Module M3301
Partie 2 - Méthodologie de construction du logiciel
Modélisation et construction des logiciels complexes

J. Christian Attiogbé

Université de Nantes

Plan du cours

Fondamentaux en ingénierie

Définitions : Exigences, besoins, spécifications, cahier de charges, ...

Besoins, Needs, *Requirement* - côté client

Requirements are the users' description of what the finished product should do.

Spécification des besoins - côté technique

A specification is an explicit set of requirements to be satisfied by the software.
A specification is the technical description of the software, covering the requirements and much more (cost, technicalities, problems, ...).

Fondamentaux en ingénierie

Notions fondamentales

Cahier de charges, Spécifications fonctionnelles,
Architecture des applications logicielles, Cycles de vie

Analyse des besoins

Exigences fonctionnelles
Elicitation, analyse,
spécification, vérification,
validation des besoins d'un (projet) logiciel.

Ingénierie système

Décomposition fonctionnelle,
Décomposition modulaire,
Construction des composants,
Assemblage des composants

Conception et réalisation de produits industriels

En sciences de l'ingénieur la démarche scientifique pour réaliser un produit industriel passe par deux principales phases : la conception puis la fabrication du produit.
Les étapes du processus de réalisation sont :

- l'élaboration du cahier des charges fonctionnel,
- l'analyse,
- la conception,
- la réalisation de prototypes,
- la mise au point produit-process,
- l'industrialisation.

Ce sont les mêmes étapes qui sont suivies pour la réalisation de logiciels ou de systèmes contenant du logiciel.

Etapes de conception et de réalisation de produits

- La durée des étapes est plus ou moins longue, selon les contextes, les méthodes, les outils.
- L'analyse des systèmes consiste à décomposer le/s produits en composants (sous-systèmes)
- L'étape de mise au point se fait généralement par simulation et essais.
  Certaines étapes/phases, comme la simulation, sont elles même assistées par ordinateur.
- Lorsqu'on adopte une démarche rigoureuse, bâtie sur des méthodes formelles de modélisation et d'analyse, on allonge le temps de l'analyse conception et on réduit considérablement le temps des mises au point. On idéalise la construction correcte immédiate.

Conception de produit (ou analyse fonctionnelle)

Objectif

L'objectif de la conception est de prévoir ce que sera le comportement du produit lors de son utilisation en conditions réelles.

Réponse d'un produit

Le produit interagit avec son environnement, ce qui entraîne une modification de certaines grandeurs physiques pouvant caractériser soit l'environnement, soit le produit lui-même. Une telle modification est appelée réponse du produit.

Conception

La tâche du concepteur est de déterminer les réponses, ou du moins celles qui font partie des objectifs de l'étude, afin de s'assurer que les performances réalisées seront bien conformes aux performances attendues.

Etude d'un système

Dans les sciences de l'ingénieur, deux approches scientifiques permettent de déterminer et prévoir les réponses et les performances réalisées par un produit, à partir des informations dont on dispose sur l'environnement.

- l'approche par simulation (on travaille dans le virtuel) : elle est basée sur une modélisation mathématique
- l'approche par essai ou mesure (on travaille dans le réel) : consiste à réaliser un prototype/une maquette du produit, le mettre dans les conditions réelles ou identiques, puis de mesurer (en s'aidant d'instruments) les réponses et les modifications des grandeurs de l'environnement.

Diagnostic en conception

La conception est validée lorsque l'écart entre les résultats/comportements/performances attendus et ceux observés est le plus faible possible.
De fortes hypothèses :

- le modèle pour la simulation est fidèle au produit.
- l'essai peut venir confirmer les résultats de la simulation
- les conditions de l'essai sont représentatives des situations réelles

Lorsque l'écart est grand, on doit effectuer un diagnostic pour trouver les causes et les corriger.

Raccourci - construction de logiciels

- Chaîne : algorithmique, programmation, compilation
  Algorithmique avancée (Cormen &Al ; Knuth, ...)
  Ingénierie de la compilation : logique, langages, automates
- Programmation modulaire/structurée : développement de bibliothèques
- Analyse structurée (SA), fonctionnelle, relationnelle (BD)
- Divers cycles de vie (Cascade/*waterfall*, Spirale ! autour de prototype ; incrémental ; cyce de Balzer) - Sciences de l'ingénieur : cycle d'un produit
- Conception et programmation orientée Objets (1990) : développement de bibliothèques, de langages
- Programmation fonctionnelle
- Réseaux, Internet, Services : Middleware
- Environnements d'ingénierie - IDE - Open Source

Cycle de vie

Cycle de vie = toutes les étapes du développement logiciel, de sa conception à sa construction, sa maintenance, et à son abandon ou disparition.

Le cycle de vie est constitué des étapes suivantes :

- Définition des objectifs et des finalités du logiciel ou du système global comprenant le logiciel
- Analyse des besoins : recueil et formalisation des besoins d'un client, et des contraintes qui sont liées aux besoins et au contexte
- Spécification (*Software requirement analysis document*, le quoi du logiciel à construire), les fonctionnalités, et les contraintes
- Conception générale puis détaillée : architecture globale du projet/logiciel ; puis détail de chaque composant ; c'est le comment.
- Implantation : c'est la phase de réalisation (dans un langage et un environnement précis) du logiciel élaboré lors de la conception.

Etapes du cycle de vie (suite)

- Test unitaire des composants
- Test global (ou d'intégration)
- Validation de la conformité de la réalisation aux objectifs initiaux (spécifications)
- Mise en production : déploiement auprès du client, et validation
- Maintenance (corrective et évolutive)

Plusieurs cycles de vie

Il y a plusieurs cylces de cie du logiciel. Par exemple les cycles :

- cycle en Cascade (1970),
- cycle en V (années 1980),
- cycle en Spirale (Boehm 88),
- cycle de Balzer (Balzer, 1989),
- cycle itératif (1990, utilisé dans les méthodes *RAD Rapid Application Development*)
- Méthodes agiles (issues des pratiques de ces précédents cycles, spirale, RAD, XP)
- etc

Ils sont choisis et appliqués selon les besoins du projet à développer.

Cycles de vie

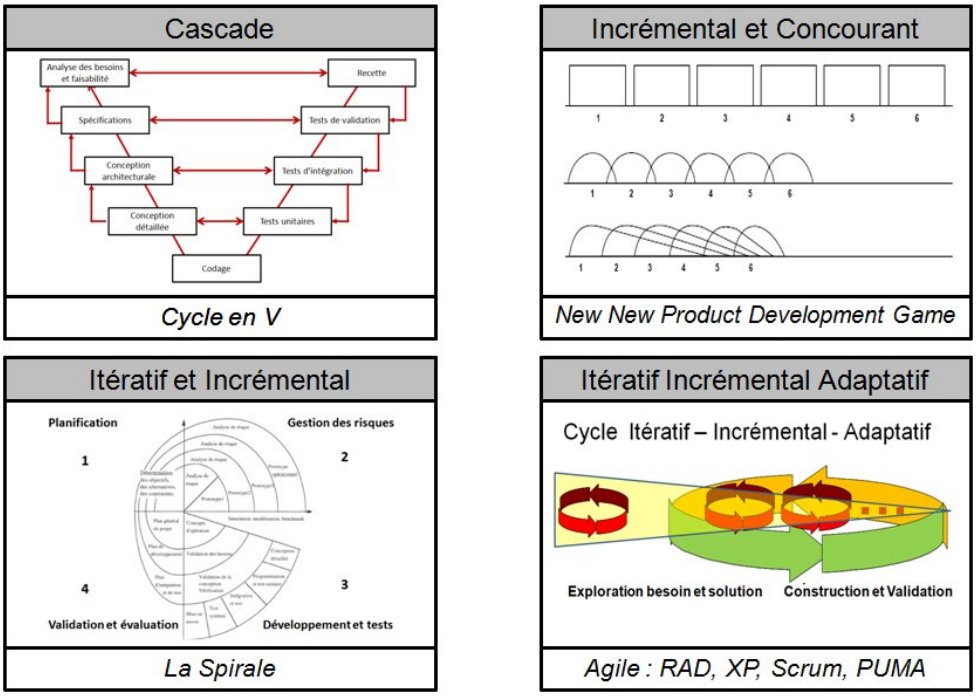

Figure: Cycle de vie (source :wikipedia)

Cycles de vie ou de développement

Tous les cycles de vie classiques ne marchent pas avec les méthodes formelles. Il faut les adapter.
Le cycle de vie de Balzer représente une nouvelle famille de cycles de vie adaptés au développement formel (ou rigoureux).

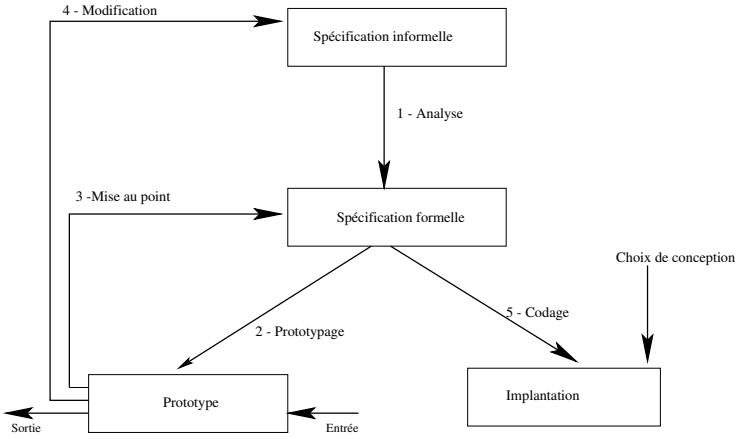

Figure: Cycle de vie de Balzer

[Balzer et al., 1989] Balzer, Robert, Bogeman, Michael, Garg Pankaj K., Schwartz, Mayer, and Shneiderman, Ben, Panel Discussion on Hypertext and Software Engineering, Proceedings of Hypertext '89, ACM Press, 1989

Références

- Software Engineering, Ian Sommerville
- Software Engineering Body of Knoweledge (Swebok V3)
- ACM SE 2004 `http://sites.computer.org/ccse/`
- ACM SE 2014 `https://www.acm.org/education/se2014.pdf`
- Ihe International Council on Software and Systems Engineering (INCOSE),
  Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 3.2.2, International Council on Systems Engineering, 2012
- INCOSE, Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 3.2.2, International Council on Systems Engineering, 2012

Ingénierie système (*Systems Engineering*)

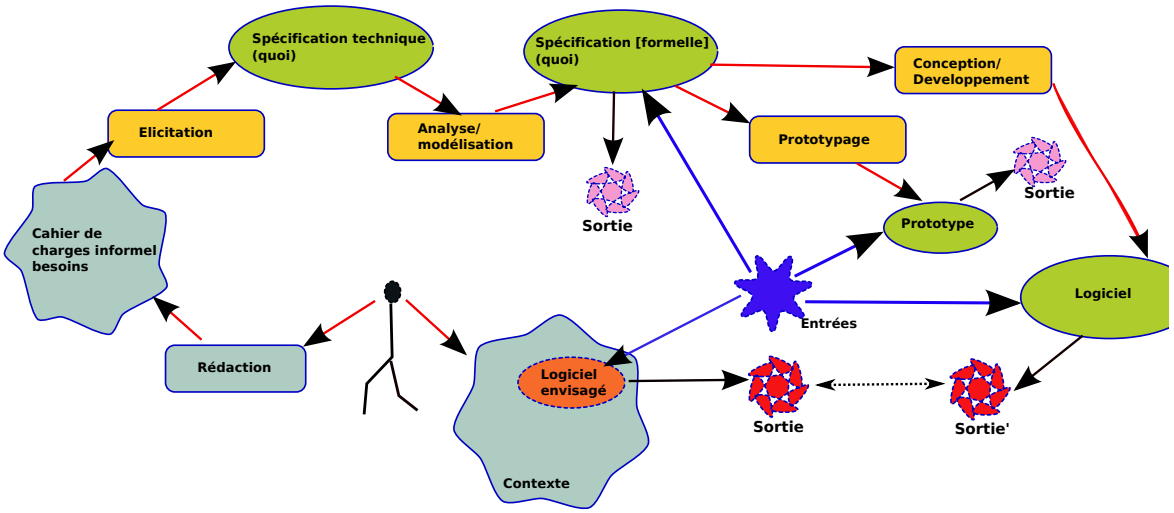

Figure: Construction du logiciel

Ingénierie système (*Systems Engineering*)

Système

Un système est un ensemble de composants en interaction ; le comportement du système dépend non seulement des comportements individuels des composants, mais surtout de l'interaction entre les composants.

- la notion de système se décline en plusieurs niveaux : un composant peut être vu à son tour comme un système et ainsi être décomposé en d'autres sous-composants (sous-systèmes).
- Un composant peut être physique (matériel) ou logiciel.
- L'ingénierie des systèmes offre un cadre adapté à l'étude des produits pluritechnologiques.
- Des langages comme SysML, AADL permettent de décrire un système avec ses composants.

Ingénierie système

Systems Engineering: the Process (*System Engineering*, Boeing)
A process that transforms an operational need or market opportunity into a system description to support detail design.

- Requirements Analysis
- Functional Analysis
- Synthesis
- Systems Analysis / Management

Ingénierie système

Systems Architecture: a Product (*System Engineering*, Boeing)
The design team's interpretation / implementation of customer requirements communicated through:

- System usage scenarios (i.e., Use Cases)
- Functional components & interrelationships
- Physical subsystems & interfaces
- Etc

Ingénierie système

Benefits of Architecting (*System Engineering*, Boeing)

- Identifies all system elementsEarlier vs. Later
- Matches function to requirements
- Capture & communicate key concepts
- Results in one design
- Manages increasing complexity
- Allows modular design

Ingénierie système

Summary

- Increasingly complex systems drive a need for better, clearer design descriptions
- Architectures convey the system designer's interpretation of the requirements
- Architectures may be presented by a variety of views which collectively describe the system
- As part of the systems engineering process, systems architecting defines and manages development of a system

Les méthodes en ingénierie

Domaines variés, tailles variables (petites ou grandes)
Projets informatiques complexes ⇒ méthodes, techniques, outils

- Méthodes d'analyse,
- Méthodes de conception,
- Méthodes de développement,

Il y a des Outils à tous les niveaux.

Méthodes d'analyse

Quelques méthodes semi-formelles

- Analyse fonctionnelle (SADT par exemple),
- Analyse structurée (SA, SSADM), SA-RT (Temps-Réel),
- Entités/Associations, Merise, Axiale,
- JSD/JSP,
- Analyse Orientée Objet, OMT, UML,
- Architecture de logicielle (*System Level*),
- etc

Méthodes : motivations

Besoin de méthodes rigoureuses pour certains domaines :

- Sécurité, Certification, Coût, Maintenance
- ITSEC (Information Technology Security Evaluation Criteria) exigent l'usage de méthodes **formelles**
- Echec (d'un vol) de ARIANE !, Erreur du Pentium, etc, etc
- Milieux hostiles à l'homme (nucléaire, chimie, marin, etc)
- Systèmes embarqués (véhicules, équipements, etc)
- Automates (domaine médical, etc)
- etc

Analyse des exigences (*Software requirements analysis*)

Analyse des exigences

Les exigences du logiciel (*Software requirements*) expriment les besoins et les contraintes liés à un logiciel qui (peut) contribue(r) à la solution d'un problème réel.

On a des exigences sur le produit/logiciel ou des exigences sur la démarche de réalisation.

Une propriété essentielle de toute exigence, est qu'elle soit vérifiable en tant que caractéristique élémentaire, comme exigence fonctionnelle ou comme exigence non-fonctionnelle au niveau global d'un système.

Elicitation, analyse, spécification, vérification, validation des besoins d'un (projet) logiciel.

Fondamentaux

Les exigences fonctionnelles décrivent les fonctions que le logiciel doit fournir.

Les exigences non-fonctionnelles sont celles qui contraignent la solution fournie par le logiciel.

Les propriétés non-fonctionnelles sont souvent vues commes des contraintes ou des exigences de qualité.

Les propriétés non-fonctionnelles peuvent être classées en plusieurs sous-catégories :

- exigences de performance, exigences de maintenabilité, exigences de sureté,
- exigences de fiabilité, exigences de sécurité, exigences d'interopérabilité,
- etc

Sources des exigences et élicitation des exigences

Les sources des exigences d'un logiciel peuvent être nombreuses et variées ;
il faut les identifier pour en faire un recueil approprié (élicitation des besoins).
Il y a différentes techniques d'élicitation des besoins :

- Interviews, scenarios (tels que les *use case de UML*),
- Réunions de groupe, observation, *User stories*, ...
- Prototypes/maquettes

Réaliser ensuite un cahier de charges (spécification informelle des besoins).

Analyse des exigences (*Requirements Analysis*)

L'analyse des exigences permet

- de détecter et résoudre d'éventuels conflits entres les exigences.
- de découvrir les limites du logiciel et comment il interagit son environnement,
- d'élaborer les exigences systèmes pour en dériver les exigences du logiciel.

Les approches classiques d'analyses des exigences intègrent la modélisation (conceptuelle) en utilisant des méthodes telles que l'analyse structurée. On peut y ajouter la classification des exigences.

Classification des exigences (*Requirements Classification*)

- Exigences fonctionnelles ou non-fonctionnelles (délais/temps, sécurité, qualité).
- Exigences sur le produit (logiciel) ou exigences sur la procédure (ITIL, SIL, ...)

Les exigences sur la procédure peuvent contraindre le choix des contractants, la démarche d'ingénierire logicielle à adopter, les normes ou méthodes à utiliser, etc.

Méthodes/Outils d'analyse des exigences

- KAOS : *Knowledge Acquisition in automated specification* ou *Keep All Objectives Satisfied*
- Tables de Parnas : formalisme rigoureux pour expliciter les besoins
- SCR (Software Cost Reduction),
- RSML *(Requirements State Machine Language)*

Modélisation conceptuelle (*Conceptual Modeling*)

L'élaboration de modèles pour un problème réel est fondamentale pour l'analyse des exigences du logiciel.

Les modèles aident à comprendre les situations dans lesquelles le problème survient, aussi bien que pour esquisser une solution.

Ainsi les modèles conceptuels comprennent les modèles des entités du domaine du problème, configurés pour reflecter leurs inter-relations et dépendances.

Modélisation conceptuelle (*Conceptual Modeling*)

Les modèles incluent par exemple :

Exemples de modèles

des modèles à états, (diagrammes états-transitons)
des modèles de flots de données,
des modèles à objets,
des modèles orientés buts,
des diagrammes de cas d'utilisation,
des interactions entre usagers,
etc

Parmi les facteurs qui influent sur le choix du formalisme de modélisation : il y a notamment la nature du problème ou le type de logiciel.

Modélisation formelle et analyse

L'expression formelle des exigences requiert l'emploi d'un langage avec une sémantique précise.
Le recours à l'analyse formelle des exigences permet :

- de les spécifier précisement et sans ambiguités évitant ainsi les mauvaises interprétations (modèles formels)
- de pouvoir raisonner sur les exigences, de prouver les propriétés du logiciel spécifié.

L'analyse formelle requiert des spécifications/modélisations formelles comme point de départ.

Des considérations pratiques

Le traitement des exigences couvre tout le cycle de vie du logiciel.

- Changement des besoins initiaux
- Maintenance des exigences dans l'état qui réflettent le logiciel à construire, ou du logiciel qui est construit.
- Tracabilité des exigences.

Références

- Parnas's Tables
  `https://cs.uwaterloo.ca/~jmatlee/talks/parnas01.pdf`
- Beck, A., Boeing, G., & Shannon, D. *Systems and Methods for Analyzing Requirements. US Patent 8650186*, 2014
- Chemuturi, M. (2013). *Requirements Engineering and Management for Software Development Projects*, ISBN 978-1-4614-5376-5, 2013
- Software Development Process–activities and steps
  `http://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf`