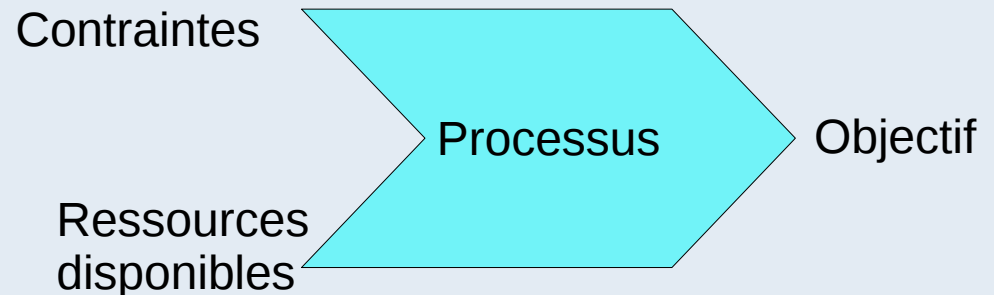


Analyse et Conception d'un S.I.

- **Définitions d'un projet**
- **Étapes et modèles d'un projet de S.I.**
 - Le tunnel, La cascade, le V, Mod. incréments, Mod. Itératif
- **Méthodes**
 - SADT, MERISE, Méthodes AGILES, RAD, XP
 - UP
- **Sur l'analyse et la modélisation**
- **UML**
- **Design Patterns**

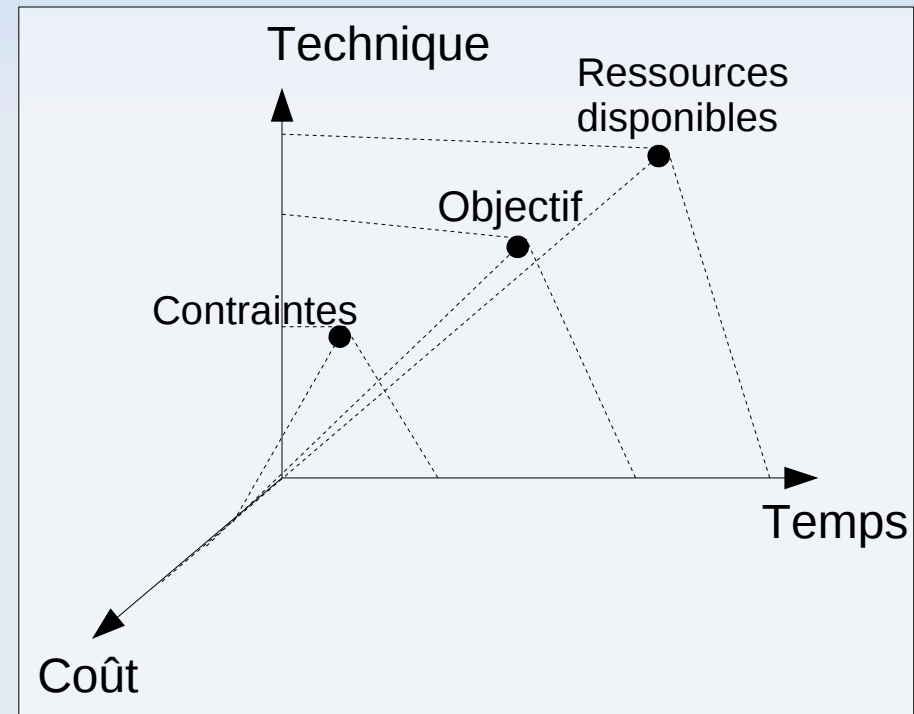
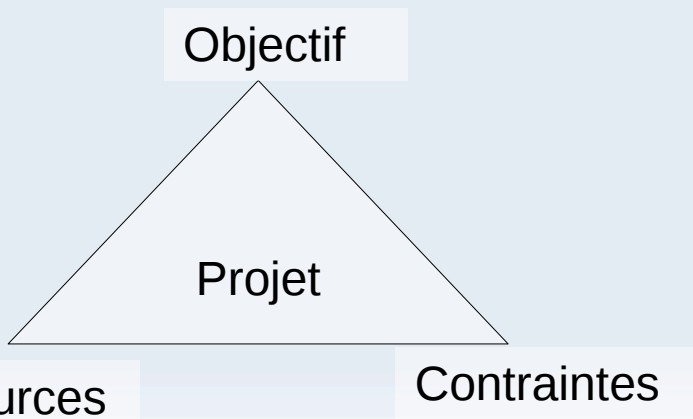
Définitions d'un projet

- **Projet :**
 - Un projet est processus réfléchi pour atteindre un objectif spécifique en un temps fini, devant respecter un échéancier et un budget, et qui est unique et en général non répétitif dans l'organisation. (Jack Meredith et Samuel Mantel 1989)
 - Objectif : ce que l'on veut faire
 - Ressources disponibles : avec quoi
 - Contraintes : sous quelles limites
 - Acteurs :
 - Client
 - Maître d'ouvrage
 - Maître d'œuvre



Définitions d'un projet

- Chaque pôle peut être quantifié par la prise en compte :
 - du coût, pour le budget
 - du temps, pour les délais
 - de la technique pour les spécifications



Définitions d'un projet

- Réussite d'un projet :
 - le résultat doit être conforme aux spécifications techniques dans le respect de l'échéance et du budget;
 - le résultat doit avoir été accepté et utilisé par le client;
 - la réalisation du projet doit avoir satisfait ses parties prenantes ;
 - le résultat doit avoir été intégré dans l'organisation et mise en production ;

Définitions d'un projet

- Problèmes :
 - L'objectif est souvent mal défini au départ et ses spécifications évolues
 - Les contraintes évoluent au cours du temps
 - Interactions entre l'objectif, les ressources et les contraintes.

Définitions d'un projet

- Principes des solutions générales :
 - Décomposer le projet sous-problèmes plus simples
 - Chaque sous-problème doit être relativement isolé des autres sinon la complexité sera reportée lors de l'intégration
 - Décomposition fonctionnelle et temporelle
 - Quantifier les ressources pour chaque sous-problème
 - Affecter ces ressources en fonction des contraintes

Définitions d'un projet

- Décomposition fonctionnelle :
 - Chaque découpage donne un résultat bien défini
 - Chaque découpage peut être quantifié par les ressources
 - Décomposition du haut vers le bas
 - Possibilité d'optimiser en factorisant les modules communs
 - Exemple : Work Breakdown Structure
 - Approche descendante
 - Modélisation arborescente
 - Utilisé en début de projet, épine dorsale du projet (GANTT)

Définitions d'un projet

- Décomposition temporelle :
 - Définition en phases suivant le schéma type suivant
 - Étude des besoins et d'opportunité: ce que l'on veut, pour qui, combien, estimation retour sur investissement (difficile). Base du projet
 - Étude de faisabilité : ce qui est possible de faire (temps, coût et technique) et comment. Choix de la solution à développer.
 - Analyse : projeter les besoins sur la solution retenue. Cahier des charges.
 - Conception : concevoir la solution pour répondre au cahier des charges
 - Réalisation : concrétiser la conception, et valider le résultat unitaire
 - Intégration : assembler les solutions unitaires. Les placer dans le contexte final
 - Qualification : valid. résultat-cahier des charges, livraison, recette, fin du projet
 - ET ne pas oublier la formation et la maintenance !!!

Étapes d'un projet SI

- Les grandes phases d'un projet informatique :
 - Spécificités du logiciel (souplesse, demandes et solutions complexes)

Phase élabor.	But	Acteurs principaux
Besoins	objectifs	client, experts métier
Faisabilité	réalistes	architecte SI, client, experts
Analyse	cahier des charges	architecte-analyste, experts, client
Conception	modules, tests	Analyste-programmeurs
Implémentation	programme	programmeurs
Tests unitaires	validation locale	programmeurs, testeurs
Tests d'intégration	validation conception	testeurs, programmeurs, analyste
Validation globale	recette, livraison	client, architecte, analyste
Maintenance	ajustements, consolidat.	assistance, Service clientèle

Étapes d'un projet SI

- Norme IEEE 1074 (1997)

Process Group	Processes
Life Cycle Modeling	Selection of e Life Cycle Modeling
Project Management	Projet Initiation
	Project Monitoring and Control
	Software Quality Management
Pre-development	Concept Exploration
	System Allocation
Development	Requirements
	Design
	Implementation
Post-development	Installation
	Operation and Support
	Maintenance
	Retirement
Integral Processes	Verification and Validation
	Software Configuration Management

- http://standards.ieee.org/reading/ieee/std_public/description/se/1074-1997_desc.html

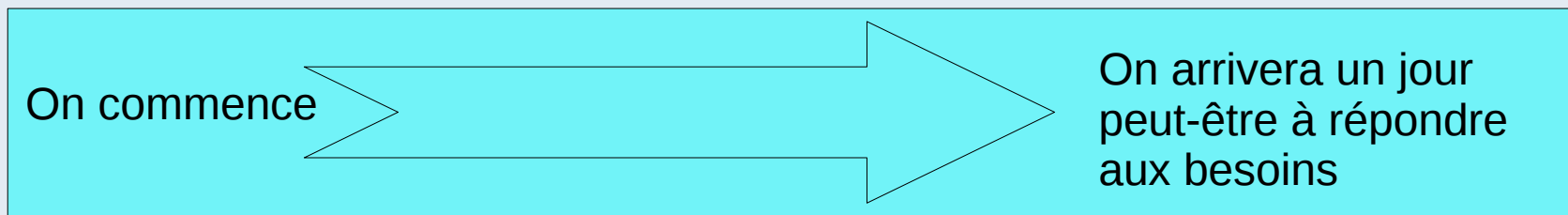
Modèles de projet SI

- Modèles d'un projet informatique
 - **Le tunnel** : on connaît le début on sait ce que l'on veut, entre, c'est la boîte noire ...
 - **La cascade** (Royce, 1970) : analyse, conception, codage, tests et validation
 - **Le modèle en V** : variante avec un ensemble de repères prédéfinis pour les validations
 - **Modèle itératif** (1988) : le projet s'affine par petits objectifs prévus
 - **Modèle par incréments** : un noyau croît par rajout de nouveaux objectifs simples.

Modèles de projet SI

- Le Tunnel

- On a des besoins ... de programmer !
- Modèle rassemblant typiquement de tout petits développements ponctuels, sans ambitions.
- Développement à 1 ou peu de participants
- Pas de structure, pas de jalons, seul compte le résultat
- Peu efficace plus le temps avance. Réservé à ceux qui ont les idées claires !

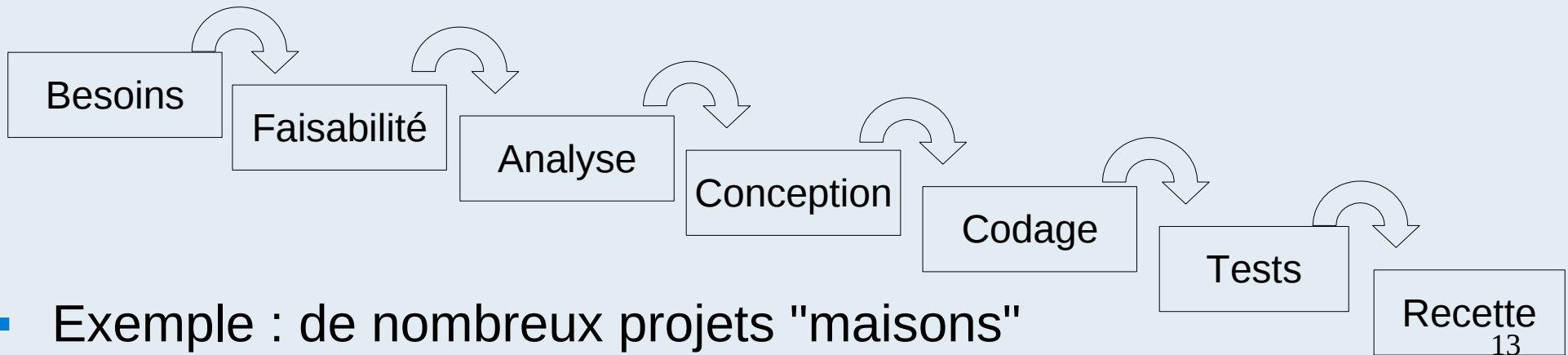


- Exemple : Linux ... à ses débuts uniquement

Modèles de projet SI

■ La Cascade ou Développement Linéaire

- Directement inspiré des modèles industriels ou du bâtiment
- Besoins, Faisabilité, Analyse, Conception, Codage, Tests, Recette
- Enchaînement linéaire des étapes, jalonné par des documents qui servent de point de mesure sur l'avancée du projet
- Mais pas ou peu de retour. Tout doit être prévu à l'avance.
- Sert de schéma de base pour les modèles suivants

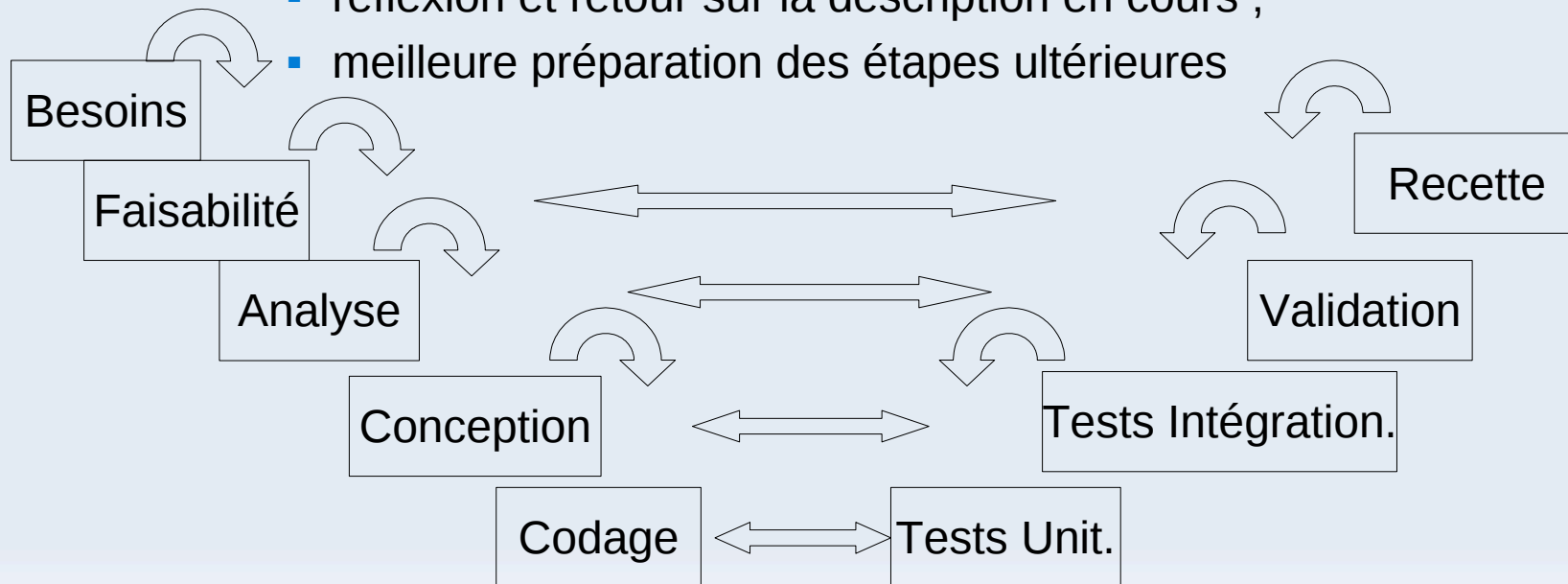


- Exemple : de nombreux projets "maisons"

Modèles de projet SI

■ Le V

- Reprends le modèle en cascade
- Rajoute la nécessité de préparer les phases ultérieures au fur et à mesure de l'avancement
 - obligation de concevoir les jeux de test et leurs résultats ;
 - réflexion et retour sur la description en cours ;
 - meilleure préparation des étapes ultérieures

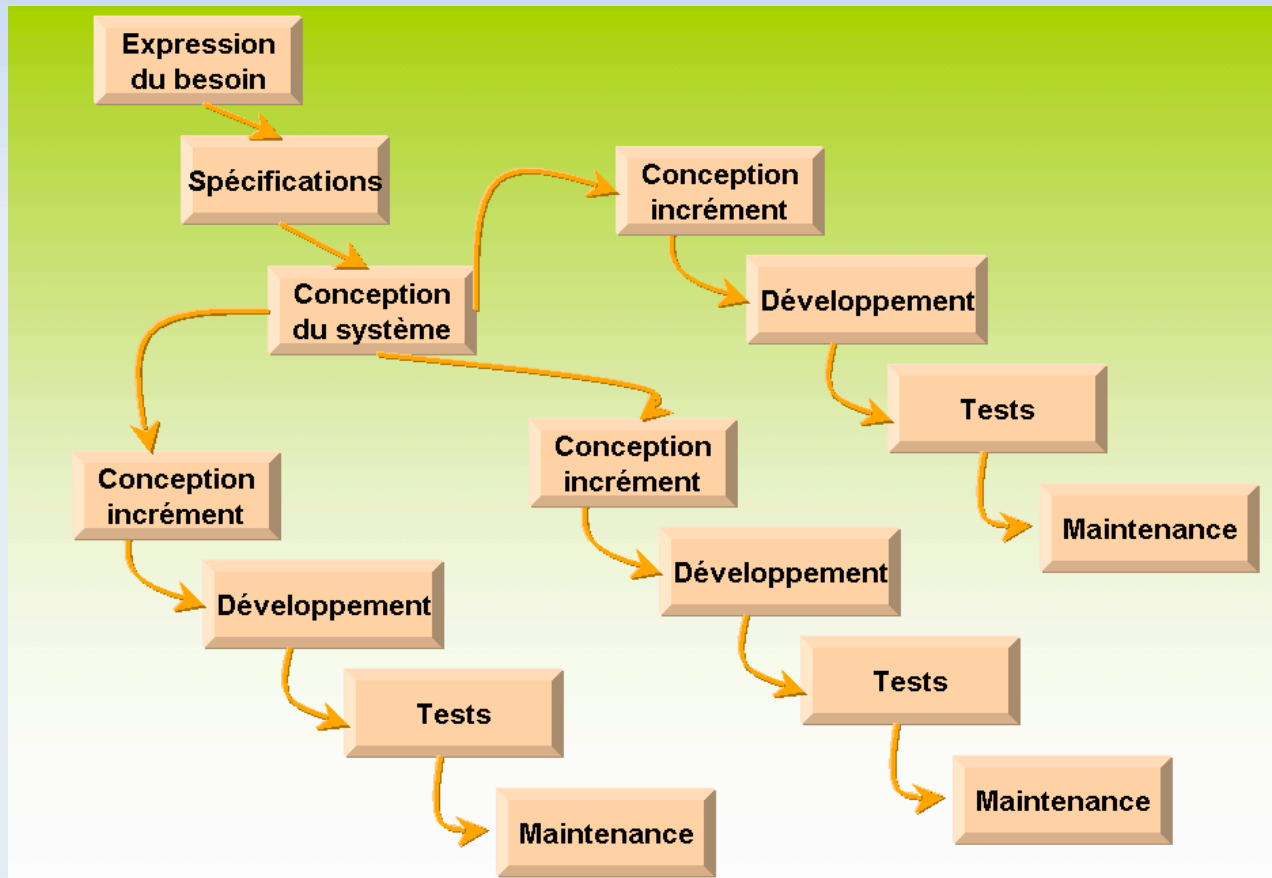


Modèles de projet SI

- Le modèle par Incréments
 - Le projet est subdivisé en plusieurs sous-projets, chacun adoptant le modèle en cascade ou en V.
 - On classe les sous-projets par ordre d'importance. Le premier développement concerne la partie la plus importante du projet, le noyau. Le noyau est complété au fur et à mesure de l'avancée du projet.
 - Avantages :
 - La complexité du projet général est subdivisée en autant de parties que d'incrément
 - On ajoute de la souplesse dans la conduite du projet.
 - Ainsi, on intègre la notion de risque
 - Inconvénient :
 - Le noyau doit être solide, il ne doit pas être retouché sinon tout l'édifice tombe.

Modèles de projet SI

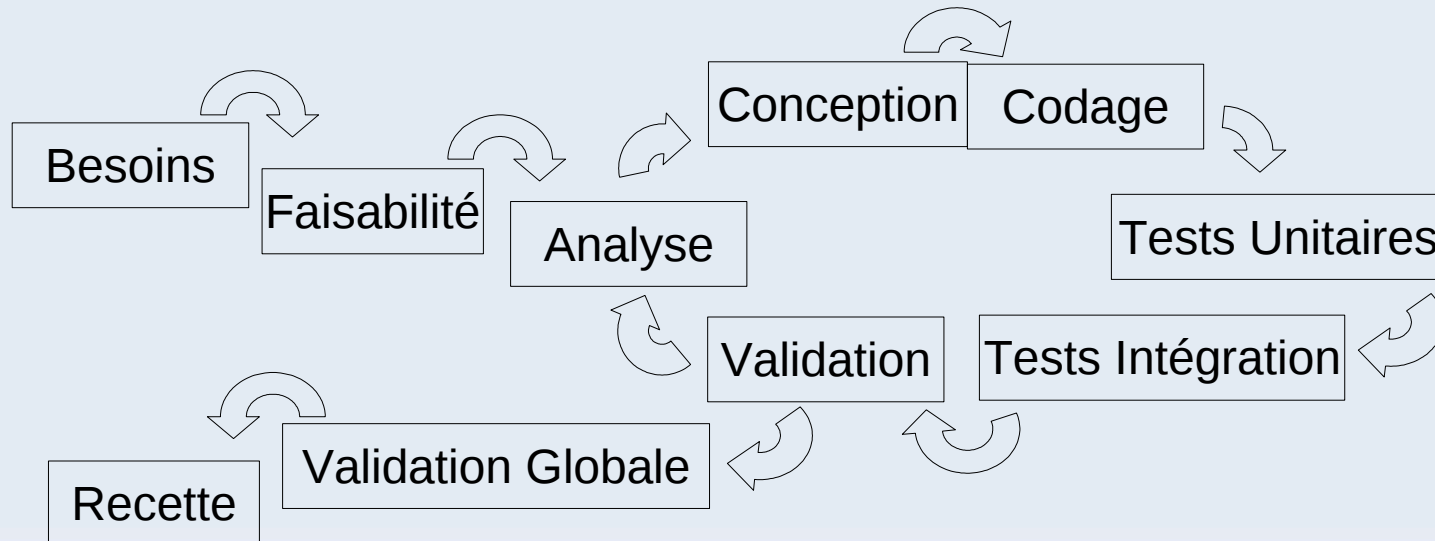
- Le modèle par incréments



Modèles de projet SI

■ Le modèle Itératif

- Reprends le modèle en V , mais le projet final est le résultat d'une répétition de plusieurs cycles de développement en V.
- Chaque cycle permet de revoir les besoins et les objectifs. Tout n'est pas prédéfinis par avance contrairement aux modèles précédents.
- Toutes les étapes peuvent être intégrées au cycle.



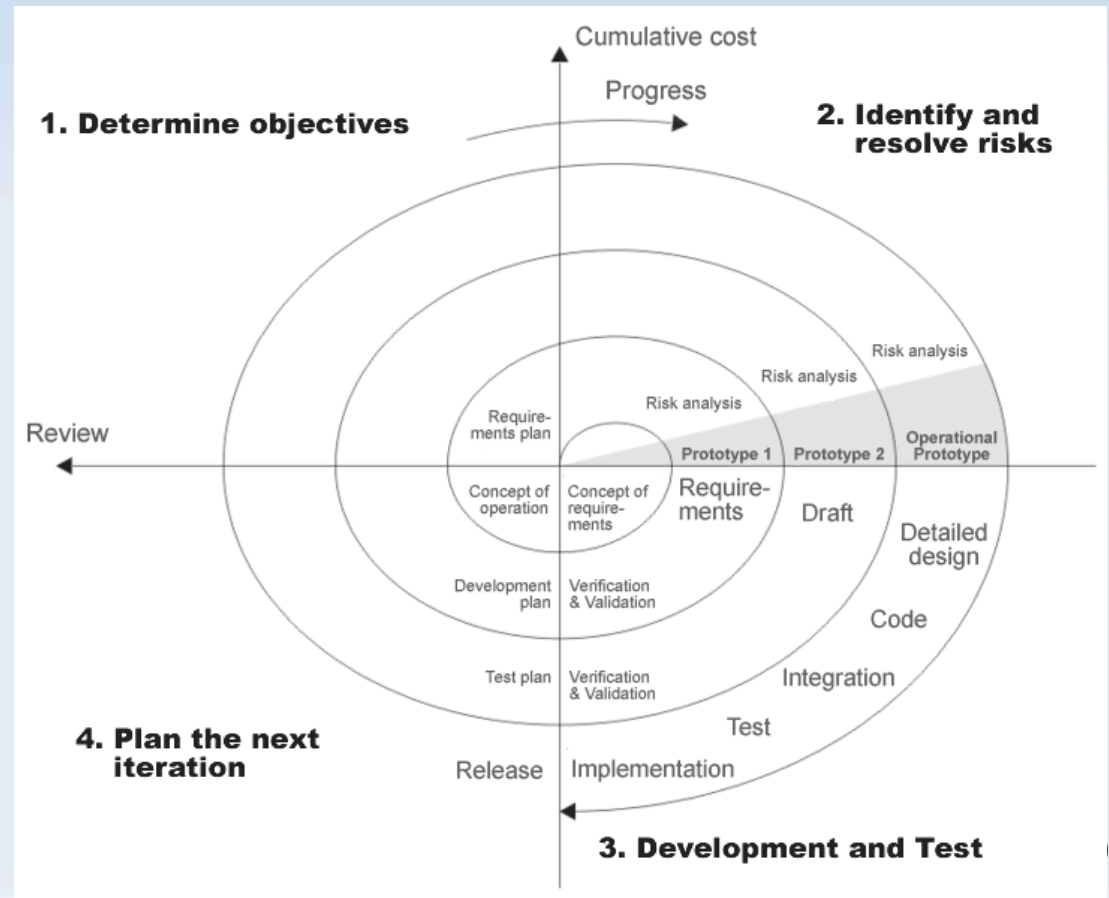
Modèles de projet SI

- Le modèle Itératif
 - Ce modèle permet de prendre en compte les évolutions de besoins et de contraintes
 - Il tends à impliquer le client tout le long du projet
 - Ce modèle intègre naturellement le modèle incrémental
 - Incrémental : Quoi, un noyau grossit au fur à mesure du développement
 - Itératif : Comment
- Avantages :
 - Offre une grande souplesse dans la gestion du projet
 - Intègre et gère la notion de risque tout le long du projet
- Inconvénients :
 - Ne donne pas facilement une vue précise de la planification du projet
 - Problème du bon découpage de chaque cycle et de leur point d'arrêt

Modèles de projet SI

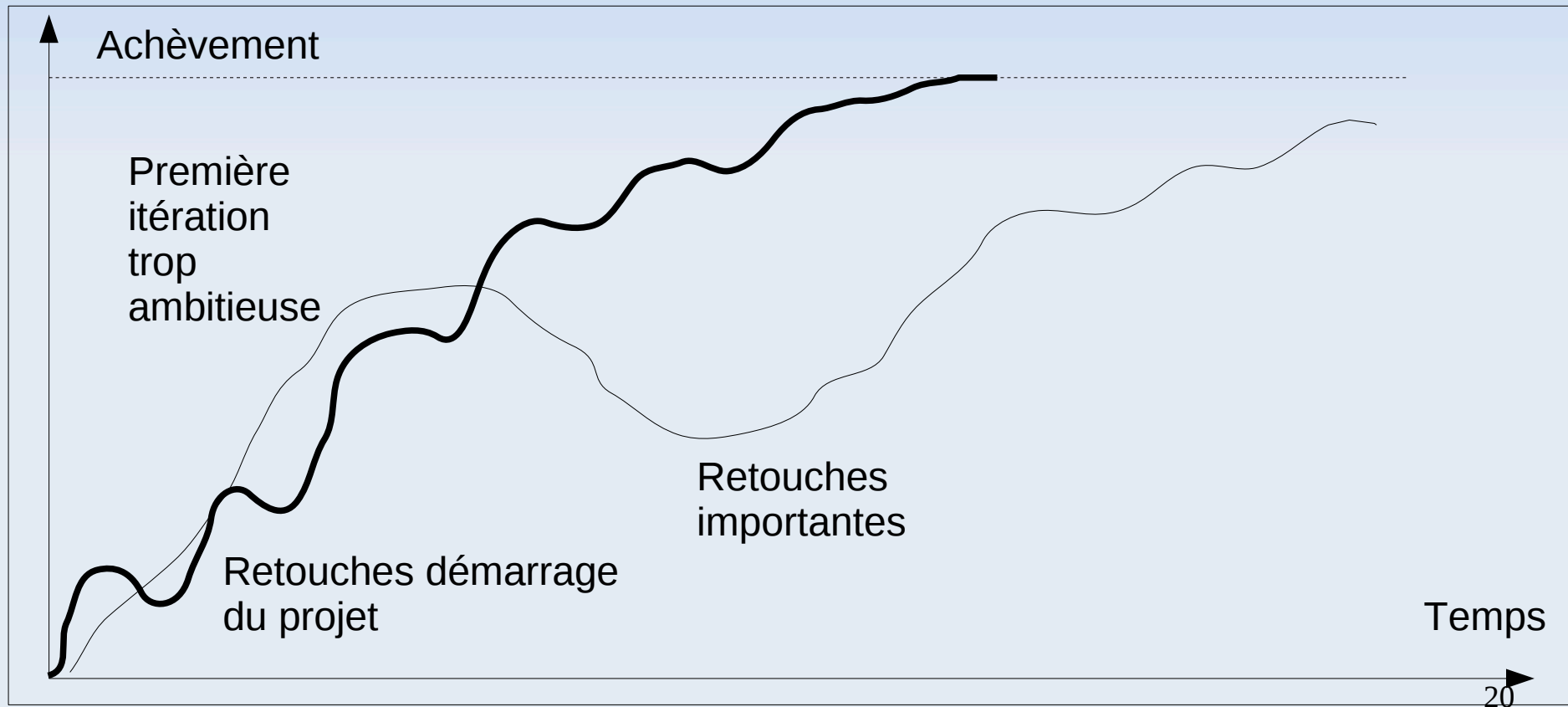
- Le modèle Itératif : la Spirale de Boehm (1988)

1. Besoins
2. Faisabilité - Analyse
3. Conception – Développement – Tests
4. Préparation de la nouvelle Itération



Modèles de projet SI

- Déroulement de projets itératifs
 - Ambition et optimisme de la première itération



Modèles de projet SI

- Comparaison de différents modèles de conception

Modèles	Avantages	Inconvénients	Type projet
Tunnel	Pas d'administration	Manque de visibilité totale	Petits projets, peu d'acteurs
Cascade (Royce, 1970)	Structure, jalons d'avancée	Validation tardive, retours correctifs.	Projets avec des besoins bien définis et évoluant peu.
Modèle en V	Structure, points de mesure, meilleure qualité	Validation tardive, retours correctifs. Les besoins sont figés.	Projets avec des besoins bien définis et évoluant peu.
Modèles incrémentaux	Meilleure appréhension des risques.	Risque de remettre en question le noyau.	Projets complexes de grande envergure
Modèles itératifs (dont la variante Spirale de Boehm 1988)	Flexibilité sur la gestion du risque. Implication du client. Recentre tôt les besoins sur les principales fonctions.	Visibilité générale moins poussée. Structure du résultat moins ferme.	Petit, moyen et grands projets suivant la variante
Modèles Itératifs-Incrémentaux	Production rapide d'un résultat tangible. Très bonne appréhension des risques.	Risque de remettre en question le noyau. Structure moins solide, sauf usage de CADRES	Projets dont les objectifs sont susceptibles d'évoluer. Favorise l'approche par composants (ou objets).

Méthodes

- Méthode : ligne directrice pour mener à bien un projet
 - Rationalise le déroulement d'un projet
 - Se base sur un ou plusieurs formalismes pour énoncer le problème et la solution
 - Pose des jalons pour mesurer l'avancement du projet
- Il n'existe pas de méthode miracle !

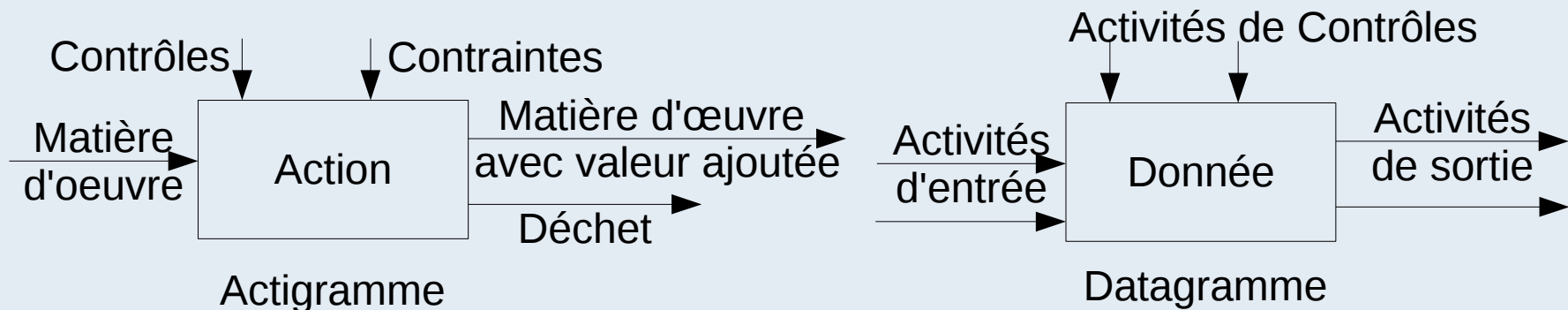
Méthodes

- Quelques méthodes

Méthode	Modélisation	Cycle de développement	Année	Contexte
<i>SADT</i>	Fonctionnelle et systémique	Cascade, V	1976	Contexte évoluant peu, bien cerné
<i>MERISE</i>	Systémique	Cascade, V	1978	Refonte et évolution de SI
<i>AGILE - RAD</i>	Systémique ou Objet	Itératif	1991	Grands projets
<i>AGILE - XP</i>	Objet, composants	Itératif, incrémental	1997	Petite équipe, contexte mouvant
<i>UP</i>	Objet	Itératif, incrémental	1999	Grands projets – variantes + 'agiles'

Méthodes

- SADT : Structured Analysis and Design Technic
 - Méthode d'approche descendante, par boîtes noires .
 - Chaque boîte exprime une fonction
 - Entrées : Information a traiter, Contraintes, Contrôles
 - Sorties : Information traitée, entrée d'autres boîtes



Méthodes

- SADT :
 - Décomposition en sous-fonctions
 - Définitions d'interfaces fonctionnelles
- Offre un cadre structurant pour décrire les systèmes
- Problèmes :
 - la modification d'une interface entraîne une remise en cause des fonctions appelantes
 - Si objectif évolue, revoir la décomposition fonctionnelle
 - Ne met pas en avant la dynamique.
- => Méthode pour un environnement évoluant peu, formellement analysable

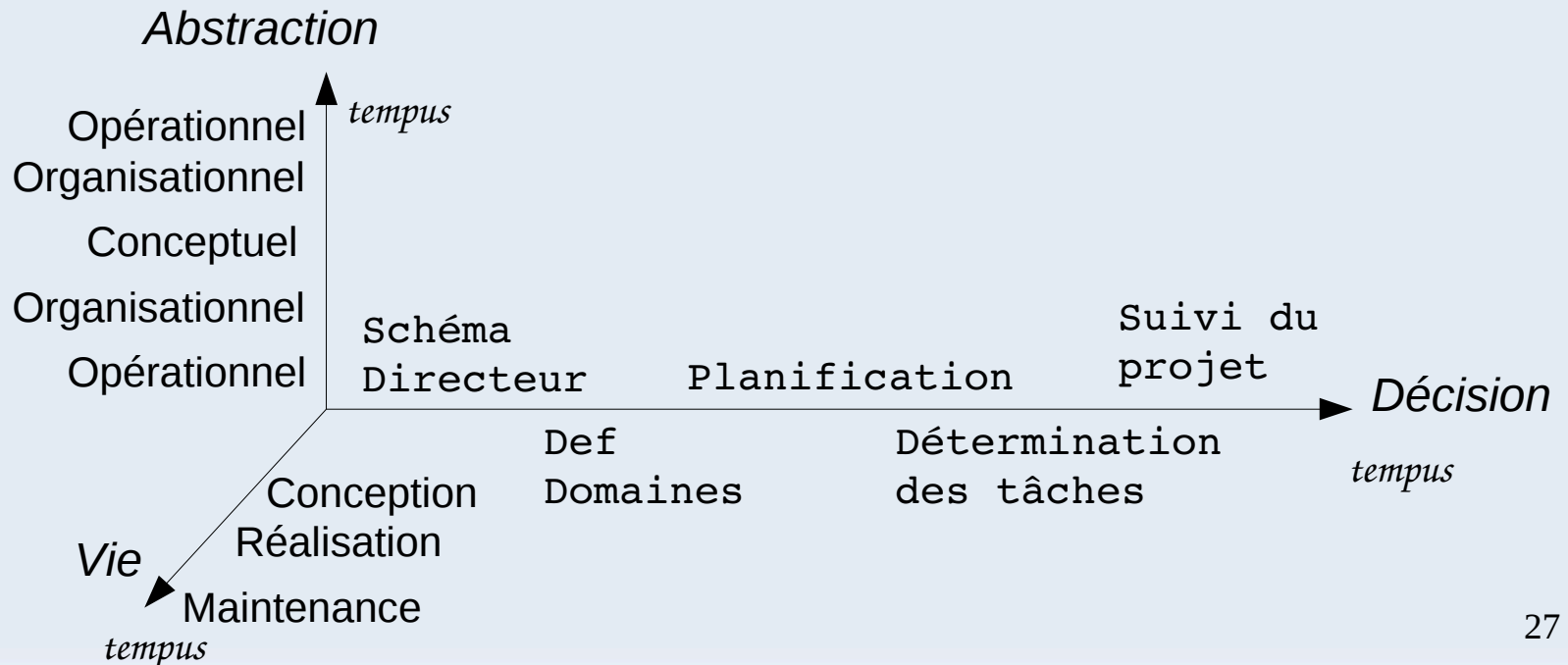
Méthodes

- **MERISE** : Méthode d'Étude et de Réalisation Informatique pour les Systèmes ? Ou fruit du Merisier, arbre qui se greffe facilement ?
 - Méthode française, basée sur le modèle entité-relation et les automates.
 - Vue systémique (partie opérante, d'administration et de décision)
 - Déroulement du projet selon 3 axes:
 - Cycle d'abstraction (Conceptuel, organisationnel, opérationnel)
 - Cycle de vie (conception, réalisation, maintenance)
 - Cycle de décision (Découpage du SI en domaines, orientation dans l'organisation du projet, planification, choix entre procédures manuelles et automatiques, détermination des postes de travail et des tâches, ...)

Méthodes

■ MERISE :

- L'équipe du projet mène le travail suivant les 3 axes en même temps.



Méthodes

- **MERISE :**

- Approche fonctionnelle : Notion de processus, se décomposant en opérations (au niveau conceptuel), puis en procédures fonctionnelles (MOT)
- Séparation des données des traitements, avec bien entendu des points de rencontre et de validation des 2 approches.
- Approche d'analyse :
 - Descendante
 - En 'cycle du Soleil' : on part de la réalité, on conceptualise, on réorganise, puis on intègre à nouveau les aspect réels

Méthodes

- **MERISE :**

- Principe de la méthode :
 - Analyse par les flux en descendant vers le détail
 - Déduction et description séparée des données et des traitements
 - Validation et fusion des 2 vues, avec les modèles externes, pour aboutir au cahier des charges
 - La développement est l'application systématique d'un cahier des charges pouvant être très détaillé.

Méthodes

- MERISE : méthode d'analyse par flux de données :
 - C'est une approche systémique
 - de l'analyse des flux, on déduit les données et les évènements déclencheurs des traitements
 - On sépare l'analyse des données de celles de traitements avant de rapprocher l'ensemble.
- Problèmes :
 - Difficulté de modéliser les traitements complexes
 - Problème de convergence Données et Traitements
 - Difficulté d'intégrer les évolutions des besoins

Méthodes

- **Méthodes AGILES :**
 - Terme désignant les méthodes qui ont pour caractéristiques communes :
 - Privilégier une équipe soudée et communicante sur la haute technicité
 - Mise en avant d'un logiciel fonctionnel même incomplet plutôt qu'une analyse claire et complète mais non réalisée
 - Mettre le client au centre du projet et privilégier sa collaboration plutôt que de figer son rôle derrière un contrat pour cibler au mieux ses besoins.
 - Avoir une approche permettant d'intégrer les changements d'objectifs plutôt que des procédures pour appliquer des planifications.

Méthodes

■ Méthodes AGILES, 12 principes :

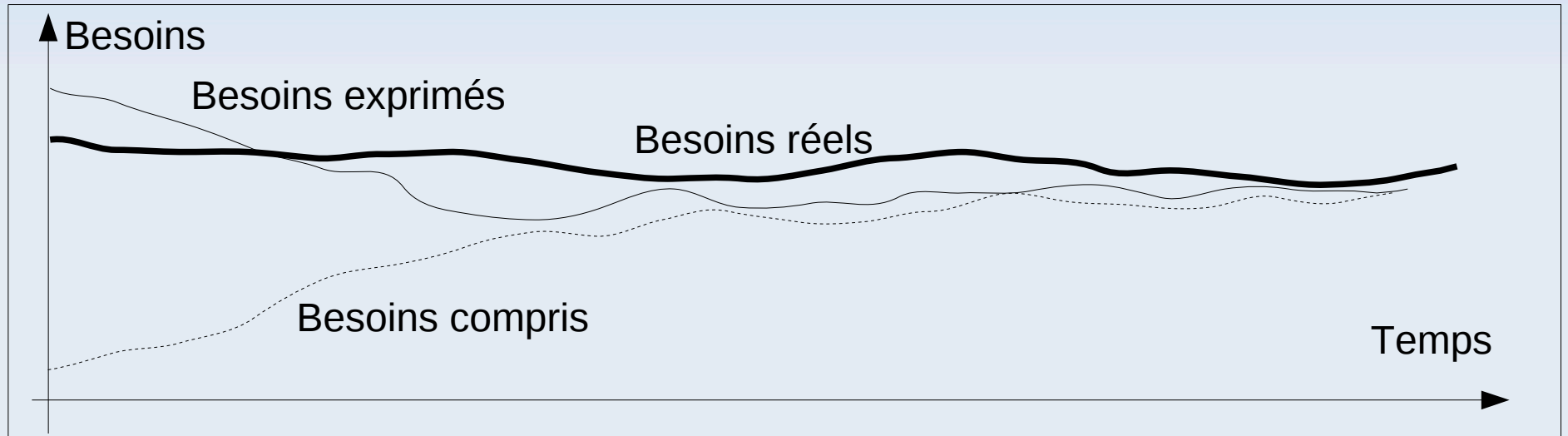
- Satisfaire le client
- Accepter les changements à l'initiative du client
- Livrer fréquemment une version du logiciel
- Les clients et intervenants doivent travailler ensemble sur tout le projet
- Pour obtenir un travail bien fait, soutenez leurs besoins, faites leur confiance
- Converser en tête à tête pour échanger efficacement de l'information
- Le premier indicateur d'avancement est le bon fonctionnement du logiciel
- Développement durable : Commanditaire doit pouvoir maintenir l'effort indéfiniment
- Favoriser l'excellence technique
- L'art de maximiser la quantité de travail à éliminer : simplifier !
- Favoriser l'auto-organisation
- Prendre régulièrement du recul afin d'améliorer le process

Méthodes

- **Méthodes AGILES :**
 - Avantages
 - Adaptabilité et réactivité
 - Réduction des risques, plus près de la réalité
 - Qualité, c'est à dire coller aux besoins
 - Rapidité et Efficacité
 - Inconvénients
 - Devoir gérer la communication, moteur de cette approche
 - Mauvaise visibilité à long terme qui peut dérouter
 - Problème d'une intégration continue

Méthodes

- Méthodes AGILES :



Méthodes

- **RAD, Rapide Application Development**

- **Cycle de vie :**

- **Initialisation** : réunir les acteurs, voir l'ensemble du travail, créer une dynamique, définir les limites (budget, temps). (6% du projet, 15 jours max)
- **Cadrage** : c'est l'expression des besoins (9% du projet, 30 jours max).
- **Design** : conception de la solution, mise ne place de la parallélisation, avec au final le premier prototype validé (23% du projet, 60 jours max).
- **Construction** : améliorations progressives du prototype par itérations qui offrent un produit fonctionnel (50% du projet, 120 jours max).
- **Mise en œuvre** : Intégration dans le contexte final, validation globale, bilan du cycle (12% du projet, 30 jours max)

Méthodes

- **RAD, Rapide Application Development**
 - **Caractéristiques :**
 - Outils : privilégier des AGL facilitant le prototypage, la génération du code ainsi que des batteries de tests.
 - Personnes : implication de l'équipe technique et du client
 - Management : plus en réseau que bureaucratique
 - Méthodologie : bien décrite et souple, formalisme au choix
 - **RAD 2 :**
 - Formalismes MERISE, Flux DFD, UML
 - Formalisation des entretiens
 - Validation permanente

Méthodes

- XP eXtreme Programming:
 - 4 bases :
 - **Communication** : Client, communication utilisant des formalismes simples et clairs. Intra-équipe : documentation avec le code ! (Mais aussi documents annexes, comme schéma des classes ou relationnel)
 - **Retours** : Client, retours lors de tests unitaires. Équipe : retours sur le codage des uns et des autres. Programmation en binôme, avec changement régulier de partenaire.
 - **Simplicité** : supprimer le superflus pour conserver la productivité. On préfère du code simple qui répond à une problématique simple, et non du code structuré pour évoluer vers des fonctions potentielles mais non validées.
 - **Courage** : Savoir mettre une partie de son code à la poubelle. Ne pas avoir peur de ses propres faiblesses.

Méthodes

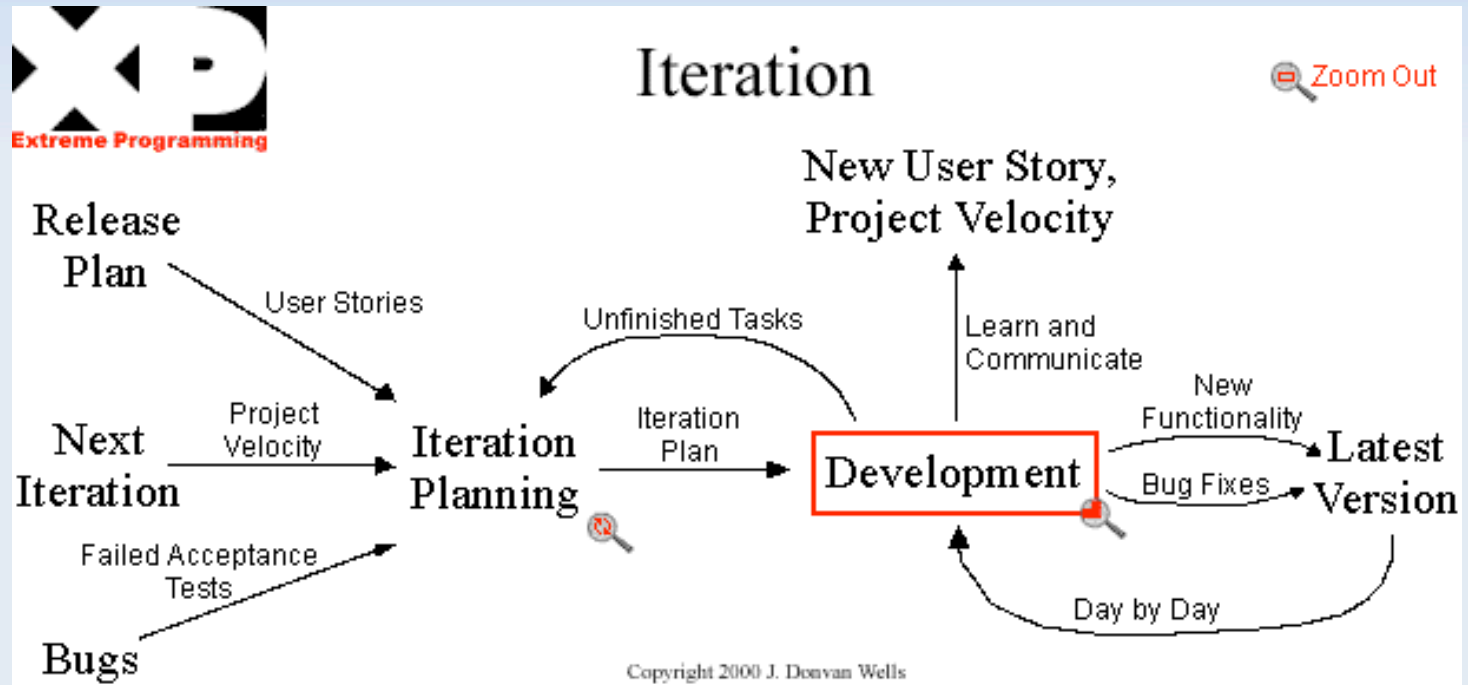
- XP eXtreme Programming:
 - Caractéristiques :
 - Fait une large place aux aspects techniques
 - Fort esprit d'équipe, programmation en duo. La confiance en avant.
 - Simple à mettre en place, pour de petites équipes (<10 personnes)
 - Minimisation des procédures de gestion (sans les éliminer)
 - Méthode adaptative plus que prédictive
 - Forte implication du client qui doit être souvent sur place
 - Inconvénients :
 - étapes amont non prises en compte
 - Minimise l'analyse, donc risque d'incohérence
 - Peu structurant
 - Savoir mobiliser l'équipe et la souder

Méthodes

- **XP eXtreme Programming, cycle de vie :**
 - **Spécifications** : Subdiviser en problèmes plus petits. Les prioriser . C'est le client qui rédige les spécifications (% du projet, 1 jour)
 - **Écriture des tests** : tests fonctionnels définis avec le client, sert à valider le produit. Tests unitaires par les développeurs.
 - **Programmation** : classes simples. Besoins de reculs réguliers pour avoir une vision d'ensemble. Programmation en binôme
 - **Documentation** : celle des utilisateurs qui doit être complète, celle des programmeurs qui doit être peu chargée pour un usage d'évolution
 - **Livraison** : réception par le client du logiciel opérationnel. Le code doit être propre pour faciliter les évolutions futures.

Méthodes

- XP eXtreme Programming: Cycle de Vie



Méthodes

- Unified Process
 - Années 1995 - 1999.
 - Synthèse de diverses méthodes objets (OMT, ...)
 - Normalisation de bonnes pratiques, basées sur l'architecture, générale au début, s'affinant ensuite.
 - Dérivé du modèle en spirale, modèle itératif et incrémental
 - Commercialisé par Rational sous le terme de RUP, avec des outils comme une plateforme WEB commune.
 - Il existe d'autres variantes : EUP, XUP, AUP

Méthodes

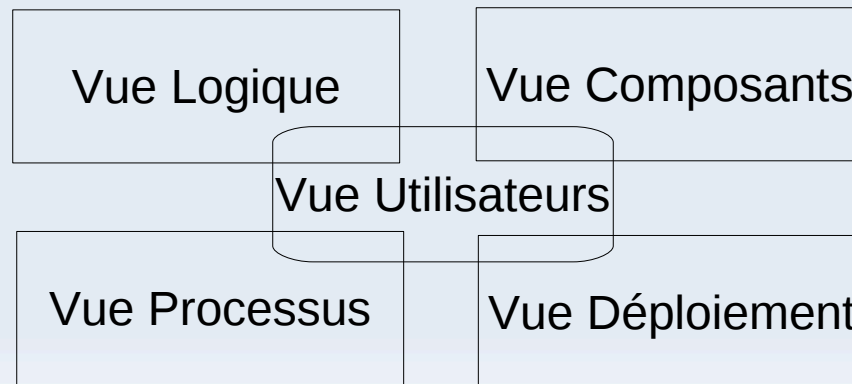
- Unified Process, Caractéristiques générales :
 - Itérations de courte durée (15 jours à 2 mois suivant les projets)
 - Fonctionnalités principales développées dès le départ
 - Prise en compte des risques, re-cadrages dès le début
 - Expression des besoins par prototypes
 - Implication du client : versions exécutables
 - Privilégie une approche par objets
 - Mets en avant l'architecture du système
 - Couvre tout un projet dont l'analyse et le développement
 - Intermédiaire entre les méthodes Prédictives et Adaptatives, pouvant être qualifié d'Agile ou non suivant son application.

Méthodes

- Unified Process, Caractéristiques générales :
 - Ce sont les cas d'utilisation qui dirigent le déroulement. Ils décrivent les objectifs, QUOI FAIRE
 - Cependant, le projet repose sur l'architecture dès le départ. Grossière au début, elle s'affine à chaque itération. Elle décrit le COMMENT FAIRE.

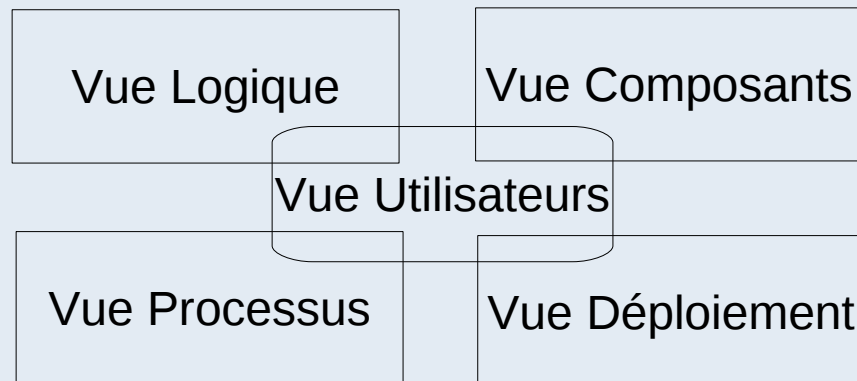
Méthodes

- Unified Process, Caractéristiques générales :
 - Cette méthode se base sur la vue 4+1 (Ph. Krutchen)
 - Utilisateur : guide l'analyse des besoins et des solutions adoptées. Elle cimente les autres vues
 - Logique : centre la représentation du SI sur les données
 - Processus : montre le SI sous l'aspect fonctionnel
 - Composants : montre la traduction du SI en modules de programme
 - Déploiement : Présente le SI dans son contexte physique



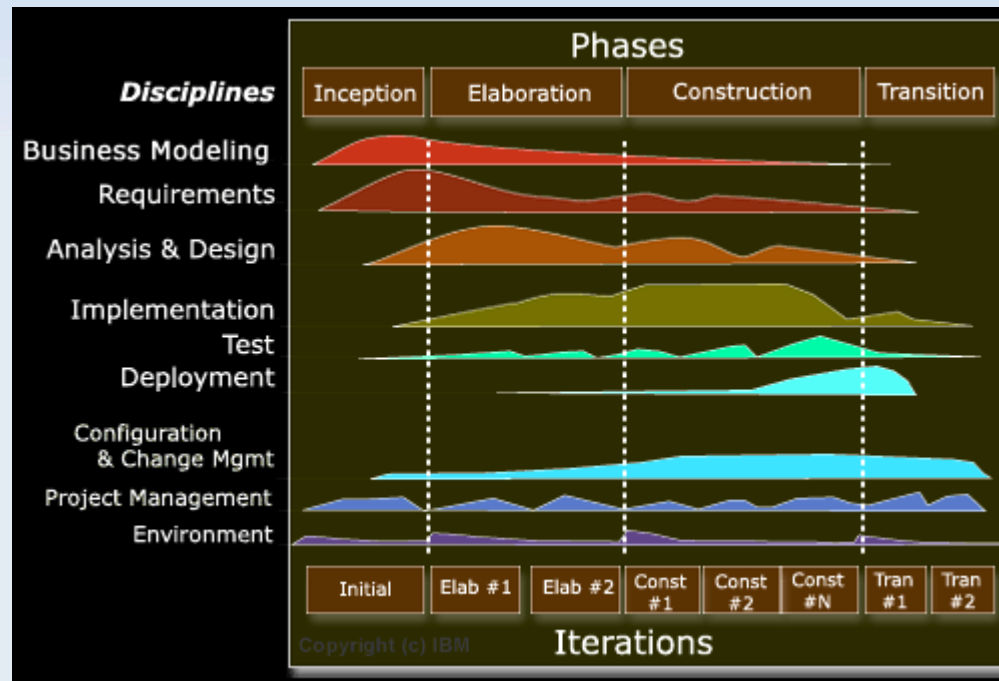
Méthodes

- Unified Process, Caractéristiques générales :
 - Utilisateur : Cas d'utilisations, scénarios (diag. cas d'utilisation, séquence, activité)
 - Logique : architecture des classes, dynamique (diag objets, classes, séquence, collaboration)
 - Processus : aspect dynamique, concurrence, distribution, tolérance aux pannes (diag. états transition, activité)
 - Composants : montre la traduction du SI en modules de programme (diag composants)
 - Déploiement : projection des composants sur le matériel (diag. déploiement)



Méthodes

- Unified Process, le déroulement général :



Méthodes

- Unified Process, les phases :
 - *Elles représentent les grandes étapes du projet et peuvent se subdiviser chacune en une ou plusieurs itérations.*
 - **CRÉATION** : opportunité-faisabilité, cahier des charges, architectures candidates.
Risques majeurs (faisabilité, financements, ...)
Évènement final : on poursuit ou non le projet. Maquette
 - **ÉLABORATION** : Planification, architecture de référence.
Risques impactant le fonctionnement du projet (planning, ...)
Évènement final : architecture du système, prototype de l'architecture ou du noyau

Méthodes

- Unified Process, les phases :
 - **CONSTRUCTION** : programmation incrémentale.
Risques locaux et spécifiques (développements, ...)
Évènement final : version bêta utilisable
 - **TRANSITION** : installation, préparation de la maintenance
Risques locaux et spécifiques (installations, paramétrages)
Évènement final : première version finale utilisée
- *La maintenance n'est pas incluse dans UP, c'est à part.*

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - *Elles se retrouvent dans un projet traditionnel en cascade. Chacune est appliquée dans l'ordre pour chaque itération même la première, mais avec plus ou moins d'importance.*
 - INGÉNÉRIE
 - Expression des besoins
 - Analyse
 - Conception
 - Implémentation
 - Tests
 - GESTION ET SUPPORT
 - Gestion de projet
 - Configuration
 - Environnement

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Expressions des besoins :
 - Besoins définis par l'utilisateur
 - Besoins dus aux contraintes du projet
 - Classer les besoins par importance (Gestion du risque)
 - **UML** : Cas d'usage, Séquence, États Transition, Activités
 - *Définitions des limites de l'itération*
 - *Cas d'utilisation, acteurs, scénarios principaux, maquette IHM, dictionnaire*

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Analyse :
 - Précision des besoins et du dictionnaire
 - Découverte de la structure du système:
Cas d'utilisation et de scénarios => classes et collaboration.
 - Classement avec l'aide de stéréotypes généraux :
Frontière (boundary), Contrôle, Entité
 - **UML** : Objets, Classes, Collaboration, Séquence, États Transition, Activités
 - *Définitions des éléments candidats au système*

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Conception :
 - Construction du système à partir des éléments d'analyse
 - Définition de sous-systèmes par rapport à des critères de cohérences
 - Définition ou enrichissement de composants réutilisables
 - **UML** : Objets, Classes, Collaboration, Séquence, États Transition, Activités
 - *Donne l'architecture du système à implémenter*

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Implémentation :
 - Traduit les composants dans le système sous-jacent
 - **UML** : Composants, Séquence, États Transition , Déploiement
 - *Produit un ensemble de programmes*

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Test :
 - Procédures de test :
 - Unitaire
 - Intégration logique, Intégration système
 - Recette (tests validés par le client)
 - Environnement de test, automatiser !
 - Vérifier si les besoins sont satisfait
 - Détecter les faiblesses de l'architecture
 - **UML** : Cas d'usage, Collaboration, Séquence, Activités
 - *les composants implémentés validés dans le système* 54

Méthodes

- Unified Process, les activités ou 'disciplines' :
 - Gestion de projet :
 - Planification, diagrammes de GANTT, ...
 - *Définition et attribution des tâches*
 - Gestion de l'environnement :
 - *Définition et configuration des outils. Formations de l'équipe*
 - Gestion des versions :
 - *synchronisation des modifications*
 - *arrêt de chaque version*

Méthodes

- Unified Process, taxinomie des classes :
 - Classes Entités :
 - Ensemble des classes qui modélisent les données métiers
 - Ces classes sont persistantes
 - Classes d'Interfaces :
 - Délimitent le système avec l'extérieur. (IHM, Impression, Capteurs, ...)
 - Classes Processus :
 - Organise les traitements et les contrôles
 - En pratique : un Use Cas => une classe processus

Méthodes

- Unified Process, Classes Entités :
 - Entités de référence :
 - Modélise les informations de base : Pays, Entreprise, Client, Adresse, Fournisseur, systèmes externes ...
 - Entités de gestion :
 - Modélise les informations de gestion, et repose sur les entités de référence : contrat, ordre, opération, dossier ou document, ...
 - Entités de reporting :
 - Modélise des informations souvent calculées utilisées pour le pilotage : état d'un stock, état d'une commande, statistiques,

Méthodes

- Unified Process, Classes d'Interfaces :
 - Vérifie et formate les entrées d'information dans le système ou présente les informations en respectant les normes des systèmes externes.
 - À la réception d'un message, enclenche un Cas d'utilisation.
 - Capteurs :
 - Émetteurs :
 - IHM :
 - Homme = émetteur et récepteur très complexe
 - Importance de l'ergonomie
 - Utilise souvent la notion d'interface ou de classe virtuelle dans certains langages

Méthodes

- Unified Process, Classes Processus :
 - Suite coordonnées d'appels de méthodes. Représente un service du système.
 - Déclenché par un événement : interne, externe ou temporel
 - Processus Métier : evts externes ou temporels. Entités de références.
 - Processus Support : soutient l'accomplissement des processus métiers (transformation, communication, ...). Tout evts. Entités de gestion
 - Processus Pilotage : evts internes ou temporel. Entités de reporting

Méthodes

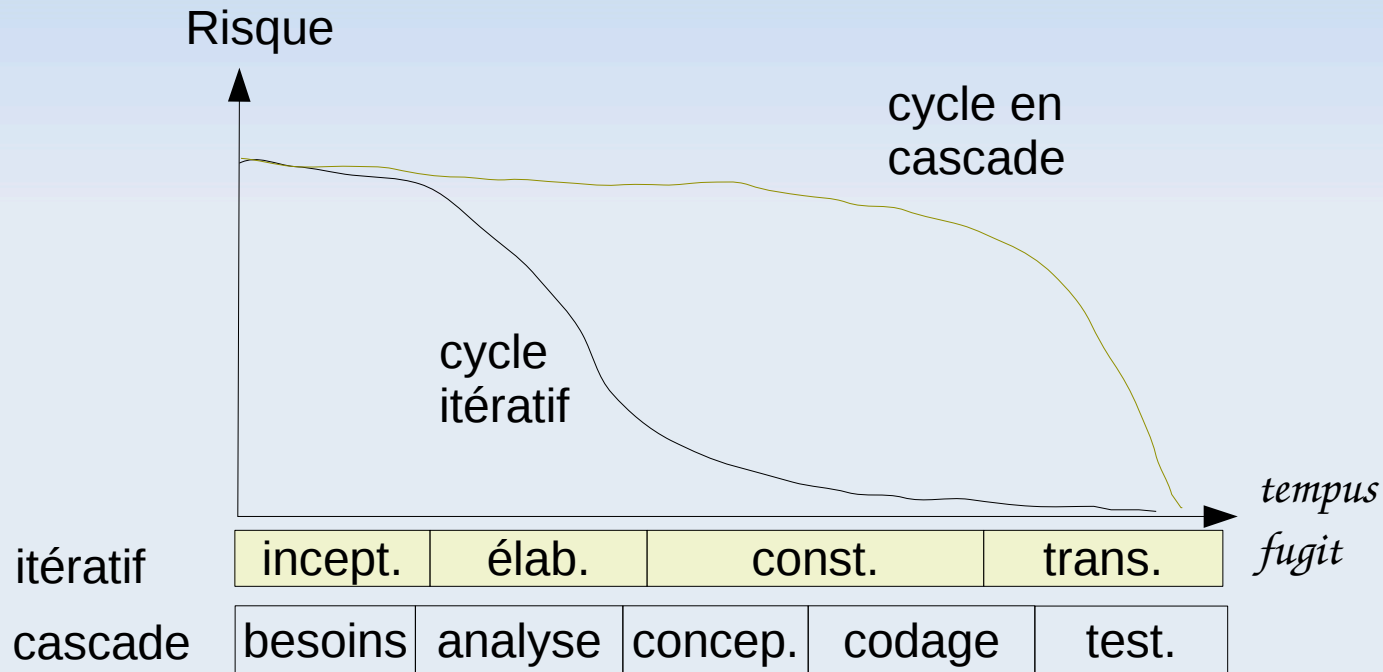
- Unified Process, Classes Processus :
 - Suite coordonnées d'appels de méthodes. Représente un service du système.
 - Déclenché par un événement : interne, externe ou temporel
 - Processus Métier : evts externes ou temporels. Entités de références.
 - Processus Support : soutient l'accomplissement des processus métiers (transformation, communication, ...). Tout evts. Entités de gestion
 - Processus Pilotage : evts internes ou temporel. Entités de reporting

Méthodes

- Unified Process, Les risques :
 - Premier risque : ne pas voir ou vouloir voir le danger
 - Au final : ne pas répondre aux besoins
 - Risques commerciaux
 - Concurrence ? Occuper le terrain avec solution minimale ?
 - Risques financiers
 - Capacités de financement non surpassées
 - Risques techniques
 - Choix technique éprouvée ?
 - Risques de développement
 - équipe productive immédiatement, formation nécessaire ?

Méthodes

- Unified Process, Réduction des risques :



Méthodes

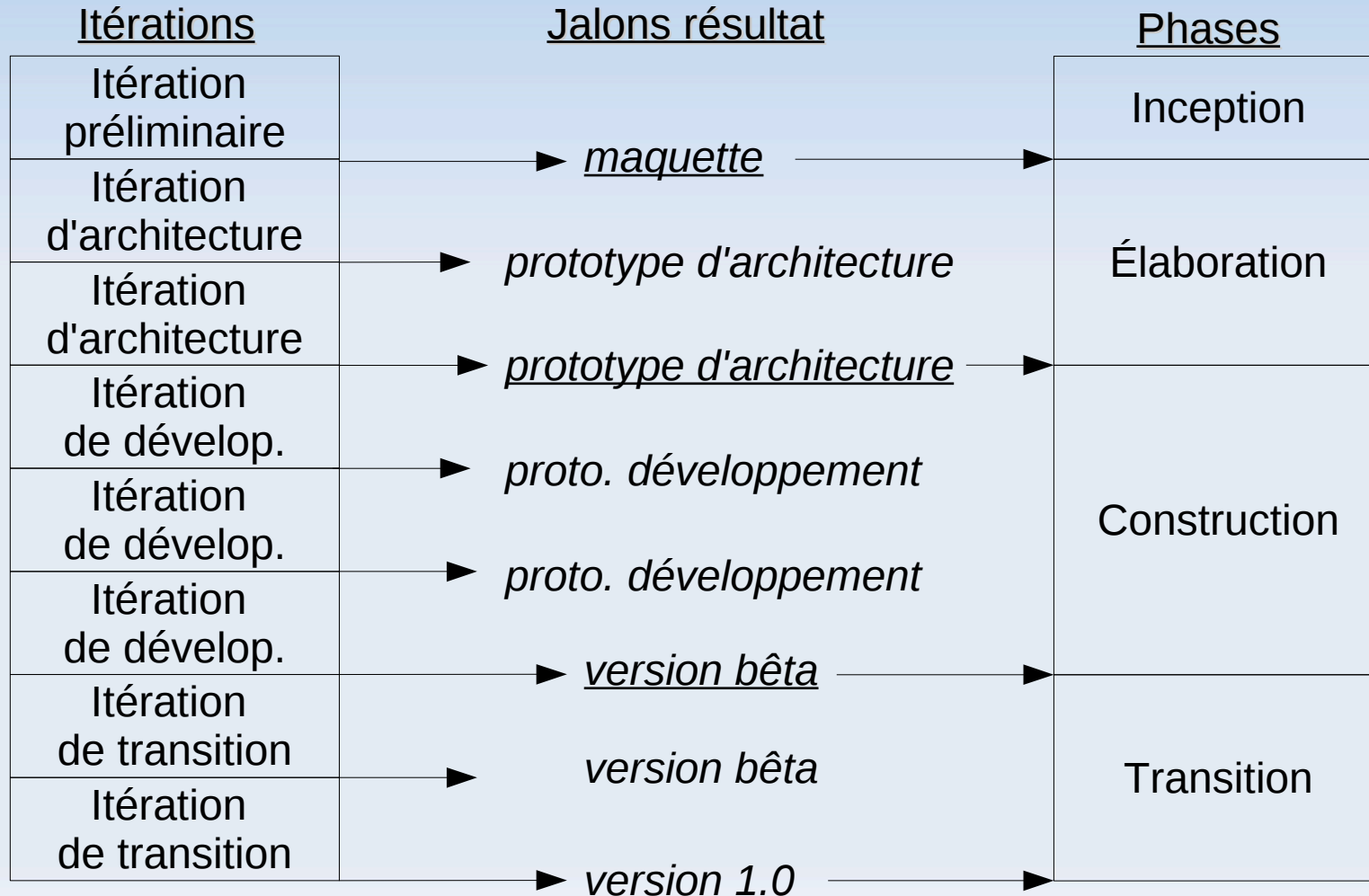
- Unified Process, Réduction des risques :
 - Étude d'opportunité : limitation du risque du projet maquette. Maquette retouchable de 50 à 100 %.
 - Élaboration : réduction des risques d'incompréhension avec les usagers. Appréhension des risques d'architecture. Prototype d'architecture retouchable à 25% environ.
 - Construction : intégration progressive des besoins, du plus au moins prioritaires. Version bêta retouchable à moins de 10%.
 - Transition : risque de prise en main réduit par un déploiement progressif et par l'implication de l'utilisateur dans les phases précédentes. Versions retouchables de 4 à 1%

Méthodes

- Unified Process, Réduction des risques :
 - Réduction possible en ciblant directement les besoins
 - Illustrer concrètement les besoins par :
 - Maquette
 - Prototypes
 - Régulièrement présenté au client
 - Résultat tangible = mesure facilité de l'avancée du projet, = plus forte motivation de l'équipe

Méthodes

- Unified Process : résumé de déroulement:



Méthodes

- Comparaison MERISE et UP :

	MERISE	UP
Modèle de base	Systemique	Objet, plus proche de la programmation
Approche générale	Montante (existant) puis Descendante	Descendante
Modélisation Statique	Bien déterminée (MCD, MLD). Proche d'une représentation mémoire (base de données). Les informations calculées sont distinguées à part.	Se confond assez vite avec les méthodes. Un attribut peut être calculé.
Modélisation Dynamique	Bien déterminée (MCT, MOT). Dérivée de l'analyse des flux et associée à un modèle à évènement.	Se confond avec les données. UP possède cependant une approche fonctionnelle pour décrire certaines méthodes ou pour analyser les besoins (Use Cas).
Acteurs	Acteurs externes et internes au SI	Acteur = extérieur au SI
Contexte général	Conception de SI d'entreprise au sens large	Conception de logiciel

Méthodes

■ Comparaison MERISE et UP :

MERISE	UP
Cycle de développement linéaire	Cycle de développement itérative
Cycle de vie (Conception, Réalisation, Maintenance)	Phases (Inception, Elaboration, Conception, Transition)
Cycle d'abstraction (Conceptuel, organisationnel, opérationnel).	Activités d'ingénierie (Analyse des besoins, Analyse, Conception, Implémentation, Test)
Cycle de décision (schéma directeur, découpage en domaines, Planification du projet, détermination des postes de travail, suivi de projet)	Activités de gestion et de support

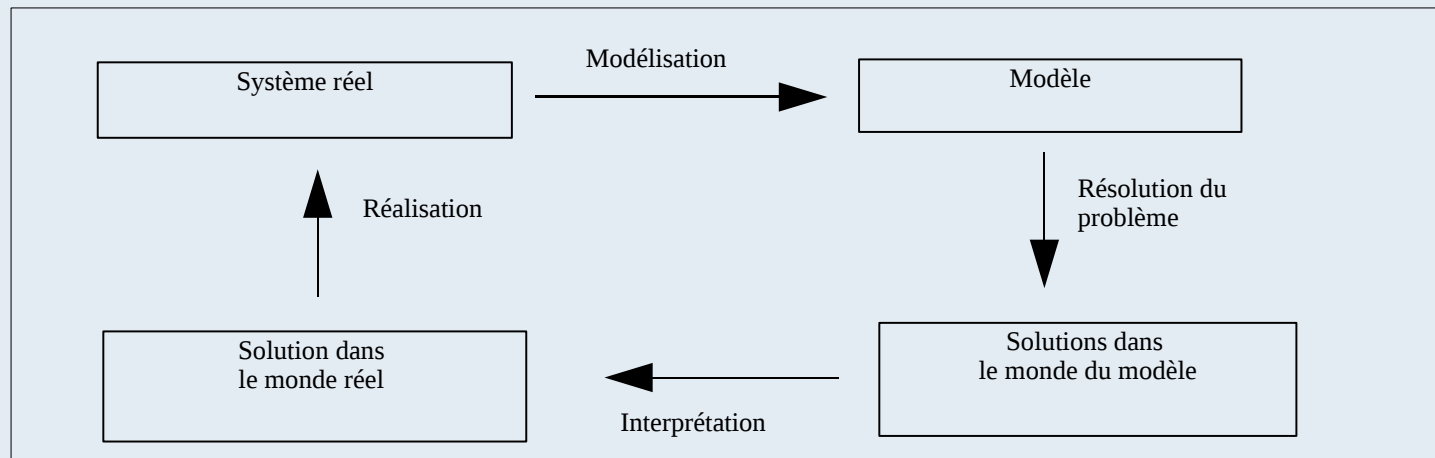
- On peut tenter une comparaison globale entre les 2 méthodes.
- Toutes 2 partent du global et finissent sur le détail.
 - Les outils et modèles en usage suivent cette ligne.
- Mais certaines notions se retrouvent réparties dans d'autres de la méthode comparée et inversement (Décision de MERISE en grande partie gérée par la phase d'inception d'UP)
- **Les 2 méthodes : cadre à suivre et jalonnement**

Sur l'analyse et la modélisation

- **L'analyse** : Décomposition d'un système pour mieux l'appréhender et aboutir à une modélisation
- **Modèle** : représentation simplifiée d'un système, résultat d'une analyse
- **Méthode d'analyse** : aide pour modéliser un système
- **Systeme** :
 - Vue statique : éléments, relations, limite, réservoir «d'énergie»
 - Vue fonctionnelle : Flux, partie de décision, de gestion et d'exécution, rétroactions

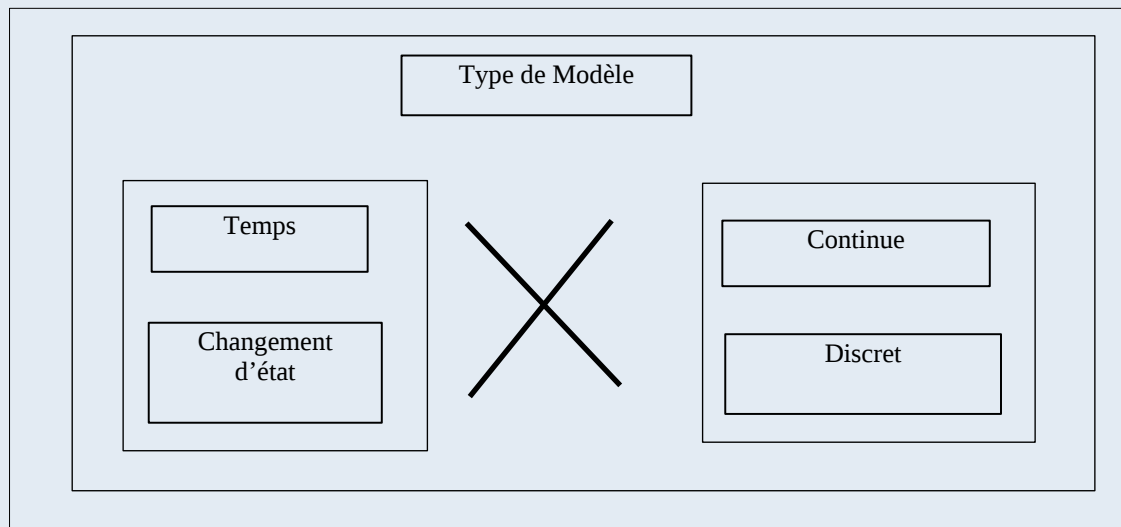
Sur l'analyse et la modélisation

- Objectifs d'un modèle :
 - 1) On observe le système réel au travers un modèle;
 - 2) On résout le problème avec l'aide du modèle;
 - 3) On interprète le résultat dans le modèle;
 - 4) On applique le résultat dans le réel;



Sur l'analyse et la modélisation

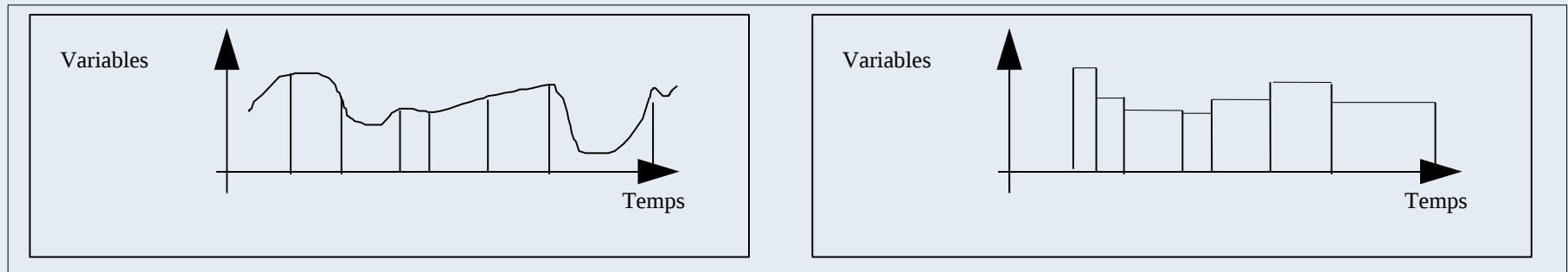
- Catégories de modèles
 - Distinction entre le temps et les états
 - Distinction entre valeur continue et discrète



Sur l'analyse et la modélisation

- Catégories de modèles

Types de modèles	Temps	Variables
équations différentielles	continu	continues
équations aux différences	discret	continues
modèle à éléments discrets/événements	continu	discrètes
Automates	discret	discrètes

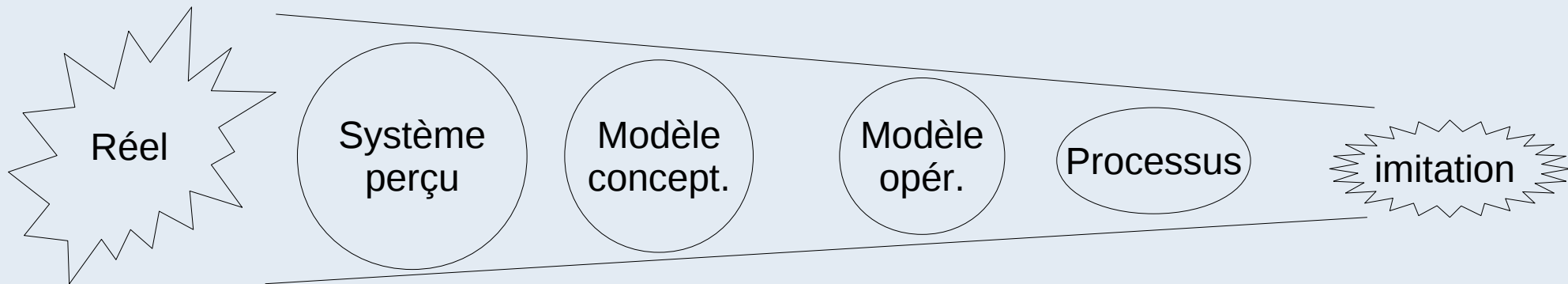
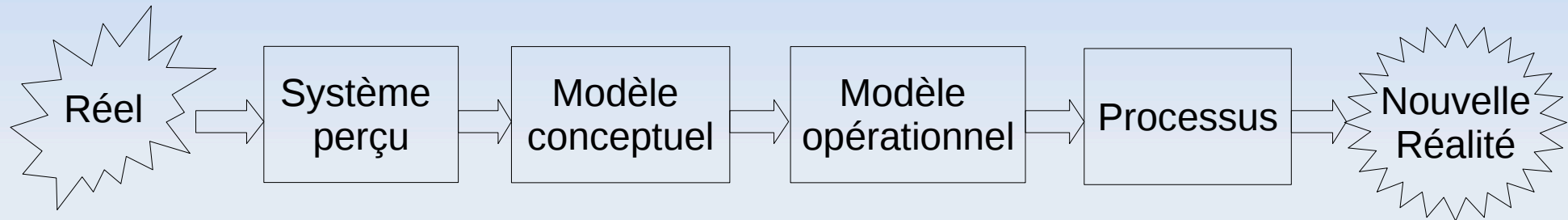


Sur l'analyse et la modélisation

- Catégories de modèles
 - Modèle bien-décrit :
 - La connaissance des valeurs à l'état t permet de connaître celles à l'instant $> t$
 - Modèle non déterministe : ne détermine pas un seul ensemble de valeurs au futur, mais plusieurs. Cela indique que tous les cas n'ont pas été pris en compte
- Un microprocesseur possède un fonctionnement d'automate. Un programme est l'application d'un 'modèle' de processeur bien-décrit, à variables et temps discrets.

Sur l'analyse et la modélisation

- Modélisation :



Cône de fidélité du
Réel modélisé

Sur l'analyse et la modélisation

- Quelques modèles en informatique :
 - Les systèmes formels (bool, ...)
 - Les réseaux sémantiques
 - Le modèle relationnel
 - Les objets
 - Les agents

Sur l'analyse et la modélisation

- Quelques principes pratiques :
 - Séparer le temps du reste :
 - Temps qui sert à la dynamique (ordre de traitement)
 - Temps qui sert de donnée statique (historique)
 - Définition des notions :
 - Expertise du domaine
 - La synonymie et la polysémie
 - Classement des notions :
 - Volatilité : constante, variable, fonction
 - Modularité : éviter les interactions
 - Abstraction : perdre du détail pour l'essentiel, ou l'inverse, spécifier
 - Les méta-informations :
 - Utilisées souvent pour gérer le système

Sur l'analyse et la modélisation

- Analyse par objets :
 - Approche plus naturelle, issue des réseaux sémantiques et de l'évolution de la programmation.
 - Encapsule la complexité par une couche fonctionnelle
 - Organisations structurelles (classe, héritage, ..)
 - Liaisons dynamiques entre les objets (messages)
- Problèmes :
 - Favorise la redondance d'informations et de traitements par sa nature réutilisable
- Méthodes d'analyse très utilisées : approche naturelle couplée d'outils de génération de code.

Sur l'analyse et la modélisation

- Le modèle par objets
 - Objet : entité avec un état et un comportement
 - Complexité encapsulée
 - identité unique dans le système
 - Communication par messages, synchrones ou non
 - Les classes : définition abstraite d'objets de même type
 - Encapsulation généralisée
 - Relations : associations,
 - composition
 - héritage, polymorphisme

Sur l'analyse et la modélisation

- Volatilité des notions objets
 - Classe
 - Attributs
 - Héritage
 - Composition (liens d'état)
 - Action

Sur l'analyse et la modélisation

- Analyse par les textes, quelques pistes :
 - Nom :
 - Propre : Objet, Commun : Objet ou classe
 - Nom concret (déterminé par nos sens) => Objet ou classe
 - Nom abstrait (volonté, réussite, effort ...) => attribut, classe
 - Nom de groupe : un troupeau de vaches => ensemble
 - RQ : cherchez un identifiant 'naturel' pour chaque classe
 - Adjectif :
 - Attribut

Sur l'analyse et la modélisation

- Analyse par les textes, quelques pistes :
 - Verbe : Verbe d'état : lien statique entre noms, ou lien d'attribut
 - être, devenir, paraître, sembler, demeurer, rester, avoir l'air, passer pour, se compose de, se trouver, posséder
 - Est une sorte de ... => Abstraction, relation d'héritage
 - Est une ... => Réalisation, lien d'instance
 - Verbe d'action (voix active ou passive): lien dynamique, déduire des fonctions

UML

- Généralités

- C'est un langage de modélisation d'objets, formalisé
 - Sous forme graphique
 - Décrivant lui-même son méta-modèle
 - Complétés de langages textuels (XMI et OCL)
- C'est une norme industrielle (OMG, 1997)



UML

- Généralités :
 - Il est conçu comme un outil de communication
 - Entre les différentes parties du projet par des représentations simples, informaticiens et gens du métier
 - C'est un cadre sur lequel reposent plusieurs méthodes d'analyse.
 - Son aspect semi-formel, permet une description très détaillée ainsi que la génération de code dans un langage de programmation cible, avec l'aide de tables de traduction.
 - Mais ce n'est pas un langage de programmation
 - Et encore moins une méthode !

UML

- Les principaux diagrammes UML 2
 - Cas d'utilisation
 - Séquences
 - Classes
 - Objets
 - Collaboration
 - Activités
 - États-transitions
 - Composants
 - Déploiement

Les Design Patterns

- Patron de conception
 - Schéma général d'une classe ou d'une architecture appliqué à un domaine
 - Abstraction de solutions jugées bonnes par l'expérience. Capitalise un savoir-faire reconnu.
 - Indépendant du langage de programmation
- Avantages :
 - Accélère la réalisation d'un logiciel en appliquant et adaptant un patron
 - Augmente la qualité en évitant d'inventer une solution générale non éprouvée
- Ex : Modèle Vue Contrôleur

Les Design Patterns

- Patron de conception
 - Leur description n'est pas formalisée par un langage de référence. Il s'agit souvent de classes abstraites à implémenter
 - Il existe de nombreux patterns :
 - Véritables catalogues
 - Mais on retrouve plusieurs patterns proches avec différences : lequel choisir ?
 - Des patterns peuvent faire appels à d'autres
- Description : Nom, Problème visé, Solution, Conséquences

Les Design Patterns

- Patron de conception pour les classes
 - Patron de création :
 - Rassemble les pratiques pour créer un objet, le référencer, dénombrer les instances, ...
 - Exemple : Fabrique (factory), Singleton (vérifie l'unicité de l'instance)
 - Patron de structures : décrit l'assemblage d'objets.
 - Adapter : rendre un objet conforme à un autre
 - Bridge : pour lier une abstraction à une implémentation distinct
 - Patron de comportement :
 - L'itérateur
 - Template