Chapitre 2 : Les modèles enrichis sous l'angle des IHMs

Objectifs spécifiques

À l'issue de ce chapitre, l'apprenant sera capable de :

- Connaître les nouveaux modèles de développement enrichis sous angle des IHM
- Distinguer la différence entre les modèles de développement traditionnelles (classiques) et les nouveaux modèles de développement enrichis sous angle des IHM;
- Faire une synthèse globale entre ses modèles de développement.

2.1 Introduction

Les modèles de développement traditionnelles (classiques) ont pour objectif commun de produire des logiciels de qualité. Mais on s'aperçoit que ces modèles sont bien trop souvent orientés vers la partie technique (le code) et non vers l'utilisateur. En effet, ils sont orientés système et n'intègrent pas explicitement par exemple l'analyse et la modélisation des tâches potentielles des utilisateurs.

Les insuffisances de ces modèles vis-à-vis des systèmes interactifs (analyse de tâches, facteurs humains, etc.) ont amené à définir des modèles dits enrichis sous l'angle de l'IHM. Ces modèles impliquant l'utilisateur (composante humaine) dans le processus de développement dits centrés utilisateur montrent des évolutions apportées par le domaine des IHM au Génie logiciel (GL) en se focalisant sur des idées essentielles pour le développement de systèmes interactifs.

Chapitre 2 : Les modèles enrichis sous l'angle des IHMs

2.2 Rappel sur le cycle de vie du logiciel

Le cycle de vie d'un logiciel est constitué de l'enchaînement des différentes activités nécessaires à son développement**.** Il comprend généralement au minimum les phases suivantes [Hugues ,1994] :

1. **Faisabilité (phase de planification et étude de faisabilité) :** Durant cette phase une discussion conjointe entre le développeur du logiciel et le demandeur (client) afin de se mettre en accord sur les coûts et la durée du projet. La phase de planification permet de découper le projet en tâches, de décrire leur enchaînement dans le temps, d'affecter à chacune une durée. Il est également important de définir la méthode de conception à utiliser durant le processus de développement. A l'issue de cette phase on obtiendra les résultats suivants :
   - Le **plan qualité ;**
   - Le **plan projet** destiné aux développeurs ;
   - Une **estimation des coûts** réels (utile pour le management) ;
   - Un **devis** destiné au client précisant le prix à payer, les délais et les fournitures ;
   - Une liste des **dépendances extérieures** (comme par exemple l'arrivée d'une nouvelle machine ou d'un nouveau logiciel).

2. **Analyse des besoins (appelé aussi spécification des besoins) :** Après avoir confirmé la faisabilité de logiciel, le développeur consiste à faire une analyse détaillée des différentes fonctions que le logiciel doit implémenter. Cette phase doit contenir la mise du logiciel dans son contexte (type de produit, nouveau/altéré) et l'étude de l'existant. L'étude de l'existant désigne l'étude des produits similaires dans le marché (veille concurrentielle) et l'étude du processus ou des logiciels similaires à l'entreprise. Les besoins de l'entreprise peuvent contenir des besoins fonctionnels (des fonctionnalités que le produit doit automatiser) et des besoins non fonctionnels (disponibilité, rapidité de calcul, etc.). A l'issue de cette phase on obtiendra les résultats suivants :
   - Un **dossier d'analyse** comprenant les spécifications fonctionnelles et non fonctionnelles du produit ;
   - Une ébauche du **manuel utilisateur ;**
   - Une première version du **glossaire** contenant les termes propres au projet.

3. **Conception (architecturale et détaillée) :** Pendant cette phase l'architecture du logiciel est définie ainsi que les interfaces entre les différents modules. On veillera tout particulièrement à rendre les différents constituants des produits aussi indépendants que possible de manière à faciliter à la fois le développement parallèle et la maintenance future. A l'issue de cette phase on obtiendra les documents suivants :
   - Un dossier de conception ;
   - Un plan d'intégration ;
   - Les différents plans de tests ;
   - Un planning mis à jour.

4. **Codage (Implémentation ou programmation) et tests unitaires :** Cette phase consiste à traduire dans un langage de programmation des fonctionnalités définies lors de la phase de conception. Ensuite, il s'agit de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications. A l'issue de cette phase on obtiendra les résultats suivants :
   - Les modules codés et testés ;
   - La documentation de chaque module ;
   - Les résultats des tests unitaires ;
   - Le planning mis à jour.

5. **Intégration :** Durant cette phase, chaque module testé est intégré avec les autres suivant le plan d'intégration et l'ensemble est testé conformément au plan de tests. Elle utilise la gestion de configuration pour assembler des versions cohérentes de chaque composant. A l'issue de cette phase, on obtiendra les produits intermédiaires suivants :
   - Un rapport de tests d'intégration ;
   - Une manuel d'installation ;
   - Une version finale du manuel utilisateur.

6. **Qualification (ou recette) :** Lorsque le logiciel est terminé et les phases d'intégration matériel/logiciel achevées, le produit est qualifié, c'est à dire testé en vraie grandeur dans des conditions normales d'utilisation et vérifié la conformité du logiciel selon les spécifications initiales. Cette phase termine le développement. A l'issue de cette phase le logiciel est prêt à la mise en exploitation.

7. **Maintenance :** Cette phase permet d'ajuster l'application après la livraison du produit au client. Elle a pour but de corriger les erreurs et les anomalies du système et modifier le système pour y ajouter des fonctionnalités. Il existe trois types de maintenance à savoir :

    a. Maintenance **corrective** : correction des bugs.

    b. Maintenance **adaptative** : ajuster le logiciel.

    c. Maintenance **perfective**, **d'extension** : augmenter / améliorer les possibilités du logiciel.

    A l'issue de cette phase de maintenance, on obtiendra soit :

    a. Logiciel corrigé ;

    b. Logiciel mise à jour ;

    c. Documents corrigés.

2.3 Principaux modèles enrichis sous l'angle de l'IHM

Dans cette section, nous présentons quelques modèles enrichis sous l'angle de l'IHM à savoir : modèle en V enrichi, modèle en couche, modèle PRODUSER, modèle en étoile, et modèle Nabla.

2.3.1 Le modèle en V enrichi

Le GL propose un cadre structurant du processus de développement des systèmes informatiques. Dans ce cadre, l'ergonomie s'inscrit en des étapes bien précises avec ses méthodes et techniques. La Figure 2.1 montre la répartition des étapes du processus de développement en deux espaces : l'espace IHM et l'espace logiciel.

- Le premier se caractérise par la priorité qu'il faudrait accorder aux aspects ergonomiques,
- Le second par l'accent mis sur les techniques d'implémentation logicielle.

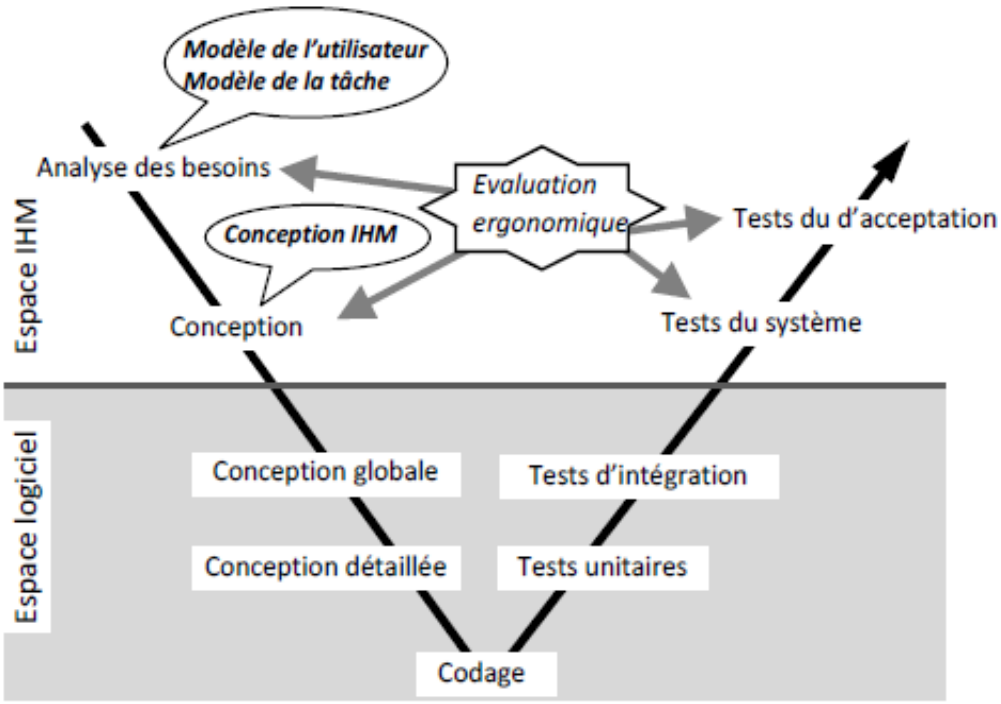

Figure 2.1 : le modèle en V enrichi

En effet, l'espace IHM inclut l'analyse des besoins, la conception et les tests du système et d'acceptation. Dans la phase d'analyse des besoins, il faut inscrire l'apport de l'ergonomie et de la psychologie pour l'analyse de tâche et la modélisation de l'utilisateur. Dans l'étape de conception intervient l'ergonomie pour la définition et la spécification de l'interface utilisateur. Les tests système et d'acceptation ainsi que l'évaluation des interfaces utilisateur devraient nécessairement faire appel à des techniques d'évaluation (comme test d'utilisabilité, questionnaire, évaluation heuristique, test à haute voix (*thinking aloud*), etc.)**.**

L'espace logiciel laisse la place aux compétences informatiques avec les conceptions globales et détaillées, le codage et les tests unitaires et d'intégration.

Néanmoins, le modèle en V enrichi autorise des retours arrière limités, ce qui peut nuire à l'aspect souvent nécessairement itératif d'une conception centrée utilisateur.

2.3.2 Modèle en couches

Le principe général du modèle en couches**,** issu de [Curtis *et al.*, 1994] est de proposer une séparation en deux parties de chaque phase du développement, une partie concernant le

développement classique de logiciel (partie de droite), et une partie concernant le problème de l'interaction homme-machine (partie de gauche) (Figure 2.1).

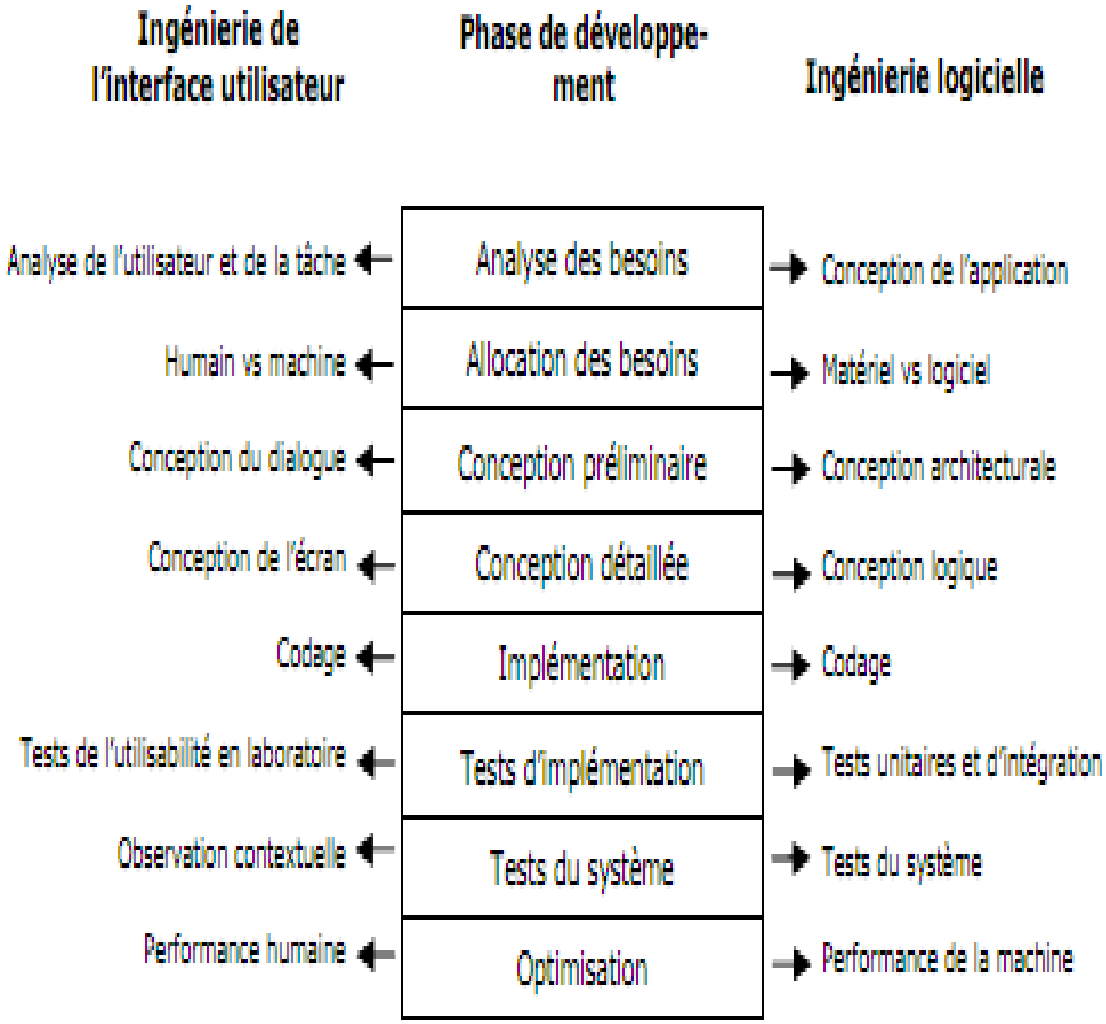

Figure 2.1 : Le modèle en couches [Curtis *et al.*,1994]

Un des avantages de ce processus est qu'il précise, pour chaque phase du développement, ce qui est à faire pour la partie fonctionnelle de l'application et sur sa partie interactive, en conservant une séparation claire entre ces deux parties d'un système interactif.

Ce modèle présente deux inconvénients :

- aucune mise en relation explicite des deux parties n'est faite ;
- le modèle est trop séquentiel, perdant les avantages d'un processus itératif comme le prototypage.

2.3.3 Le modèle PRODUSER (PROcess for Developping USER Interface)

PRODUSER (PROcess for Developing USER Interfaces) est un modèle proposé par [James ,1991], qui vise à adapter directement le modèle spirale au développement de systèmes interactifs en mettant l'accent sur leur **prototypage** (Figure 2.3) permettant d'incorporer la **gestion du risque** dans son processus pour **améliorer la qualité des interfaces utilisateur**. Il peut être applicable à toutes tailles de projet et suffisamment flexible pour aider les équipes de projet qui ne désirent pas utiliser l'ensemble des capacités de ce processus.

Le modèle propose **un cycle de quatre parties** : les besoins, les spécifications abstraites et les spécifications concrètes et l'évaluation et le contrôle du risque.

Elles sont détaillées suivant **les quatre plans** :

1. Planification de l'étape,
2. Définitions des objectifs, alternatives et contraintes,
3. évaluation des alternatives permettant de gérer le risque
4. et développement et vérification.

Ce modèle présente deux avantages :

- Il est utilisé dans les grands projets ;
- Le modèle est adaptatif c.à.d. il peut accueillir n'importe qu'elle changement des besoins des utilisateurs ;

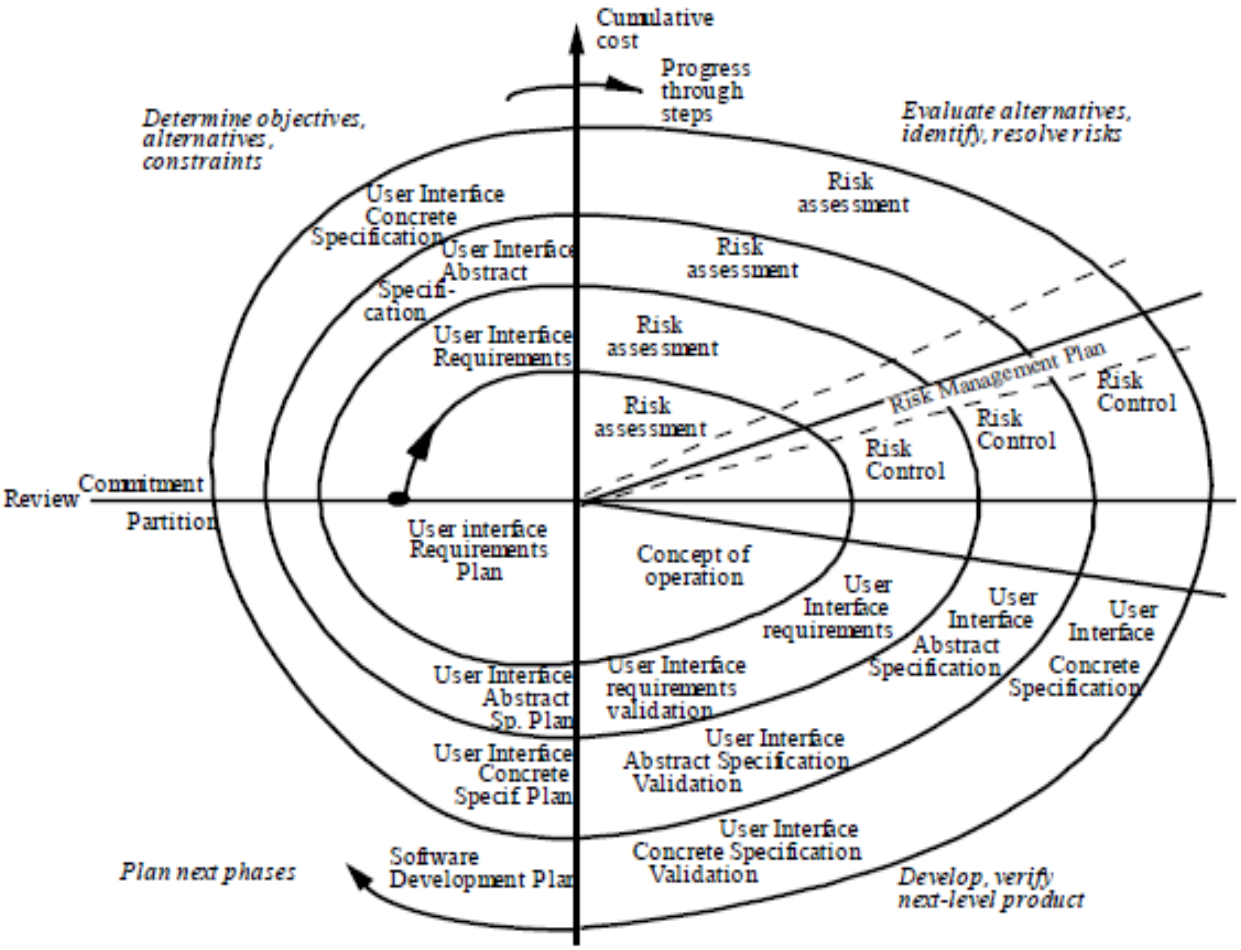

Figure 2.3 : Modèle PRODUSER James (1991).

Ce modèle présente deux inconvénients majeurs :

- L'utilisation du modèle en spirale nécessite une expérience et une expertise considérables en matière d'évaluation des risques : dans le développement de projets risqués, si le risque n'est pas identifié à temps, il entraînera inévitablement des pertes importantes.
- Evaluation des risques peut être couteux ;
- Trop d'itérations augmenteront les coûts de développement et retarderont le temps de soumission.

2.3.4 Le modèle en étoile

Le modèle en étoile [Hix *et al.*, 1993] situe l'évaluation au centre même du cycle complet. Ceci permet des interactions/itérations entre chacune des étapes. L'étape d'évaluation est vue comme une étape intermédiaire ; elle permet de protéger l'équipe de développement contre un rejet

final (Figure 2.4). Particulièrement ouvert, ce modèle est très flexible, voire un peu trop. Il n'impose pas un ordre dans lequel les étapes du processus doivent être exécutées, bien que dans la pratique les activités de développement soient placées à la fin du cycle. Il est à noter qu'il implique une conception participative (une méthode où **l'utilisateur final** d'une application **collabore de manière active à la conception**) qui vise la détection des problèmes d'utilisabilité et qu'il exige un degré élevé d'implication de l'utilisateur [Lepreux, 2005].

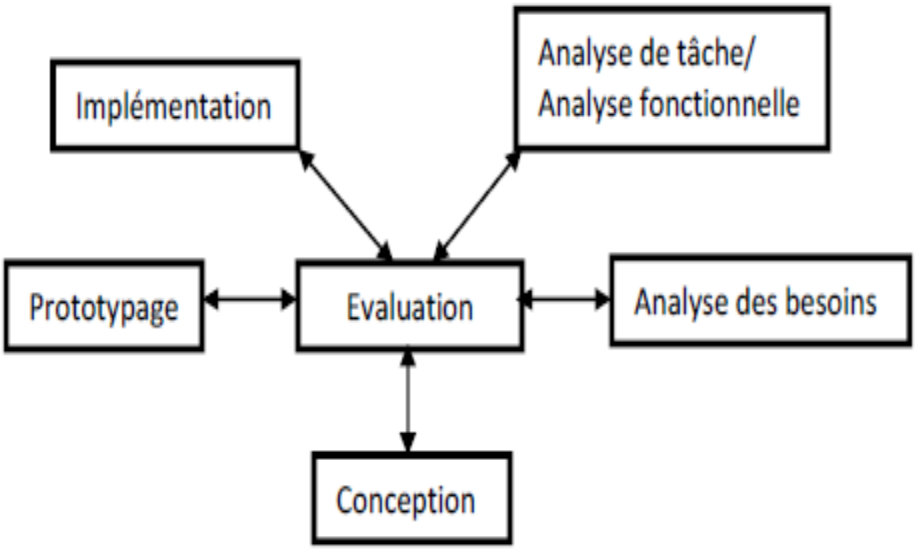

Figure 2.4 : Modèle en étoile [Hix et al. 1993]

2.3.5 Le modèle Nabla

Le modèle Nabla proposé par [Kolski, 1997,1998], et construit selon un double cycle en V situe les différentes étapes du génie logiciel visant à développer un système interactif. Deux parties sont définies dans ce modèle (Figure 2.5) :

- La partie de gauche constitue les différentes phases de réalisation de l'interface homme-machine ;
- La partie de droite se focalise sur les modules d'aide (ou applicatifs).

Ce modèle accorde une importance à l'analyse du système homme-machine qui consiste en la confrontation entre un modèle dit **« réel »** et un modèle dit **« de référence »** et ensuite en l'obtention d'un ensemble de données suffisant pour spécifier puis concevoir, réaliser et tester le système interactif.

- Un modèle dit « réel » correspondant au système homme-machine existant : trois cas peuvent être considérés lorsqu'il s'agit de mettre à jour un système homme-machine existant, la modélisation est bien entendu effectuée à partir du système existant. Lorsqu'il s'agit de créer un nouveau système homme-machine à partir d'autres systèmes déjà existants. La modélisation découle d'une synthèse des données issues de chaque analyse. Lorsqu'il n'existe pas de système homme-machine existant et que le système est entièrement à concevoir, ce modèle doit être conçu.
- Un modèle dit « de référence » qui correspond à celui d'un système homme-machine dit idéal, en considérant les points de vue et besoins des différents intervenants concernés par le système homme-machine visé. Ce modèle doit lister un ensemble de critères devant être respectés telles que : sécurité des hommes, des installations, de l'environnement ergonomie du logiciel, etc. suivant le domaine d'application considéré.

Enfin, ce modèle accorde aussi une grande importance à l'évaluation et à la validation, colonne vertébrale du modèle et aspect central du projet, qui sont vues d'un point de vue socio-ergonomique afin de vérifier la pertinence de l'intégration des solutions nouvelles dans le système Homme-Machine visé. Cette étape est située au centre du projet suggérant une démarche itérative aussi bien dans les parties gauche que droite. Elle se termine par l'étape de recette qui se différencie symboliquement en une recette orientée IHM et une recette orientée modules d'aide.

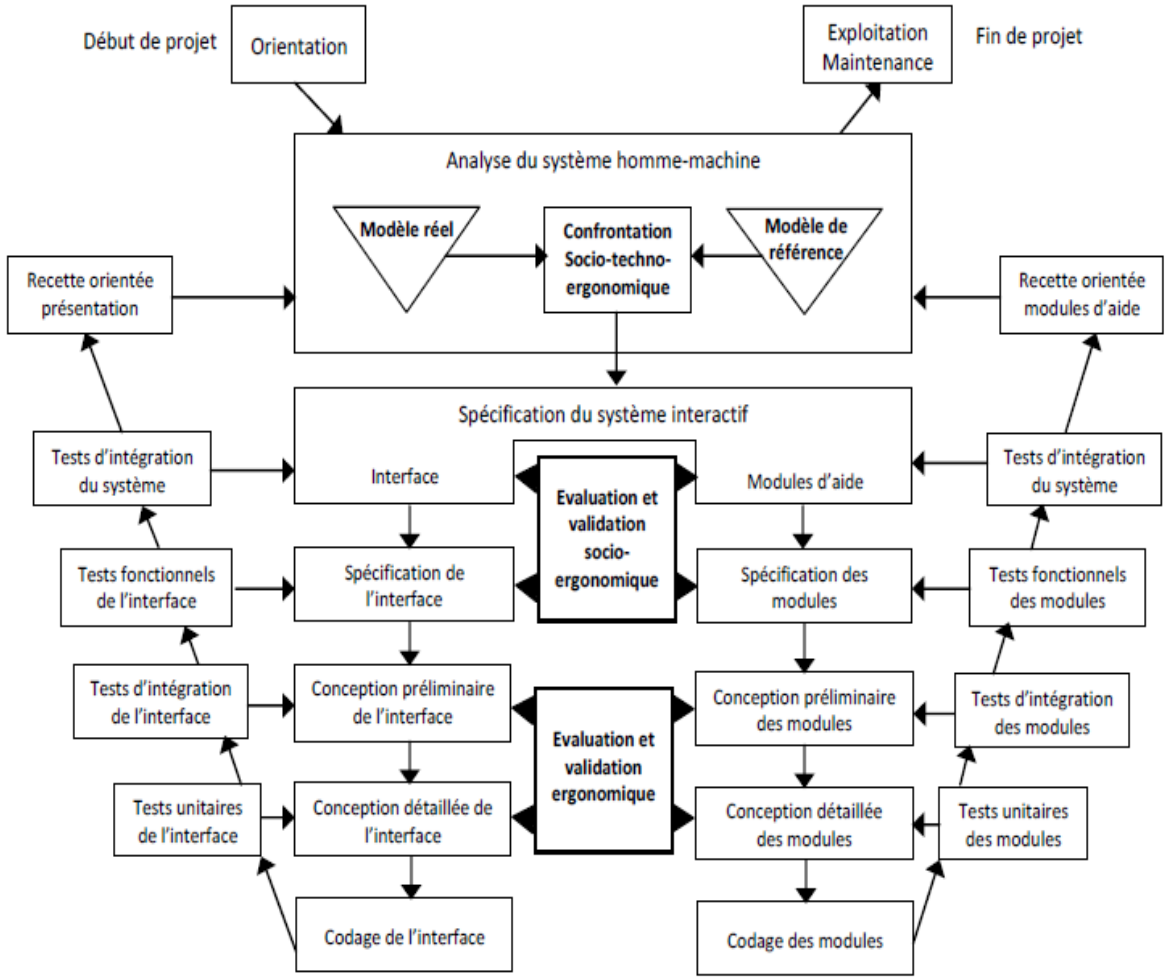

Figure 2.5 : Modèle Nabla [Kolski 1997, 1998]

Le modèle Nabla présente trois inconvénients majeurs :

- Le modèle Nabla ne positionne pas clairement la modélisation de l'utilisateur et des tâches humaines en montrant leurs relations avec la spécification de l'interface (elles sont en fait intégrées dans la boîte d'analyse du système Homme-Machine).
- Nabla (comme le modèle en V) s'exprime dans une série de retours très limités qui constituent un handicap pour la conception itérative.
- Le modèle n'indique rien au sujet du prototypage. Il présente une tentative intéressante de la part du GL pour se relier à l'ergonomie cognitive en tenant compte des facteurs humains et également de l'évaluation ergonomique.

2.4 Conclusion

Les modèles présentés précédemment, montrent les évolutions apportées par le domaine des IHM au génie logiciel. Ces modèles montrent le manque dans les modèles du génie logiciel mettent l'accent sur des idées essentielles pour le développement de systèmes hautement interactifs comme par exemple :

- Prendre en compte l'analyse et la modélisation des tâches utilisateurs ;
- Placer l'évaluation au centre du processus ;
- Modéliser les activités humaines, les interfaces homme-machine et le système ;
- et enfin confronter les modélisations des activités à effectuer (identifiées au début du cycle) avec les modélisations de ces activités réelles en utilisant le nouveau système (identifiées en fin de cycle).

Néanmoins, ces cycles enrichis sont difficilement utilisables car ils ne sont pas suffisamment complets (comme le cycle Etoile) et ils montrent de leur part des insuffisances telle que le développement itératif qui reste limité (par exemple dans le modèle Nabla).

Concernant l'évaluation, même si elle est généralement présente dans ces modèles, elle est souvent effectuée à la fin de chaque étape ou à la fin de la construction du système interactif d'où le risque de rejet en cas d'inadéquation entre le système construit et les attentes et besoins des utilisateurs. Par conséquent, ils ne permettent pas toujours une évaluation précoce.