Université Abdelhamid Mehri – Constantine 2

2021-2022. Semestre 2

Master-Génie Logiciel

Spécification et Vérification des Systèmes (SVS)

– Cours –

Chapitre 01 : Introduction

Staff pédagogique			
Nom	**Grade**	**Faculté/Institut**	**Adresse e-mail**
BELALA Faiza	Professeur	Nouvelles Technologies	Faiza.belala@univ-constantine2.dz
LATRECHE Fateh	MCA	Nouvelles Technologies	Fateh.latreche@univ-constantine2.dz

Etudiants concernés			
Faculté/Institut	**Département**	**Année**	**Spécialité**
NTIC	TLSI	Master 1	Génie Logiciel

Objectifs du cours

- Quelle est la différence entre la modélisation et la conception d'un système?
- Quels sont les types de systèmes distribués qui peuvent exister? quel est l'intérêt de leur modélisation?
- Quelle relation existe-il entre Modélisation et Vérification?

1. Introduction

La technologie des systèmes mixtes matériels et logiciels connaît de nos jours une révolution sans pareille dans tous les domaines technologiques. Le fait de mélanger matériel et logiciel, ceci a donné naissance à des systèmes très complexes. La conception de tels systèmes, nécessite l'intervention de ressources humaine, financière et technologique très importante.

Pour veiller au bon fonctionnement de leurs conceptions, ces systèmes doivent être soumis à des processus de validation très rigoureux. Une telle tâche n'est point évidente vue l'accroissement rapide de la complexité et de l'hétérogénéité des nouvelles architectures (matériel/logiciel).

Aussi correcte que possible, la conception doit tenir compte de certaines propriétés dites critiques, qui sans leur présence risque d'engendrer une situation de catastrophe et même de ternir l'image de la compagnie.

Dans ce chapitre, nous rappelons les différents types de systèmes distribués, tout en identifiant la différence qui peut exister entre leur conception et leur modélisation. Enfin, nous soulignons particulièrent l'intérêt de recourir aux méthodes formelles lors de la phase conception des systèmes distribués critiques pour pouvoir les vérifier.

2. Types de systèmes Distribués

Le terme « ingénierie » est un terme introduit de manière récente dans la langue française où il se substitue parfois au terme « génie » désignant l'art de l'ingénieur.

Le génie logiciel (en anglais : software engineering) désigne l'ensemble des méthodes, des techniques et outils concourant à la production d'un logiciel, au-delà de la seule activité de programmation.

L'ingénierie désigne l'ensemble des fonctions allant de la conception et des études, y compris la formalisation des besoins des utilisateurs, à la responsabilité de la construction et au contrôle des équipements d'une installation technique ou industrielle. Il est aussi souvent utilisé dans un sens étendu à d'autres domaines

Les systèmes informatiques aujourd'hui sont de plus en plus **importants**; omniprésents dans notre vis quotidienne. Ils sont devenus de plus en plus **complexes** vu l'utilisation des réseaux et des objets intelligents et un certain nombre de préoccupations.

Un système informatique est constitué d'un ensemble de composants (sous systèmes) agencés d'une certaine manière qui intéragissent (mémoire partagée, événements, messages, etc).

Généralement, un système est défini selon deux dimensions:

1) Aspect statique: données (types, valeurs, etc)
2) Aspect dynamique: comportement (actions sur les données, actions d'interaction, etc.)

Quelques exemples de systèmes réels: un chronomètre, un pont à une voie, etc

Il existe plusieurs types de systèmes informatiques, chacun est utilisé dans un contexte particulier:

- Système ouvert (au contraire d'un système clos): il peut interagir avec un environnement (soit un autre système ouvert, soit un utilisateur).
- Système conversationnel (au contraire d'un système réactif): il est capable de dialoguer avec pertinence avec ses utilisateurs mais le temps qu'il veut, il asynchrone et souvent non-déterministes,
- Système réactif: est un système ouvert répondant constamment aux sollicitations de son environnement en produisant des actions sur celui-ci. Il est synchrone (réponse continue) et souvent déterministe.
- Système distribué (au contraire d'un système centralisé): est un ensemble de sous-systèmes concurrents (indépendants) s'exécutant sur une ou plusieurs machines via un réseau.

- Système communicant : importance des échanges entre systèmes vs. actions internes
- **Objets communicants** : des systèmes dotés de la capacité d'échanger (recevoir, interpréter et communiquer) des informations avec d'autres objets. C'est une technologie encore jeune et en très forte croissance utilisée dans de nombreux domaines.

3. Modélisation Versus Conception

Supposons que nous nous intéresserons au développement d'un système distribué et réactif,

- Si on est un peu rigoureux ...
  - Méthode de conception : UML, …
  - Démarche/gestion de projet: RUP, …

... mais cela ne suffit pas !

- Si on veut appliquer plus de rigueur … (en vérifiant certaines contraintes)

De nombreux formalismes doivent être appliqués !

Dans la figure suivante, nous montrons la nécessité du génie logiciel et comment les modèles formels sont intégrés dans le processus de dévéloppement des systèmes complexes (cycle en V), formés de sous-systèmes et comment les propriétés correspondantes sont exprimées et vérifiées.

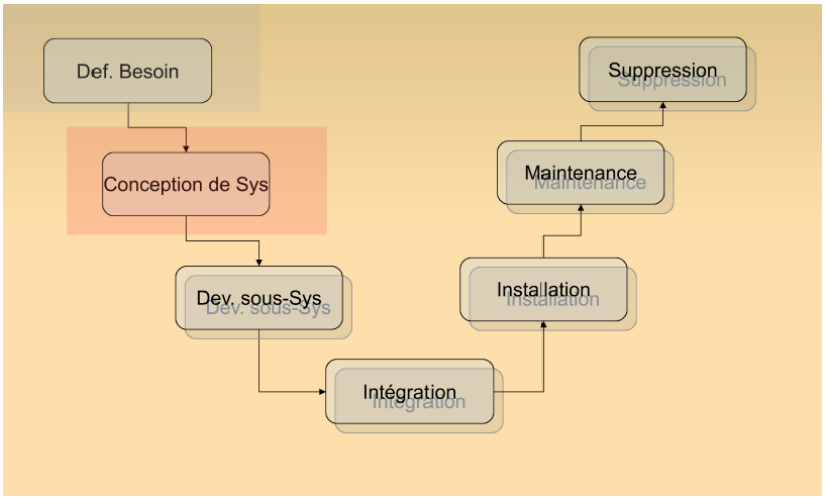


Figure 01: Cycle en V

3.1. Définition de conception

Concevoir (to design) consiste à définir un système ou un sous-système. Cela implique en général de définir un ou plusieurs modèles du système et de raffiner les modèles jusqu'à ce que les fonctionnalités désirées soient obtenues en tenant compte des contraintes de l'implémentation. On peut dire qu'un système est entièrement conçu lorsque le modèle qu'on en a est directement exécutable par la plateforme cible que l'on a choisie.

La conception et la modélisation sont donc étroitement liées, puisque la conception entraîne la modélisation. Dans certaines circonstances, des modèles peuvent être immuables : il décrivent des sous-systèmes, des contraintes ou des comportements qui sont imposés.
Par exemple, un système d'asservissement de position peut utiliser un modèle du dispositif mécanique à asservir. Ce modèle n'entre pas dans le processus de conception mais est une contrainte pour la conception de l'asservissement.
Les modèles exécutables sont parfois appelés simulations, ce qui est approprié quand le modèle exécutable est clairement distinct du système qu'il modélise. Toutefois, dans de nombreux systèmes électroniques, un

modèle qui était au départ une simulation finit par devenir une implémentation logicielle du système. La distinction entre le modèle et le système proprement dit devient alors floue. Cette situation se rencontre très fréquemment dans le domaine des systèmes enfouis.

Un modèle prédictif est construit selon un modèle de calcul (model of computation) qui est l'ensemble des lois qui gouvernent l'interaction des composants dans le modèle. La théorie newtonienne de la gravitation est ainsi un modèle de calcul des interactions entre corps dues à leur masse. La relativité générale d'Einstein en est un autre.
Le jeu de règles utilisé pour calculer le comportement des composants d'un modèle est un modèle d'exécution du modèle de calcul. Un modèle de calcul peut avoir plusieurs modèles d'exécution : il peut exister plusieurs jeux de règles qui donnent les mêmes comportements des composants. Par exemple, pour calculer le mouvement de planètes soumises à la gravitation newtonnienne (modèle de calcul), on peut utiliser la méthode d'intégration d'Euler ou celle de Runge-Kuta (modèles d'exécution) pour intégrer numériquement les équations différentielles du système.

Le choix d'un modèle de calcul dépend du type de modèle que l'on construit. La possibilité de transformer un modèle en implémentation dépend fortement du modèle de calcul utilisé.
Certains modèles de calcul sont bien adaptés à la génération de composants matériels, alors que d'autres seront plus facilement implémentés en logiciel. Pour les systèmes informatiques d'actualité les modèles les plus utiles sont ceux qui supportent le temps et le parallélisme.

3.2. Définition de modèles

Modéliser (to model) consiste à construire une représentation formelle d'un système ou d'un sous-système.
Un modèle peut-être une construction mathématique, c'est-à-dire un ensemble d'assertions sur les propriétés du système.

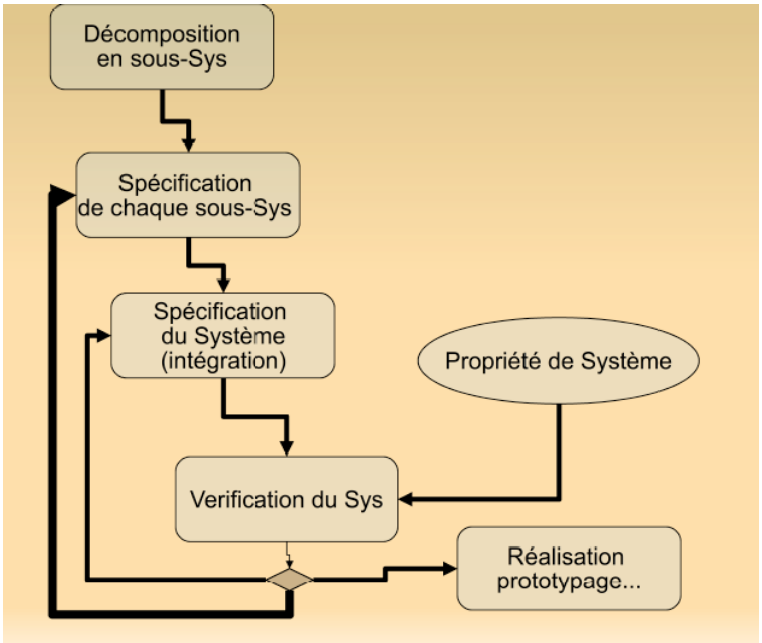
Un modèle peut aussi être **constructif**, auquel cas il définit une procédure calculatoire qui imite un ensemble de propriétés du système. Les modèles constructifs sont généralement utilisés pour décrire la réponse d'un système aux stimuli que lui fournit son environnement. Un modèle constructif est aussi appelé un modèle exécutable. Les modèles qui ne sont pas constructifs ont aussi leur utilité.

Un modèle peut en effet être simplement **explicatif** : il explique ce que l'on observe, et on peut alors croire ou non que ce modèle décrit la réalité. Certains modèles explicatifs sont plus intéressants car il permettent de prédire ce que l'on observera, ce sont les modèles prédictifs.

Un modèle **prédictif** non exécutable, par exemple la théorie newtonienne de la gravitation, peut être rendu exécutable en lui adjoignant une procédure de calcul des propriétés du système à partir des assertions que le modèle fait sur ces propriétés.
Dans le cas général, on n'obtient qu'une approximation de la valeur des propriétés (par intégration numérique d'équations différentielles dans notre exemple), mais cette approximation peut être suffisante, même pour envoyer des hommes sur la Lune.

3.3. Modèles formels et conception

Une première forme de spécification des systèmes distribués est la programmation. Bien qu'elle permet d'utiliser n'importe quel langage de programmation de haut niveau, elle reste limitée par rapport à la vérification de ces systèmes, voire impossible. De plus, leur développement est très couteux en temps et le risque de détection d'erreur est plus tardive. Par conséquent, le besoin d'un processus de spécification formelle est primordial.
Dans le cycle de vie d'un logiciel, la phase de conception d'un système est décomposée en plusieurs sous phases (voir figure ci-dessous) pour spécifier et vérifier formellement ce système.



Une spécification consiste en la représentation ou modélisation d'un système et ses propriétés désirées. En génie logiciel, les méthodes formelles sont des techniques permettant de raisonner rigoureusement, à l'aide de logique mathématique, sur un programme informatique ou du matériel électronique numérique, afin de démontrer leur validité par rapport à une certaine spécification.

Elles reposent sur les sémantiques des programmes, c'est-à-dire sur des descriptions mathématiques formelles du sens d'un programme donné par son code source.
Ces méthodes permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels. Elles sont donc utilisées dans le développement des logiciels les plus critiques.

Ainsi, les méthodes formelles peuvent être utilisées pour donner une spécification du système que l'on souhaite développer, au niveau de détails désiré.
Une spécification formelle du système est basée sur un langage formel dont la sémantique est bien définie (contrairement à une spécification en langage naturel qui peut donner lieu à différentes interprétations). Plusieurs langages (u formalismes) formels existent, nous pouvons citer sans être exhaustif, les systèmes états-transition, les réseaux de Petri, les automates communicants, les nombreuses algèbres de processus (CCS, CSP, FSP, LOTOS, mCRL2, …), les différentes logiques (Horn, rewriting logic, …)

Une description formelle du système peut être utilisée comme référence pendant le développement. De plus, elle peut être utilisée pour vérifier (formellement) que la réalisation finale du système (décrite dans un langage informatique dédié) respecte les attentes initiales (notamment en termes de fonctionnalité).

4. Modélisation pour la vérification

Une spécification peut être utilisée comme base pour prouver des propriétés sur le système. La spécification est le plus souvent une représentation abstraite (avec moins de détails) du système en développement : débarrassé de détails encombrants, il est en général plus simple de prouver des propriétés sur la spécification que directement sur la description complète et concrète du système.

Les méthodes formelles prennent tout leur intérêt lorsque les preuves elles-mêmes sont garanties correctes formellement. On peut distinguer deux grandes catégories d'outils permettant la preuve de propriétés sur des modèles formels :

1. la preuve automatique de théorème, qui consiste à laisser l'ordinateur prouver les propriétés automatiquement, étant donnés une description du système, un ensemble d'axiomes et un ensemble de règles d'inférences.
2. le model checking (voir la figure suivante), qui consiste à vérifier des propriétés par une énumération exhaustive et astucieuse (selon les algorithmes) des états accessibles.

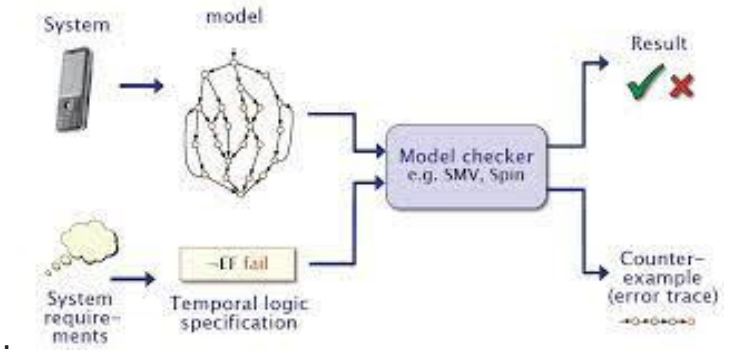
L'outil de model-checking prend en entrée:

> M : Modèle (formel) du comportement (issu de Rdp, AC, Alg.Proc,..), qui permet de décrire l'ensemble de tous les comportements du système.
>
> P : Modèle (formel) des propriétés (formalisme dédié), qui permet de formaliser les exigences à respecter.

et rend en sortie, soit:

- un résultat positive, si le modèle M satisfait la propriété P.
- un contre exemple.

L'efficacité de cette technique dépend en général de la taille de l'espace des états accessibles et trouve donc ses limites dans les ressources de l'ordinateur pour manipuler l'ensemble des états accessibles. Des techniques d'abstractions (éventuellement guidées par l'utilisateur) peuvent être utilisées pour améliorer l'efficacité des algorithmes.



5. Conclusion

On pourraît imaginer se passer des méthodes formelles pour concevoir un système critique, et se contenter du code (programmation). Malheureusement, plusieurs choses sont mixées (intérêt ou non ?) ce qui allourdit la simulation/test à la main (non exhaustif sans formel) et rend son développement (Très) compliqué, d'autant plus si le système est distribué et en interaction.

Il a bien fallu se tourner vers des méthodes plus précises. Pas pour s'exprimer dans la vie courante, ni développer des idées philosophiques. Mais simplement dans un cadre de génie logiciel, pour permettre à des intervenants de travailler ensemble sur un ouvrage commun : la conception d'une machine, d'un produit industriel, d'un ordinateur, d'une procédure informatique, d'un système d'information, etc.

L'objectif d'une méthode de *spécification formelle* est de pouvoir décrire des concepts peut-être compliqués, mais finalement rendus très simples, car on peut toujours décomposer un objet complexe en sous-objets élémentaires. Cette méthode décomposition permet au *modèle conceptuel* de représenter un aspect du réel sans erreurs d'interprétation possibles.