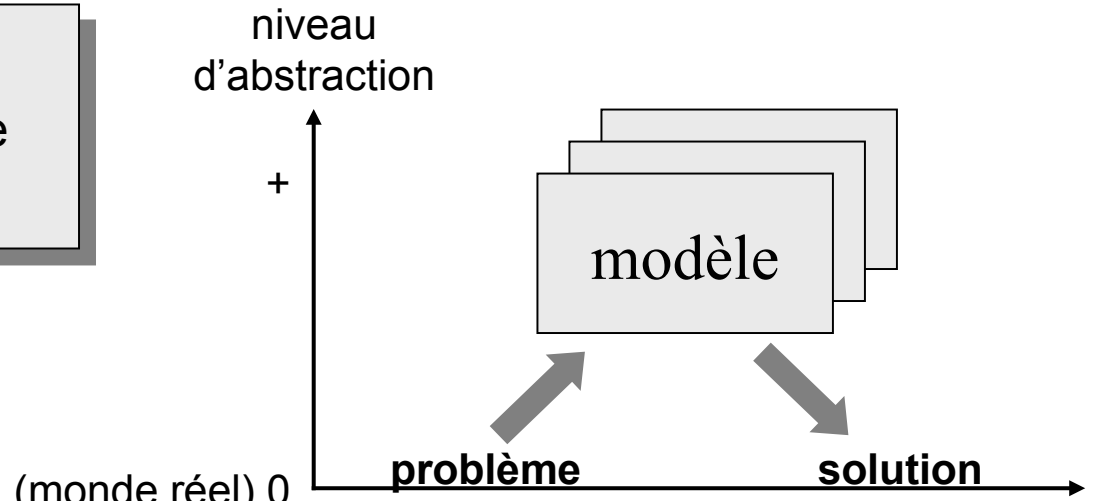
  Description de l'évolution des fonctions

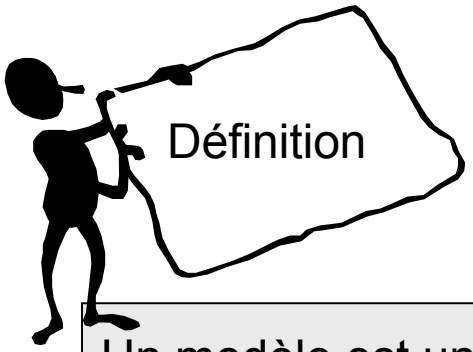
- exécutive

  Description de la partie matérielle du système

Modèles et modélisation

La modélisation est une activité qui précède toute décision ou formulation.





Un modèle est une vue partielle plus ou moins abstraite de l'existant.

Un modèle est une interprétation de la compréhension d'une situation ou d'une idée de la situation.

Modèle

- Un modèle représente une description d'un système réel, ou d'un système qui deviendra réalité après sa conception, et ceci à un certain niveau de détail.

  ➔ Les modèles diffèrent selon la description à exprimer (spécification ou conception)

- Un modèle est aussi utilisé pour décrire les caractéristiques du système à concevoir. Il sert alors de base pour la vérification de ses propriétés. Les caractéristiques spécifiées par un modèle dépendent de la nature de celui-ci

  ➔ Il faut sélectionner une classe de modèles qui induit la propriété à exprimer.

Qualités des modèles

Pour être exploitable, un modèle doit présenter les qualités suivantes :

- abstraction :
  pour exprimer le comportement de l'ensemble sans faire référence aux détails de toutes ses parties

- raffinement :
  un sous-ensemble du modèle doit pouvoir être décrit à l'aide d'un autre modèle

- lisibilité :
  le modèle doit être simple à interpréter

Types de modèles

- Il existe différents types de modèles :
  - iconiques :
    reproduction en miniature d'un objet
  - analogiques :
    exploitant une apparence physique différente
  - analytiques :
    utilisant des relations mathématiques et logiques pour représenter les lois physiques du comportement
  - conceptuels :
    basés sur l'utilisation de symboles pour la représentation des aspects qualitatifs
- Une méthodologie ne peut pas être basée sur un seul modèle.
- Les méthodologies reposent essentiellement sur des modèles conceptuels.

Modèles conceptuels

- 3 espaces de représentation
  - espace des données
    description structurelle des données du système
    point de vue principal d'un système d'information
  - espaces des activités
    description structurelle des fonction du système
    point de vue principal d'un système de calcul
  - espaces des états
    description du comportement du système
    point de vue principal d'un système de contrôle

Modèles pour les données

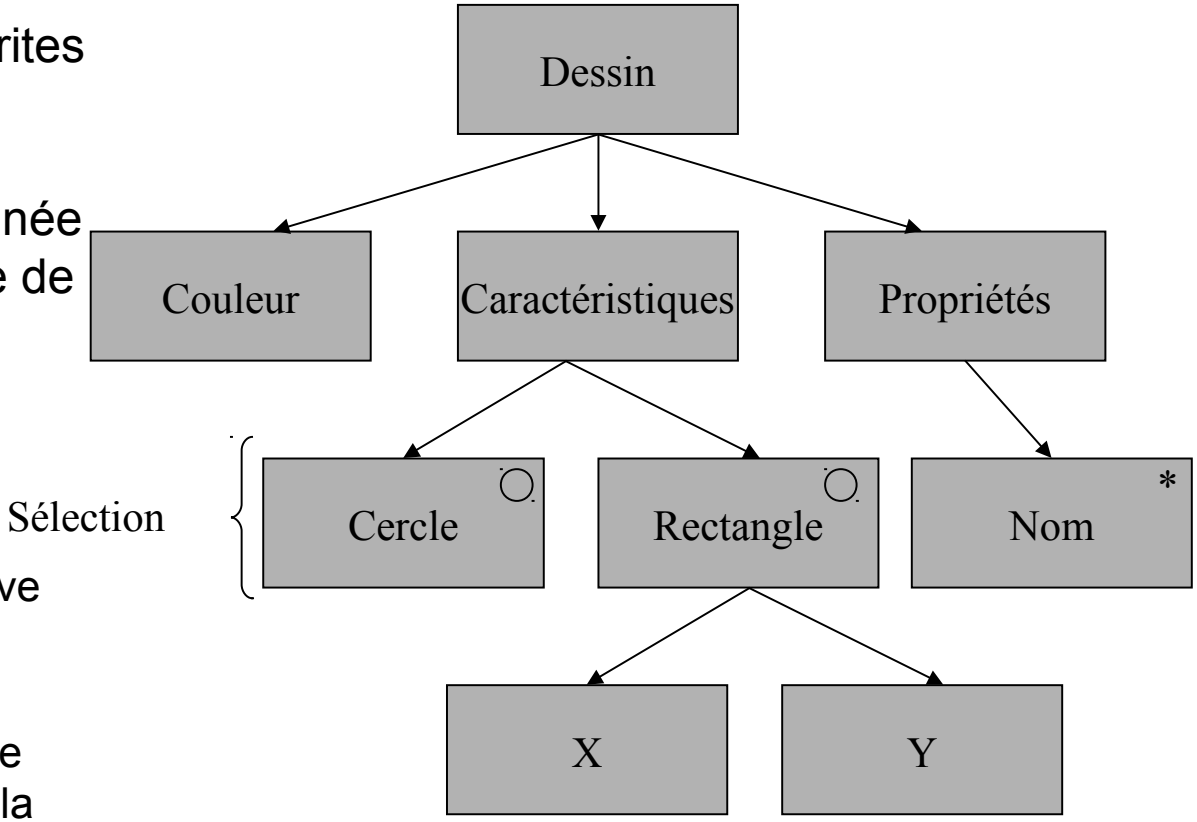
2 catégories de modèles structurels sont utilisés selon que

- la donnée forme un tout indivisible
  exemple : Diagramme de Jackson

- ou correspond à une collection de données liées entre elles
  exemple : Modèle Entités-Relations

Diagramme de Jackson

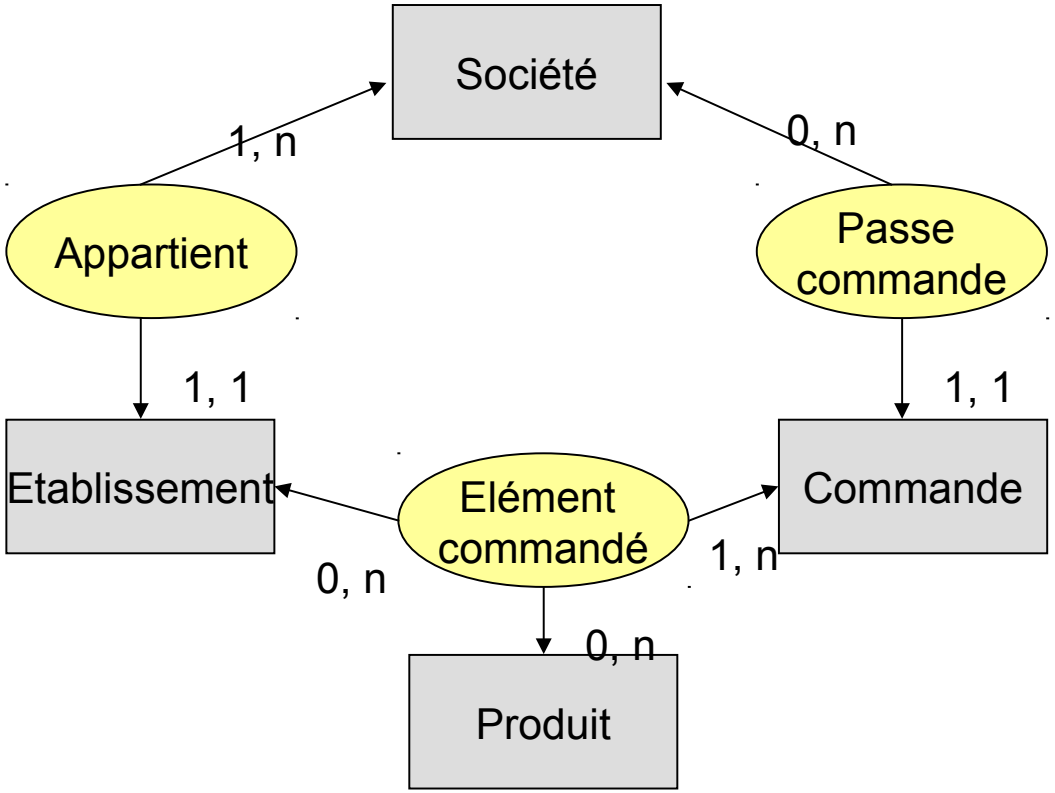
- Les données sont décrites par leur structure.
- La structure d'une donnée peut se décrire à l'aide de
  - l'élément Terminal
  - l'opérateur de Composition
  - l'opérateur Ensemble
  - l'opérateur Alternative
- Avantages
  - Lisibilité intéressante
  - Déduction facile de la structure logique.



Modèle Entités-Relations

Ce modèle permet de représenter un "monde" en termes d'entités, de leurs attributs et des relations entre ces entités.

- Une entité est une "chose" (objet, personne, information, …), ce n'est pas nécessairement un objet physique.
- Chaque type d'entité a ses propres attributs.
- Les relations expriment les faits du "monde" considéré.



Modèle Entités-Relations

- Limitation
  Modèle de description statique qui ne décrit pas de caractéristiques temporelles.

- Les bases de données, noyau des systèmes d'information, sont basées sur ce type de modèle, dit relationnel.

Modèles pour les fonctions

Cette classe de modèles permet de représenter par une structure un ensemble de fonctions interconnectées, qui peuvent aussi bien être du niveau fonctionnel que du niveau exécutif ou ressources.

2 classes :

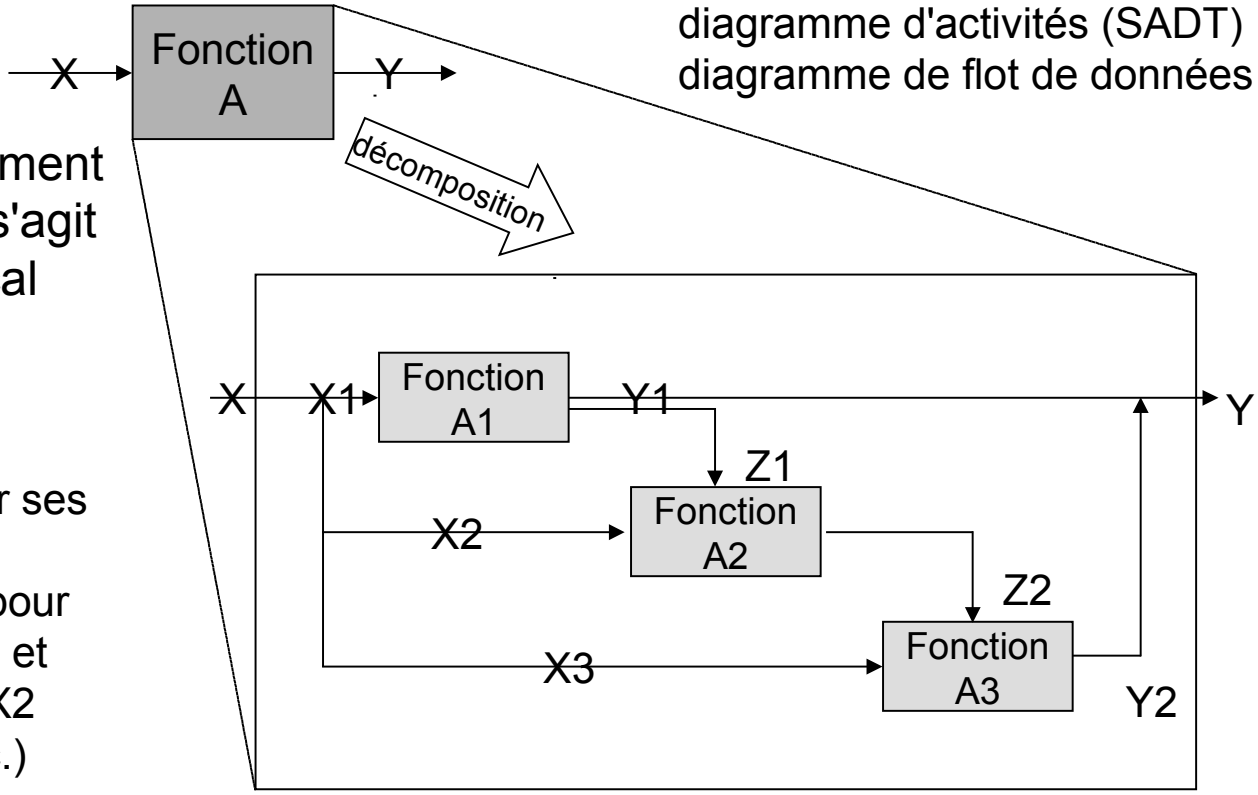
- **diagramme hiérarchique de fonctions**
- **modèle de structure objet**

Diagramme hiérarchique de fonctions

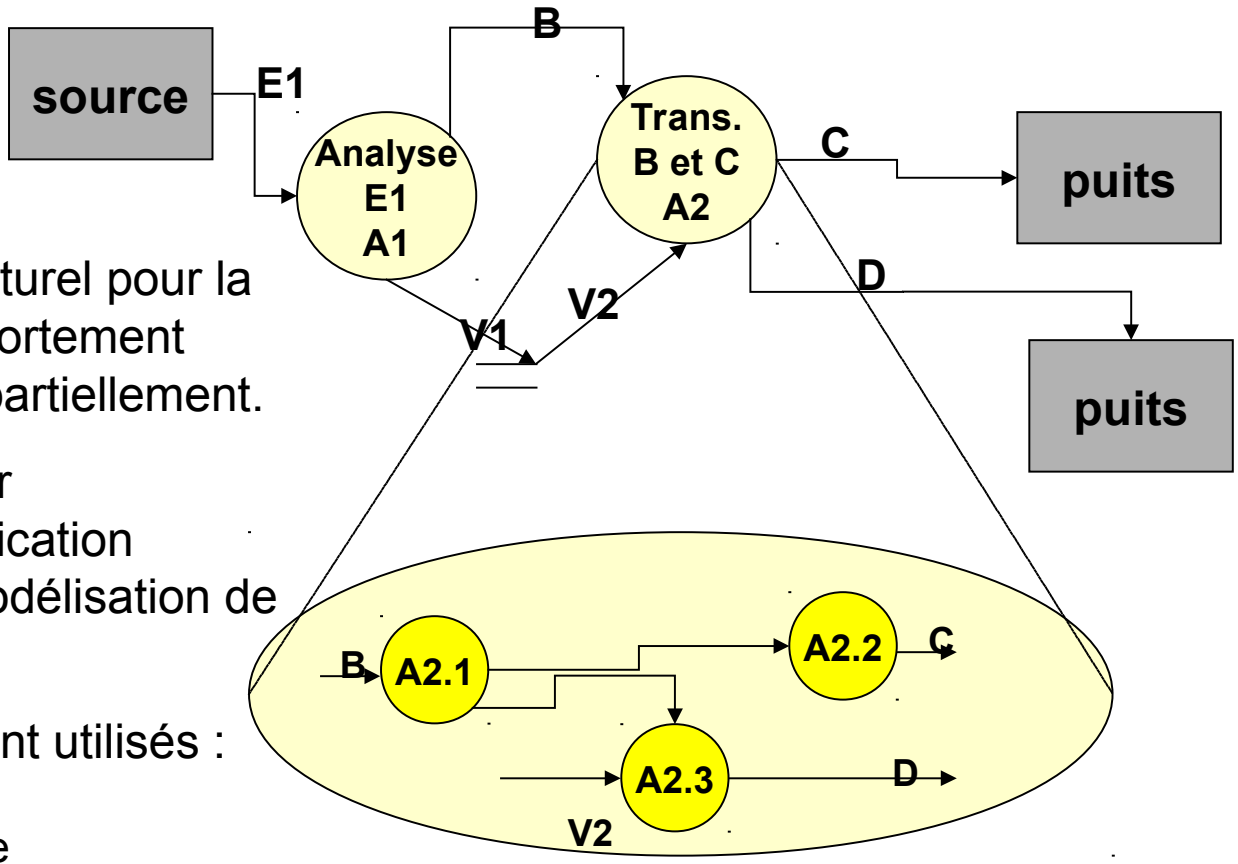
Ce type de diagramme exprime la décomposition d'un système complexe en une collection de sous-ensembles, ceux-ci reliés par des relations entrées vers sorties.

Exemples :
diagramme d'activités (SADT)
diagramme de flot de données

- Il permet le raffinement et l'abstraction. Il s'agit d'un modèle vertical
- Il n'indique pas :
  - la chronologie temporelle pour ses constituants
  - les conditions pour les entrées (X2 et Z1, X2 ou Z1, X2 suivi de Z1, etc.)



Le diagramme de flots de données



- C'est un modèle structurel pour la spécification, le comportement global y est exprimé partiellement.
- Il permet de modéliser globalement une application sans passer par la modélisation de chaque entité.
- 4 types d'éléments sont utilisés :
  - l'activité
  - la variable mémoire
  - la source et le puits
  - le lien

Le diagramme de flots de données

- Le diagramme est correct si :
  - toutes les entrées sont situées à gauche
  - toutes les sorties sont à droite
  - les données ont été nommées pour l'identification et la définition du contenu
  - les fonctions ont été nommées pour la compréhension
  - il n'y a pas de boucle.
- Avantage : C'est un modèle hiérarchique ce qui favorise l'analyse descendante.
- Limitation : il n'exprime ni le contrôle ni le séquencement temporel des actions.

Modèle de structure objet

- Ce modèle représente chaque objet par un diagramme possédant des points d'entrée et de points de sortie.
- La forme de l'objet et les caractéristiques des points d'entrée définissent assez précisément son type et donc son rôle vis-à-vis de son environnement.
- Les connexions entre les objets sont du type demandeur ou transmetteur vers demandé ou récepteur.
- Exemples : diagrammes de Booch, de Buhr, de Wasserman, de Bishop, etc.

Modèles pour le comportement

- Le comportement est très souvent exprimé en temporel.
- Cette catégorie est riche en variété de modèles :
  - modèle mathématique
  - modèles formels
  - pseudo-code
  - automate à états finis
  - statechart
  - réseau de Petri
  - etc.

Modèle mathématique

- Ce modèle décrit
  - le domaine des variables d'entrée et de sortie,
  - la transformation des entrées vers les sorties.

- Cette transformation peut être

  - du type paramétrique : expression par un modèle analytique (relation mathématique)

  - par une relation non-paramétrique (expression des grandeurs de sortie pour des entrées particulières)

Modèles formels

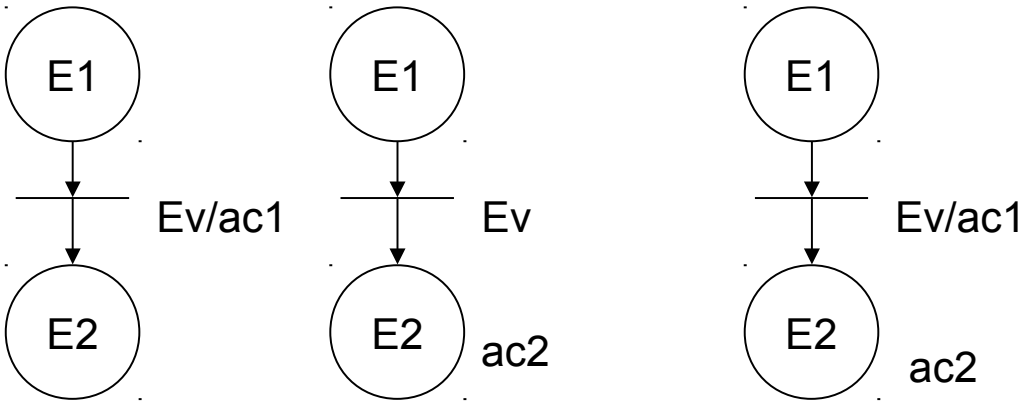
- C'est un ensemble d'énoncés exprimés dans un langage formel
- Ce modèle est composé de 2 parties
  - la définition abstraite des données et des variables, pour représenter l'état interne du système
  - la définition des opérations et des fonctions agissant sur les données et les variables. Cette définition comprend :
    - le nom de l'opération et les paramètres d'entrée et de sortie,
    - une pré-condition sur les valeurs des paramètres d'entrée et de l'état initial qui définit l'exécution,
    - une post-condition qui précise les valeurs prises par les variables et l'état final à l'issue de l'opération.

Pseudo-code

- Ce modèle utilise une description textuelle structurée.
- Le langage peut être proche du langage naturel ou des langages informatiques.

Automate à états finis

- Ce modèle permet une description comportementale simple, en exprimant les états discrets d'une entité et les conditions de changement d'état.C'est un modèle "plat".
- Un automate est caractérisé par des états et des liens entre états représentant les transitions possibles Un des états est l'état initial.
- L'expression d'une simultanéité nécessite la juxtaposition de plusieurs automates



Mealy Moore combinaison

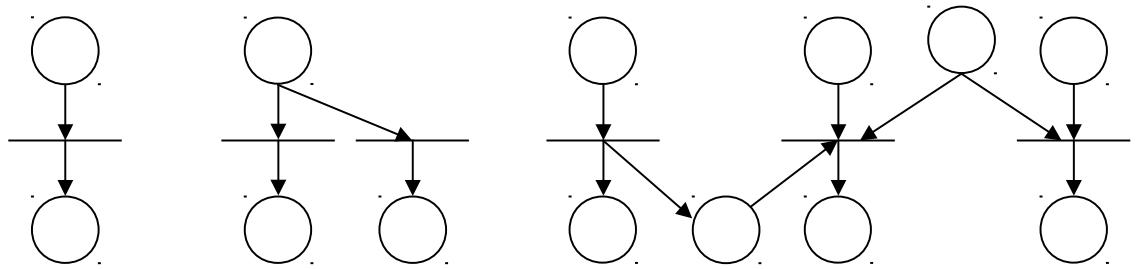
Statechart

- Extension des automates à états finis, développée par David Harel.
- Ils permettent
  - le raffinement en détaillant un état par plusieurs automates
  - la spécification d'un état de départ pour un ensemble d'automates et des synchronisations multiples,
  - l'association de contraintes de temps aux états

Réseau de Petri

- C'est un graphe bipartite composé de places et de transitions. La notion de marquage des places permet d'exprimer le séquencement temporel et la concurrence.

- Les règles d'évolution sont très simples : une transition n'est franchissable que lorsque toutes les places en amont possèdent au moins un jeton. Sur franchissement (rien ne dit quand), il y a retrait d'un jeton dans toutes les places précédentes, et ajout d'un jeton dans toutes les places suivantes.



Séquence Sélection Synchronisation Concurrence

Réseau de Petri

- Diverses interprétations sont données aux places et aux transitions :
  - place = activité, transition = condition de fin
  - place = état, transition = condition d'évolution et action atomique associée
  - place = ressources, transition = activité
- Des propriétés intéressantes pour les systèmes parallèles peuvent être vérifiées : réseau sauf, propre, vivant.
- Utile pour exprimer et valider le comportement, il ne permet pas de décrire les données. C'est un modèle "plat", qui devient rapidement complexe. Toutefois, il existe de nombreuses extensions : réseau de Petri hiérarchique, réseau de Petri coloré, etc.

Conclusion

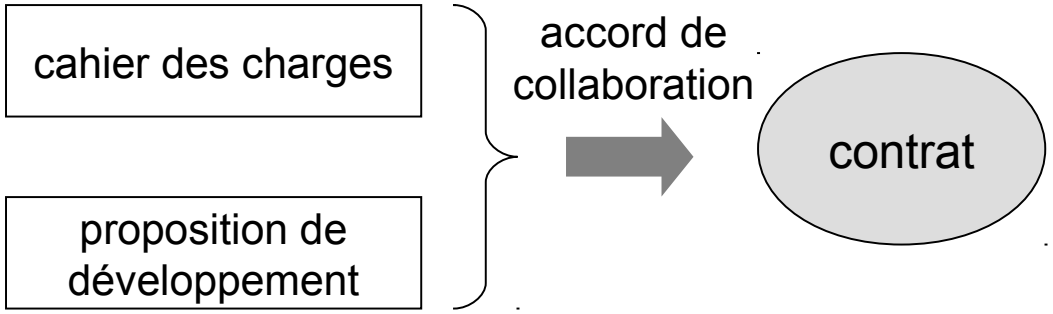
Modélisation

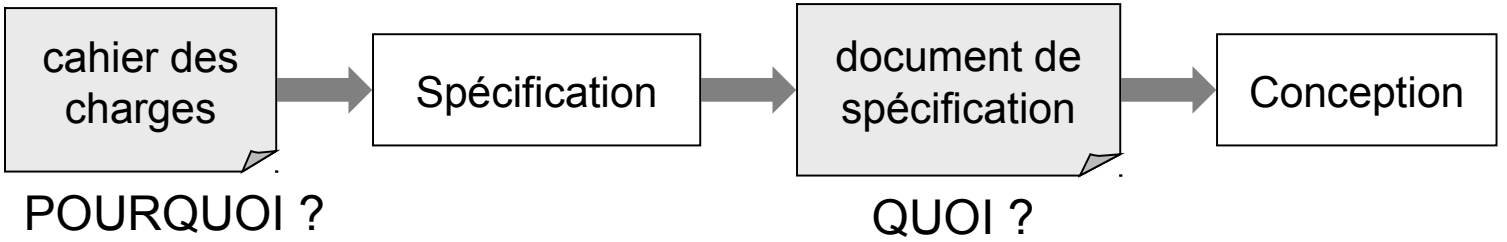
Plusieurs vues

Plusieurs modèles


La spécification

Le cahier des charges





- Le cahier des charges est exprimé et rédigé par le demandeur (client).
- Il est très incomplet et insuffisant pour décider de la réalisation. Pourtant, il conduit à la signature d'un contrat.

Plan possible d'un cahier des charges

Le contenu d'un cahier des charges est très variable.

- Présentation du document
- Présentation du produit
- Description de l 'environnement
- Description des fonctions à satisfaire
- Contraintes
- Documentation
- Plan de certification
- Plan de développement

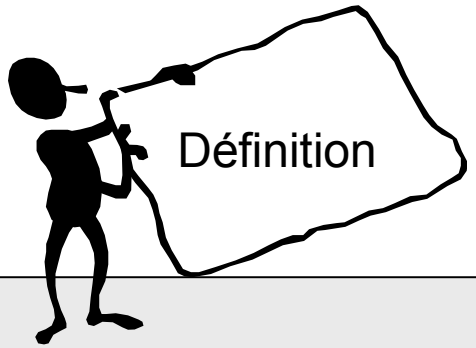
Réponse à un cahier des charges

- Présentation du prestataire
- Descriptif de la proposition
- Plan de développement
- Coût de la prestation

Le client

- Il possède l'expertise du domaine d'activité dans lequel va intervenir le système. Ce domaine peut n'avoir aucun point commun avec le domaine d'expertise des concepteurs.
- Il existe plusieurs types de clients :
  - l'acquéreur
  - les utilisateurs
  - les installateurs
  - l'équipe de maintenance
- Toutes les catégories de clients doivent être satisfaites sans compromis.

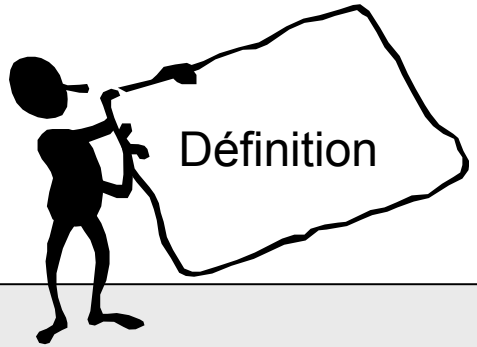
La spécification



La spécification exprime les caractéristiques du système selon une vue externe :

- comportement observable,
- propriétés,
- contraintes

Les 3 points de vue de la spécification



La spécification décrit les caractéristiques externes complètes du système selon 3 vues complémentaires :

- données QUOI ?
- activités COMMENT ?
- comportement QUAND ?

} structure

Les 3 points de vue de la spécification

- Données

  Description détaillée du problème à résoudre ; description du contenu, des relations, du flux d'information et de la structure de l'information.

- Fonctions

  Description de chacune des fonctions nécessaires à la solution du problème

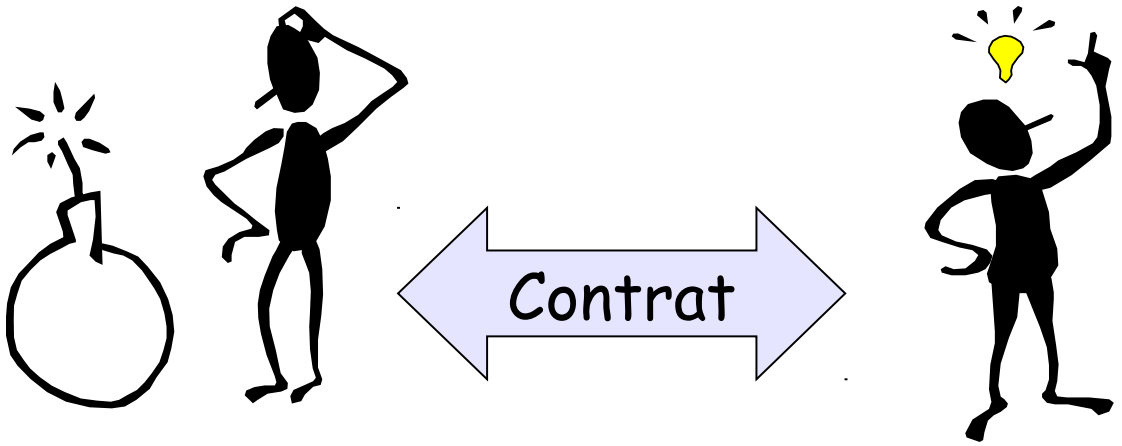
- Comportement

  Description en termes d'événements externes et description en termes de réactions internes

Le document de spécification

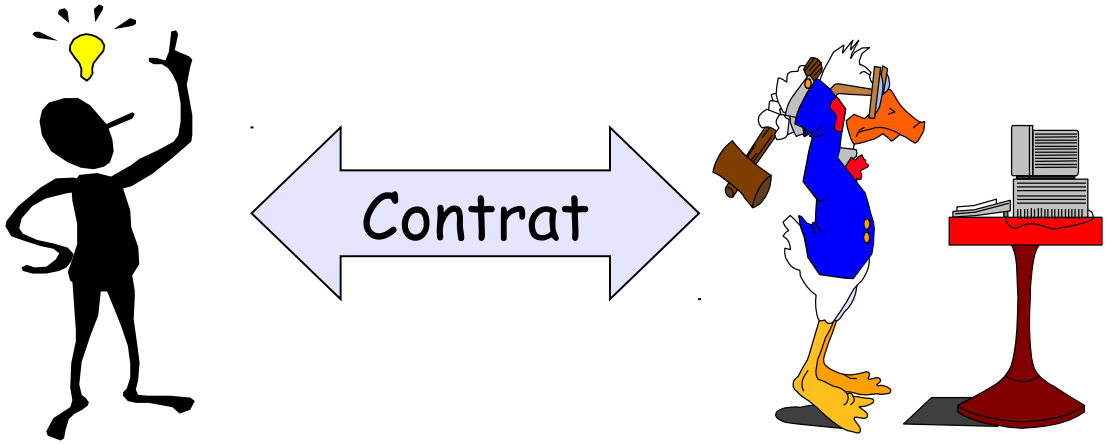
- Une spécification est une description précise des besoins du clients, écrite dans une notation (langage).
- Le document de spécification est un document vérifiable qui permet de s'assurer de la bonne compréhension du problème.
- Pour cela, le document de spécification doit être correct, complet, cohérent, compréhensible, vérifiable, exploitable par les concepteurs, modifiable ➔ lisible, testable et maintenable
- Le document de spécification
  - est une garantie en cas de mobilité du personnel,
  - permet de préparer un plan de test,
  - sert de document de négociation et de référence pour d'autres affaires (réutilisation),
  - est l'interface entre client et fournisseur,
  - sert de base pour la vérification

La spécification doit être comprise par le client



- Réécrire la spécification en langage naturel (en évitant les ambiguïtés).
- Dans le cas des spécifications exécutables, différents scénarios doivent être testés.
- Utiliser des exemples et des contre-exemples pour assurer la compréhension.

La spécification doit être comprise par les concepteurs



Une description en langage naturel justifie et explique une spécification, par exemple une description textuelle pour expliquer des formules mathématiques.

Validation et vérification de la spécification

- La validation de la spécification est l'activité consistant à s'assurer que la spécification correspond aux besoins du client. On ne peut pas prouver que la spécification est valide. Pour ce faire, il faudrait avoir une description formelle des besoins, qui si elle existait constituerait la spécification elle-même.
- On peut vérifier la spécification :
  - prouver sa consistance (pas de contradictions)
  - fixer ses propriétés et vérifier qu'elle les satisfait. Plus les propriétés sont énoncées, plus on augmente la confiance en la validation de la spécification. La spécification est considérée valide jusqu'à ce que l'on prouve qu'une propriété n'est pas satisfaite.
- La validation est un processus empirique.

Vérification et validation des spécifications

Vérifications

- Respect des règles syntaxiques des modèles utilisés
- Vérification sémantique des différents éléments
- Vérification de la cohérence de l'ensemble

Cycle auteurs-lecteurs

Les lecteurs sont définis par les auteurs et le responsable de projet et doivent représenter les catégories concernées par le résultat.

avec au moins 3 stades de lecture

Après la modélisation de l 'environnement, après les spécifications fonctionnelles, et à la fin des spécifications.

Les documents de spécification peuvent ainsi être **validés** et acceptés comme base contractuelle.

Difficultés du travail de spécification

- Une spécification n'est jamais définitive, ni pleinement satisfaisante.
- Pour spécifier correctement, il faut être à même
  - d'évaluer la faisabilité du produit ➔ compétences en conception et réalisation
  - d'estimer le coût ➔ expérience (or les situations ne sont jamais identiques)
- Une spécification joue un rôle de nature défensive : il s'agit beaucoup plus d'éviter les erreurs que de garantir le succès

La spécification

- La spécification est l'étape la plus difficile. Le spécifieur assume un travail particulièrement difficile, dont l'impact sur un projet est essentiel mais peu mesurable. Sa compétence doit être multiple.
- La spécification est le point de départ du processus de développement : elle est supposée correcte
  ➔ importance de la validation et de la vérification de la spécification
- Décrire un système du point de vue purement externe revient à élaborer un modèle de comportement de ce système. Le type de modèle dépend de la classe de problème considérée, le choix est difficile.
- Toute réalisation possédant les propriétés des modèles établis sera conforme aux spécifications.

Modèles pour la spécification

La spécification utilise

- des **modèles implicites (externes)**

  qui expriment exclusivement le comportement observable en utilisant des règles, des relations, des axiomes, des pré-conditions, des post-conditions, des prédicats.

  Exemples : spécification algébrique, spécification Z

  C'est une modélisation très formelle, peu compréhensible par un non spécialiste et plutôt adaptée pour exprimer des transformations.

- des **modèles explicites**

  qui utilisent des données, des activités ou des états a priori internes

  Exemples : diagramme de flots de données, automate à états fini, diagramme de Jackson

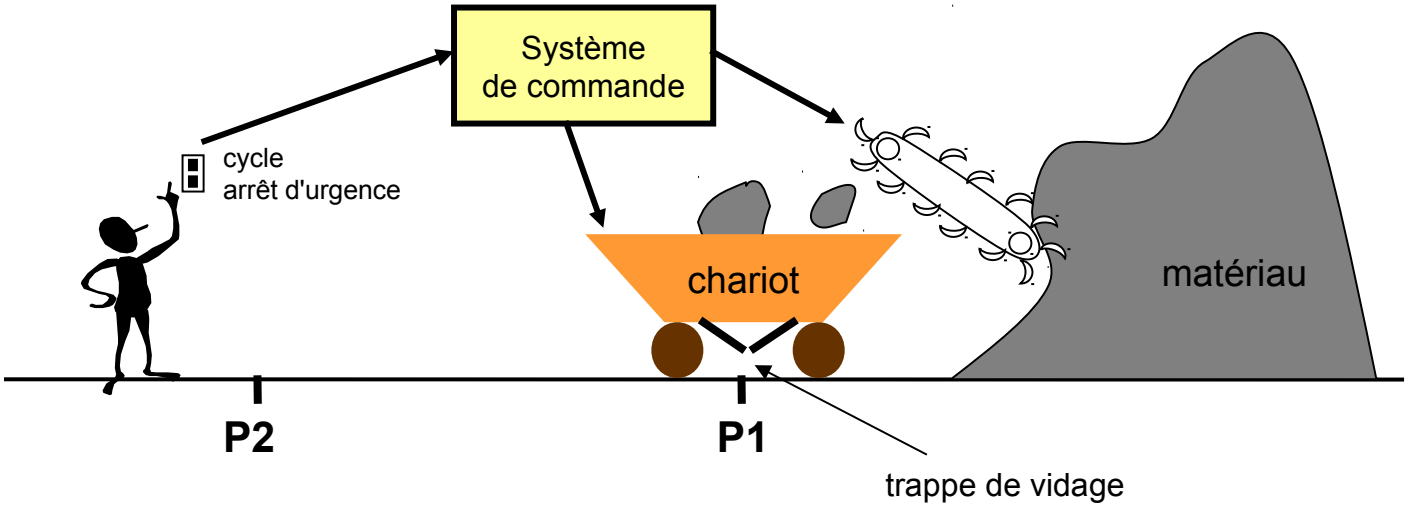
  C'est une modélisation plus simple à élaborer, interprétable par des non spécialistes et particulièrement adaptée pour décrire des objets existants.

Les 4 phases de la spécification

- Modélisation de l'environnement du système
- Définition des entrées-sorties du système
- Spécification du comportement souhaité du système
- Description de l'utilisation du système

Quand une spécification est complexe, il faut décomposer le problème en sous-problèmes fonctionnels et spécifier chacune de ses parties. La spécification est alors une collection de spécifications de sous-ensembles.

Exemple



Système de commande pour le chargement automatique d'un chariot

Un cycle consiste à déplacer le chariot de P2 à P1, à charger le chariot, à revenir en P2, et à vider le chariot.

Les objets de l'environnement

- Il faut caractériser les **objets** de l'environnement (qui existent ou vont exister) en utilisant des modèles externes ou explicites
- Le niveau d'abstraction du modèle est fonction des détails de comportement souhaité.

  Exemple : la vitesse du chariot est
  - soit 0 (arrêt), +V (déplacement à droite), ou -V (déplacement à gauche)
  - soit une fonction continue de la commande et de la charge qui introduit un effet d'inertie
- D'autre part, une modélisation peut présenter des vues différentes d'un même objet (statique, dynamique, données, activités, …)

Les objets de l'environnement

- Un **objet** peut avoir une existence physique ou matérielle (un utilisateur, un chariot) ou être un objet fonctionnel ou logique (un département d'une entreprise) ou un objet abstrait (le vol d'un avion).
- Pour faciliter la modélisation, les objets sont groupés en catégories (**types**) reflétant chacune des caractéristiques communes.
- Chaque type a des éléments caractéristiques appelés **attributs**. Les valeurs des attributs permettent de différencier les objets d'un même type. Exemple : Le type personne a les attributs poids et taille.
- Il existe 3 catégories d'attributs :
  - **événement**
  - **donnée**
  - **activité /action**

Les événements

- Un **événement** spécifie l'instant d'un changement d'état significatif

  Exemple : la température a dépassé 50°C.

- Au changement d'état peut être associée l'apparition d'une donnée

  Exemple : la consigne de température vient d'être modifiée (événement auquel est associé la nouvelle valeur de la consigne).

Les données

- Une **donnée** est une grandeur ou un ensemble de grandeurs dont l'existence est considérée permanente. Seule sa valeur évolue dans le temps.

  Exemple : Le chariot est caractérisé par plusieurs variables, les attributs : position, vitesse, quantité de matériau transportée.

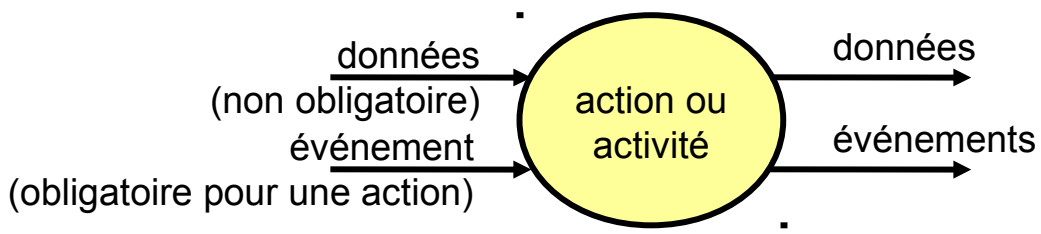
- Cas particulier : une **variable d'état** est une grandeur permanente servant à caractériser de manière unique et au niveau d'abstraction souhaité, l'état de l'objet.

  Exemple : Une modélisation du chariot pourrait utiliser 2 variables d'état P (position) et V (vitesse) avec :

  - pour P : en P2, entre P2 et P1, en P1
  - pour V : arrêt, déplacement à gauche, déplacement à droite

Les actions et les activités

- Une **action** est une opération ponctuelle agissant à un instant donné durant l'évolution de l'objet ; elle est indécomposable (atomique).
- Une **activité** est une vue de plus haut niveau, représentant l'enchaînement d'une suite d'actions. Une activité couvre une période de temps. Une activité est une transformation de données en entrée en d'autres données en sortie.
- Une activité se caractérise par un état (actif ou inactif), alors qu'une action se caractérise par un événement.
- Exemple : L'action de mise en marche du chariot par le système génère une activité de déplacement (et donc l'état en déplacement).



Les objets de l'environnement

- Le caractère dynamique d'un objet résulte des activités et actions internes. Seuls les événements, les données et les états assurent le couplage avec son environnement.

- Il y a 3 vues complémentaires et indissociables de description d'un objet : les données, les activités et le comportement

1° phase : Modélisation de l'environnement

Modélisation des données : identification des données utiles pour le problème et description de ces données et des relations entre données.

- Exemple : Données voiture (date d'achat, propriétaire, numéro d'immatriculation) et personne. Relation : une personne est propriétaire d'une voiture.
- Définition complète, concise et compréhensible de la nature et de la structure des données : type, composition, type de chaque constituant.
  ➔ modèle de composition hiérarchique (opérateurs de composition, alternative, ensemble, héritage) et modèle entité / relation
- Description conceptuelle (signification et non pas implémentation)
- Modélisation statique, indépendante des 2 autres vues (activités, comportement)

1° phase : Modélisation de l'environnement

Modélisation du comportement : Description des instants d'apparition des événements et des changements d'état des données engendrés par les activités, i.e. expression des dépendances entre événements et actions.

➔ modèle de type continu

Description d'une évolution permanente. Modèles mathématiques.

➔ ou modèle à états discrets

Description limitée à des états particuliers qui ne changent qu'à des instants particuliers. Modèles de type état / transition (diagramme à états finis, grafcet, statechart, réseau de Petri) avec une condition associée à chaque transition, les activités associées aux états et les actions associées aux transitions. Une condition est définie par une occurence d'événement et/ou un état particulier des données.

2 démarches : modélisation par les états ou par les stimuli/réponse.

Il faut utiliser des modèles de type continu lorsque la modélisation ne peut pas se faire sur la base d'états discrets.

Modélisation dynamique et hiérarchique.

1° phase : Modélisation de l'environnement

Modélisation des activités : Description des relations entre les données ou les événements et les activités d'un objet. La modélisation est basée sur l'utilisation d'un graphe des dépendances, exemples : modèle SADT, diagramme de flot de données.

Description hiérarchique telle que, au niveau le plus bas, une activité ne peut être décrite que par sa spécification qui caractérise la transformation des données, c'est-à-dire une modélisation du comportement.

Cette modélisation concerne le cas le plus général où un objet est caractérisé par une collection de données et d'événements et une collection d'activités.

1° phase : Modélisation de l'environnement

- La complexité de la modélisation selon les 3 vues est fonction de la nature de l'objet à décrire et du niveau de détail souhaité.
- Il y a redondance entre les 3 vues ➔ vérification de la cohérence
- Enfin, la synthèse fonctionnelle fait apparaître les relations entre les objets de l'environnement, pour les objets couplés, ce qui permet :
  - de présenter une vue graphique synthétique de tout l'environnement
  - de pouvoir éventuellement regrouper des objets pour simplifier la suite des spécifications

1° phase : Modélisation de l'environnement

Démarche

- Déterminer les objets utiles de l'environnement (ceux qui ont une influence directe sur le système)
- Caractériser les détails nécessaires et suffisants pour chaque objet
- Les classer en catégories :
  - données (données, variables d'état, événements),
  - activités (actions, activités),
  - relations (relations entre données et activités, entrées et sorties avec l'environnement)
- Modélisation selon les 3 vues

  Si la modélisation des données conduit à définir toutes les entrées et les sorties du système, elle est suffisante. Sinon, il faut encore modéliser le comportement dynamique de l'objet. Enfin, la modélisation des activités ne doit être faite que pour les objets relativement complexes (plusieurs activités simultanées).

  La modélisation de l'utilisateur comme objet est particulière.
- Description fonctionnelle de l'environnement

1° phase : Modélisation de l'environnement

Exemple :

- 4 objets identifiés

  le système de contrôle-commande

  l'opérateur
  le chariot
  le tapis-chargeur
  } objets de l'environnement

- Le chariot

  3 variables à états discrets :

  P (position) : P $\leq$ P2, P2 < P < P1, P $\geq$ P1

  V (vitesse) : Déplacement_droite, Déplacement_gauche, Arrêt

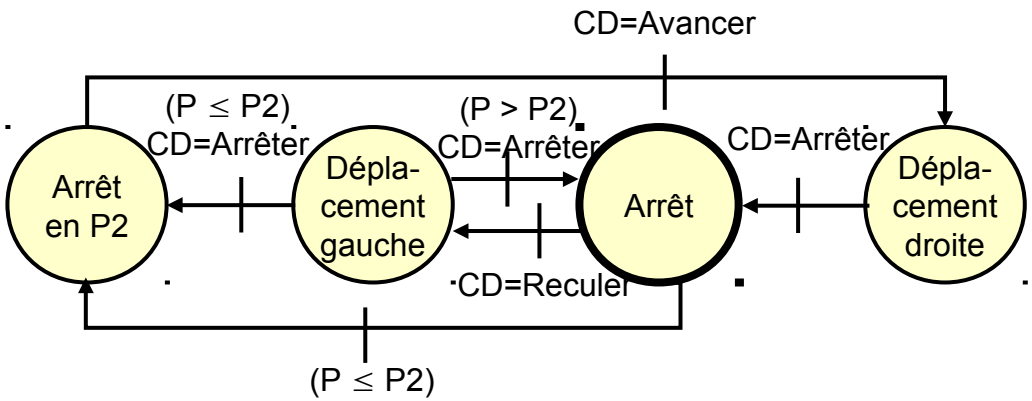
  T (trappe) : Ouverte, Fermée

  V dépend de la commande de déplacement appliquée : Avancer, Reculer, Arrêter.

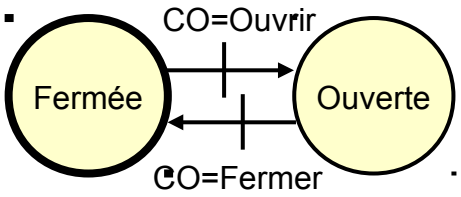
  T dépend de la commande d'ouverture appliquée : Ouvrir, Fermer.

1° phase : Modélisation de l'environnement

Déplacement



Trappe



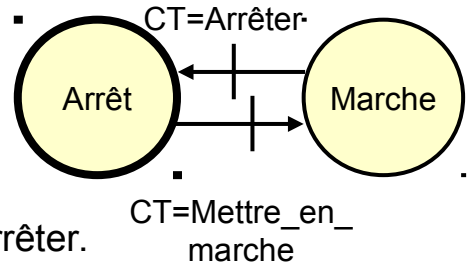
1° phase : Modélisation de l'environnement

- Le tapis-chargeur

  1 variable à états discrets :

  M (marche) : Vrai (en marche), Faux (arrêté)

  M dépend de la commande appliquée : Mettre_en_marche, Arrêter.



- L'opérateur

Source	Entrée	Action	Sortie	Destinataire
Opérateur	Cycle	Réaliser un cycle de transfert	CD/CO	Chariot
			CT	Tapis-chargeur
	Arrêt	Arrêt d 'urgence	CD/CO	Chariot
			CT	Tapis-chargeur

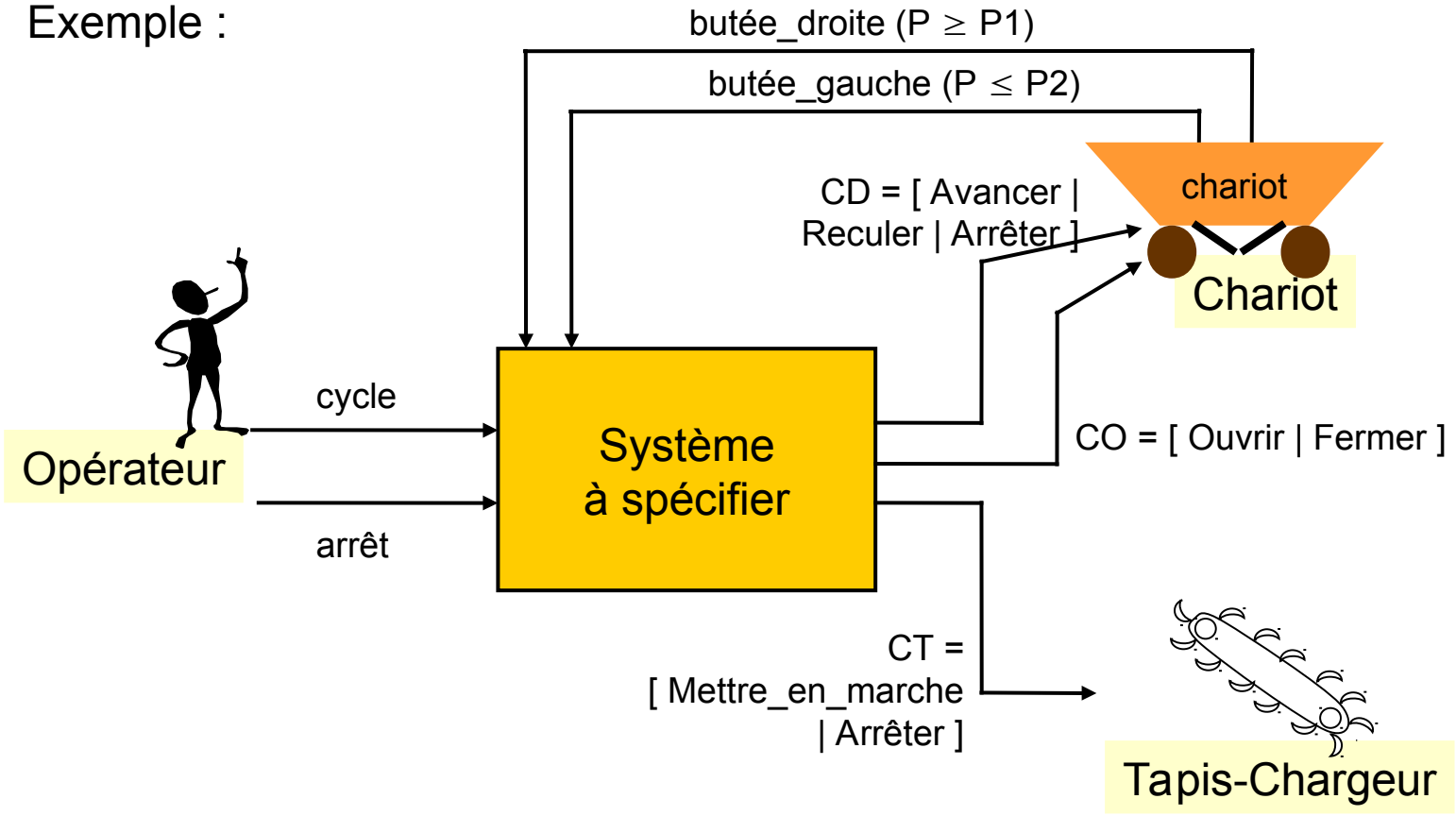
- Les objets ne sont pas liés entre eux en l'absence du système de contrôle-commande.

2° phase : Détermination des entrées et des sorties

- Cette phase conduit à délimiter le système en décrivant ses relations avec l'environnement.
  - Les entrées du système sont les sorties d'objets de l'environnement par lesquelles le système observe l'état de l'environnement
  - Les sorties du système sont des entrées pour les objets, par lesquelles le système modifie l'état de l'environnement
- Elle n'aboutit pas obligatoirement à la connaissance de toutes les entrées et les sorties indispensables. La délimitation devra alors être complétée ultérieurement.

2° phase : Détermination des entrées et des sorties

Exemple :



Les entrées butée_droite et butée_gauche ont été ajoutées tout de suite car elles s'avéreront nécessaires pour la réalisation de la fonction.

3° phase : Spécifications du système

- Les spécifications fonctionnelles décrivent les fonctions (externes) que doit assurer le système pour son environnement.
  - Une fonction externe est une fonction globale qui résulte de la collaboration du système avec des objets de son environnement, exemple : "réalisation d'un cycle de transport de matériau"
  - Elles expriment les actions effectuées sur les objets de l'environnement par les sorties du système en fonction des entrées ➔ utilisation des mêmes modèles et techniques de modélisation que dans la 1° phase
  - Un système réalisant en général plusieurs fonctionnalités, il faut le caractériser par sous-ensembles.
  - La spécification liée à l'utilisateur, le cas échéant, exprime les séquences d'événements qu'il produit, considérées comme correctes pour l'application (graphe de type stimuli/réponse).

3° phase : Spécifications du système

- Si la spécification fonctionnelle est exprimée par un graphe de type état / transition, alors
  - les états doivent être significatifs pour le système
  - les actions concernent les sorties du système ou des événements
  - les conditions utilisent exclusivement des entrées du système, des états de la spécification, des événements, des durées (temps absolu ou relatif)

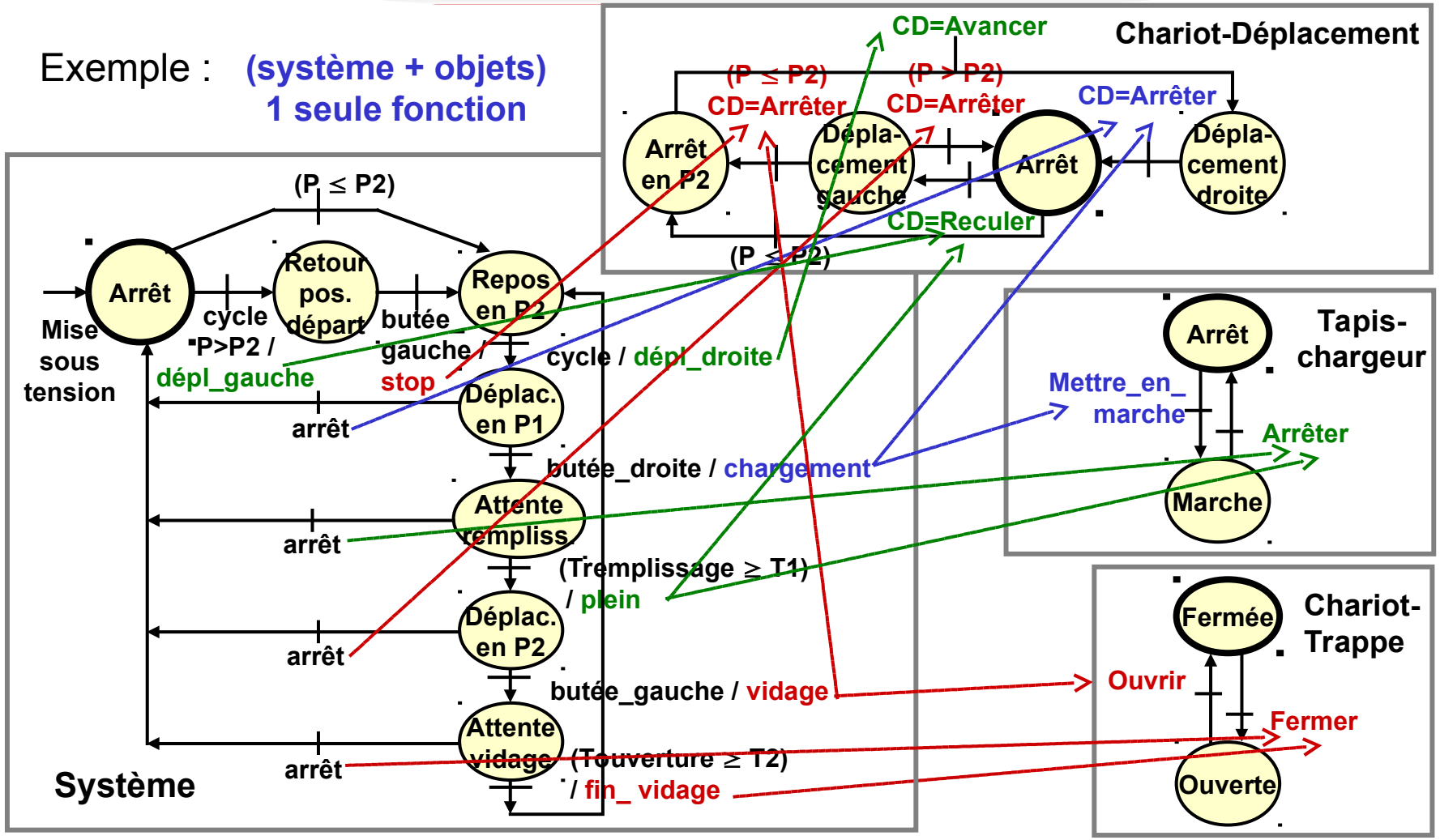
3° phase : Spécifications du système

- Les spécifications opératoires précisent la manière dont une fonction doit opérer et les conditions et domaines de fonctionnement
  Exemples : précisions sur les grandeurs apparues dans les spécifications fonctionnelles (type, domaine de définition, performance en précision, etc.), méthode de calcul proposée par le demandeur, performances du système.

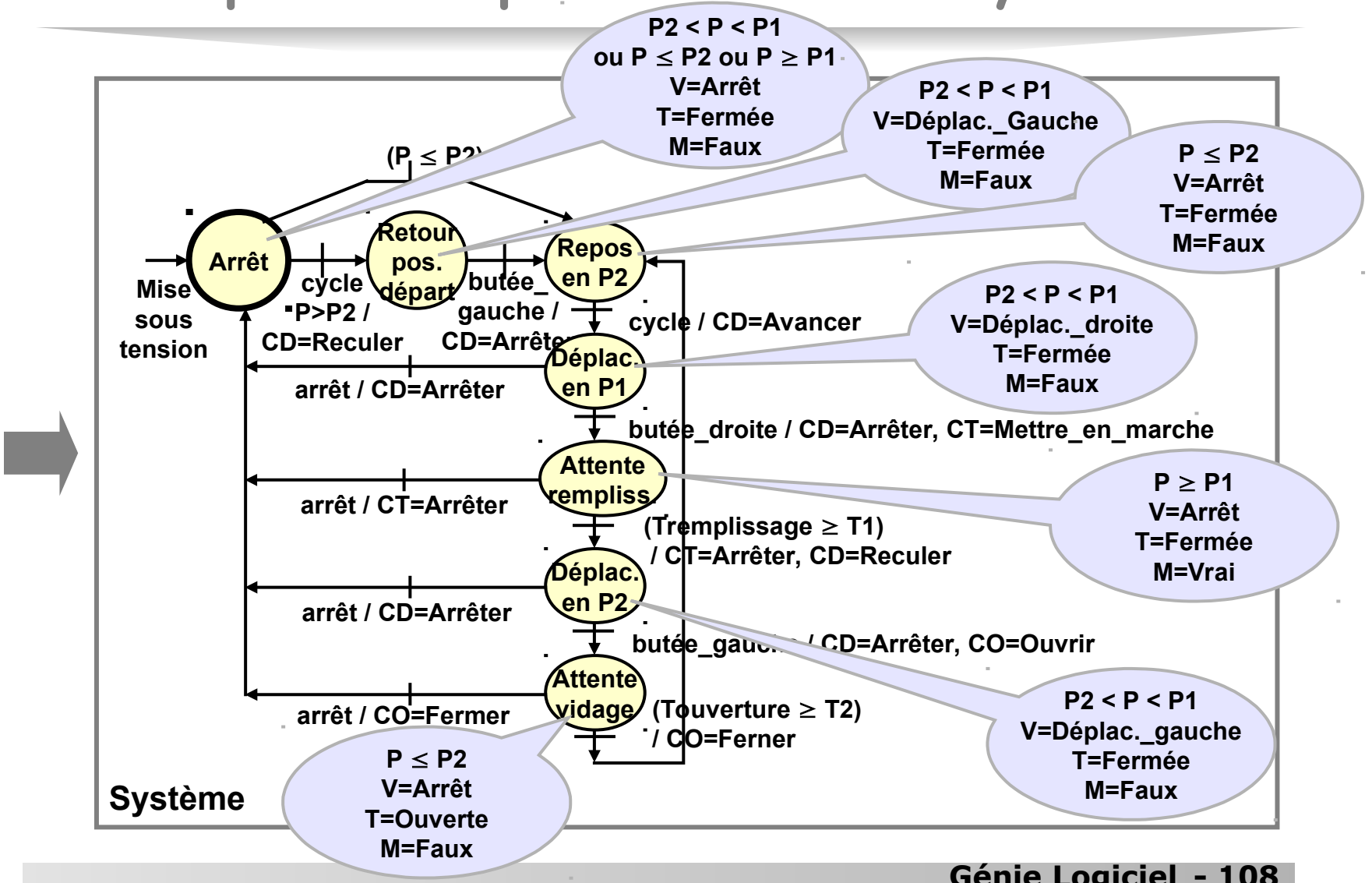
- Les spécifications technologiques décrivent toutes les spécifications en rapport avec la réalisation matérielle et l'implantation logicielle.
  Exemples : contraintes de répartition, spécifications des interfaces homme-machine, contraintes de temps, fonctions pour la maintenance ou la sûreté de fonctionnement

3° phase : Spécifications du système

Exemple : (système + objets)
1 seule fonction



3° phase : Spécifications du système



4° phase : Procédures d'installation et d'exploitation

Deux documents viennent compléter les spécifications :

- le document préliminaire d'utilisation
- le document d'installation

Résumé

- La spécification commence par la délimitation de son environnement et la caractérisation des objets qui composent cet environnement.
- Les objets sont caractérisables par un modèle explicite, tandis que le système ne peut être décrit que selon une vue externe. Ainsi, la modélisation du système sera basée sur le comportement souhaité de son environnement.
- Un objet se décrit par un ensemble d'éléments caractéristiques (événements, données, activités) et par des relations.
- La modélisation est basée sur une description selon 3 vues :
  - par les données pour une description statique,
  - par le comportement pour une description dynamique globale,
  - par les activités, dans le cas le plus général, lorsque les modélisations précédentes sont insuffisantes.

Résumé

- L'environnement est caractérisé en analysant le comportement des objets de l'application. Le système est lui-même considéré comme un objet complémentaire rapporté pour effectuer un couplage fonctionnel. Le système est spécifié par une description externe la plus complète possible.
- La spécification complète du système peut être structurée en 3 catégories (fonctionnel, opératoire, technologique) pour faciliter son utilisation dans la suite du développement.
- La phase de spécification conduit à élaborer progressivement un document complet de spécification. Ce document peut être vérifié et validé pour devenir un document contractuel.