Cinquième partie

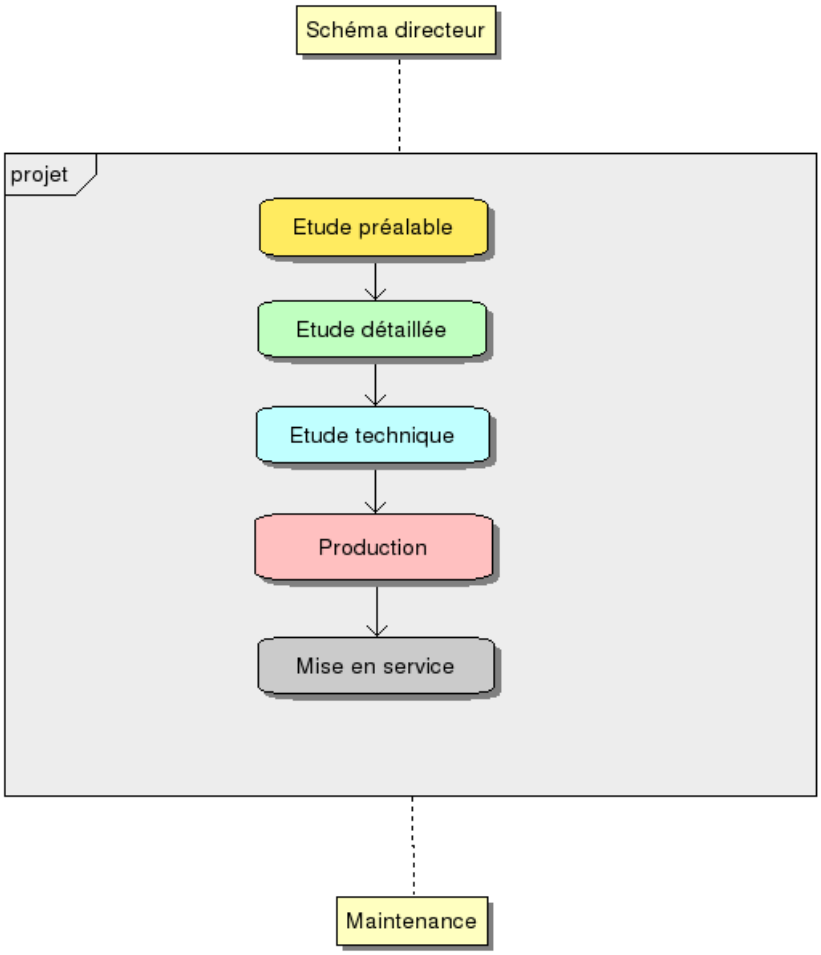
Cycle de vie

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Chapitre 11

La démarche classique de Merise



La méthode Merise est née au cours des années 80, dans un contexte où la micro-informatique restait un jouet coûteux ou un outil de travail isolé, manipulé par un travailleur enthousiaste. L'analyse concerne de grands projets, dans des grandes entreprises, sur des *mainframes*. Les PC font tourner des logiciels de bureautique et des progiciels qui modélisent de petites applications locales, le plus souvent sans aucune analyse préalable. Ce sont deux mondes séparés, dont personne ne peut encore croire qu'ils vont se rejoindre. C'est pourtant ce qui se produit. La montée en puissance des PC, leur mise en réseau leur permet de remplacer peu à peu la grosse informatique. IBM, qui a trop longtemps basé son chiffre d'affaires sur les gros systèmes, sera à deux doigts d'y laisser sa peau.

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Cette méthode reste d'actualité pour les grands projets utilisant la programmation traditionnelle.

11.1 Schéma directeur

Il permet la détermination des domaines. Utile donc à l'analyste pour intégrer son projet dans l'entreprise. Les techniques, complexes à ce niveau, concernent surtout les projets dans de grosses entreprises.

Il s'agit d'une stratégie globale des objectifs, de l'organisation générale et des perspectives d'évolution, spécifiée pour l'entreprise dans son ensemble ou pour un de ses secteurs. La description du système d'information y tient une place importante. C'est à ce niveau aussi que se situe la segmentation de l'entreprise en différents domaines, auxquels peuvent s'appliquer des analyses locales, susceptibles de donner naissance à des logiciels limités à certaines parties de l'entreprise. Dans les grosses entreprises, la prise en compte des différents domaines est aussi le seul moyen de parvenir à conserver la maîtrise de l'analyse.

La définition du schéma directeur échappe à l'analyste. L'informaticien intervient ici à titre de consommateur, en prenant acte d'une série de décisions et de contraintes de nature culturelle, organisationnelle, matérielle et budgétaire. Son avis sera peut-être sollicité pour réorienter la politique de l'entreprise sur un point précis, mais une fois ce schéma défini, il ne sera pas remis en question. Renaud Bonhomme faisait remarquer malicieusement : «le plus souvent, il faut bien dire, hélas, que [le schéma directeur] n'existe que dans les neurones des animateurs de l'entreprise, ce qui n'en facilite ni la manipulation, ni la communication».

11.2 Étude préalable

Elle s'applique uniquement au niveau d'un domaine. Elle doit être complète, mais suffisamment rapide (c'est contradictoire). Pour cela, on va travailler sur un sous-ensemble représentatif, en négligeant les procédures exceptionnelles ou peu fréquentes.

L'étude préalable se fait sans *a priori*. L'analyste essaie de comprendre le fonctionnement de l'entreprise ou du service avec lequel il doit interagir. Le but de l'étude est de donner à un problème posé une ou plusieurs solutions, sommairement esquissées, afin de pouvoir prendre une décision quant à la solution réellement adoptée. L'une des hypothèses plausibles est la non-faisabilité, qui aboutira à renoncer au projet. La difficulté de l'étude préalable réside dans la nécessité de mener à la fois une étude complète, faute de quoi la solution retenue risque d'être mal adaptée, et une étude partielle, dans la mesure où ce n'est qu'une étude préalable qui sera suivie par un travail plus en profondeur. Il n'existe pas de technique permettant d'éviter l'écueil. L'étude préalable doit évidemment prendre en compte le schéma directeur. Cette étude préalable se compose de quatre phases successives :

- le lancement
- l'analyse de l'existant
- la conception des solutions
- l'évaluation des solutions

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

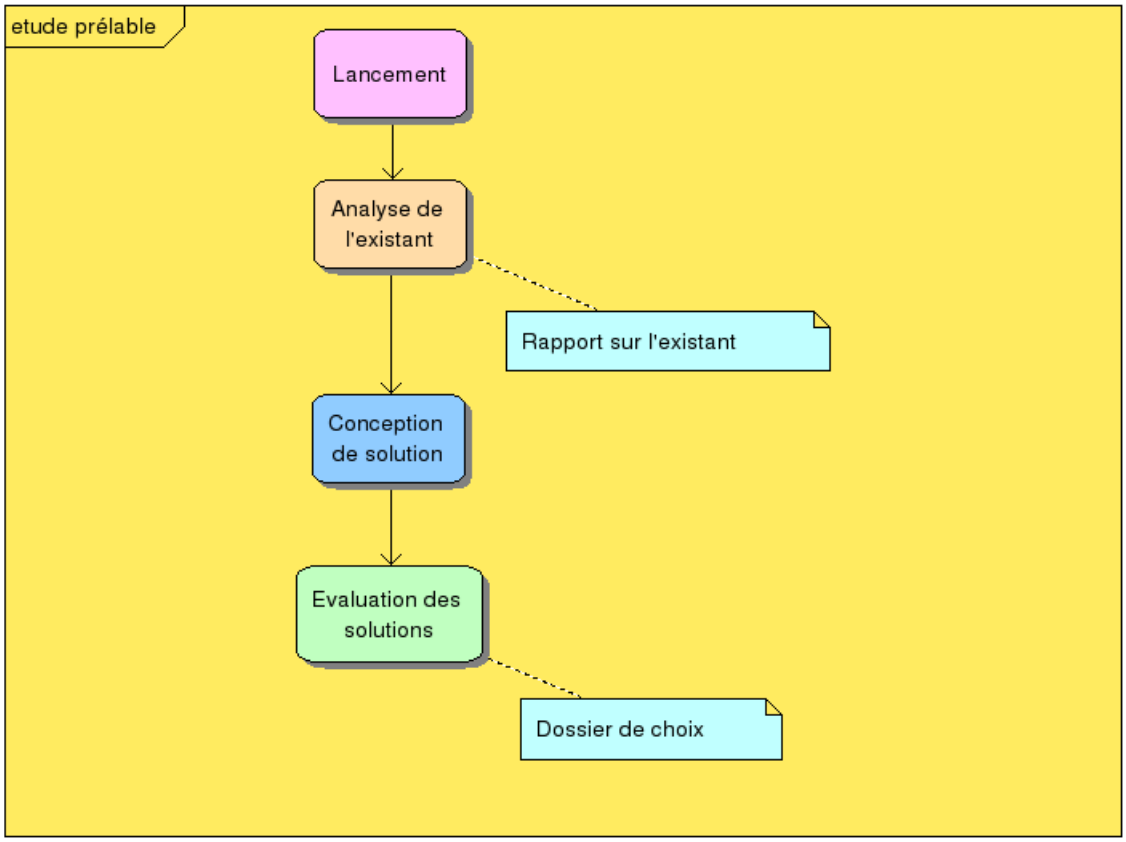
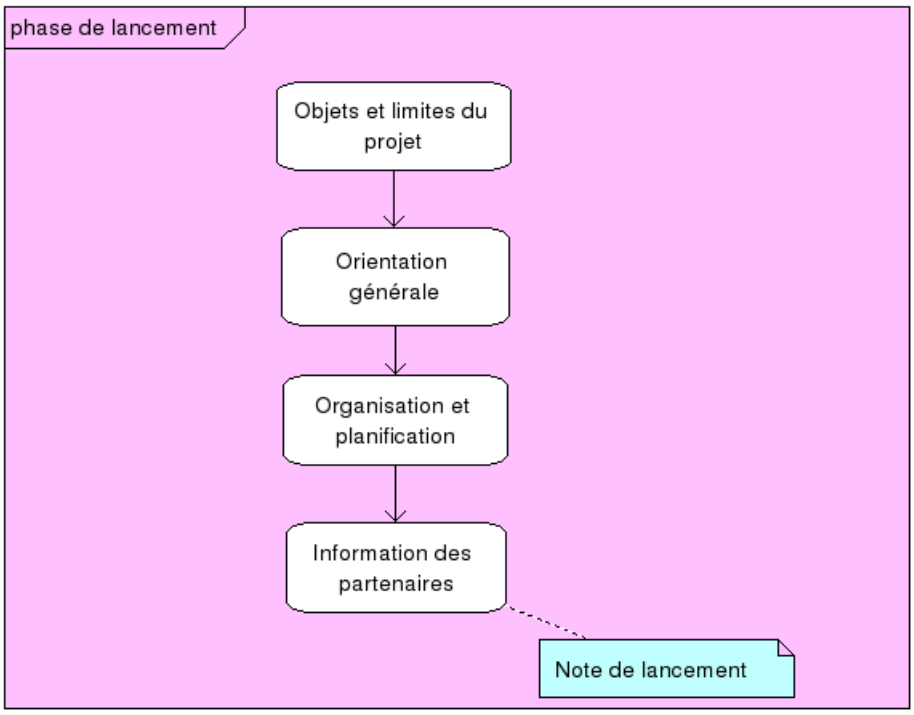


FIG. 11.1 – Étude préalable

11.2.1 Phase de lancement



©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

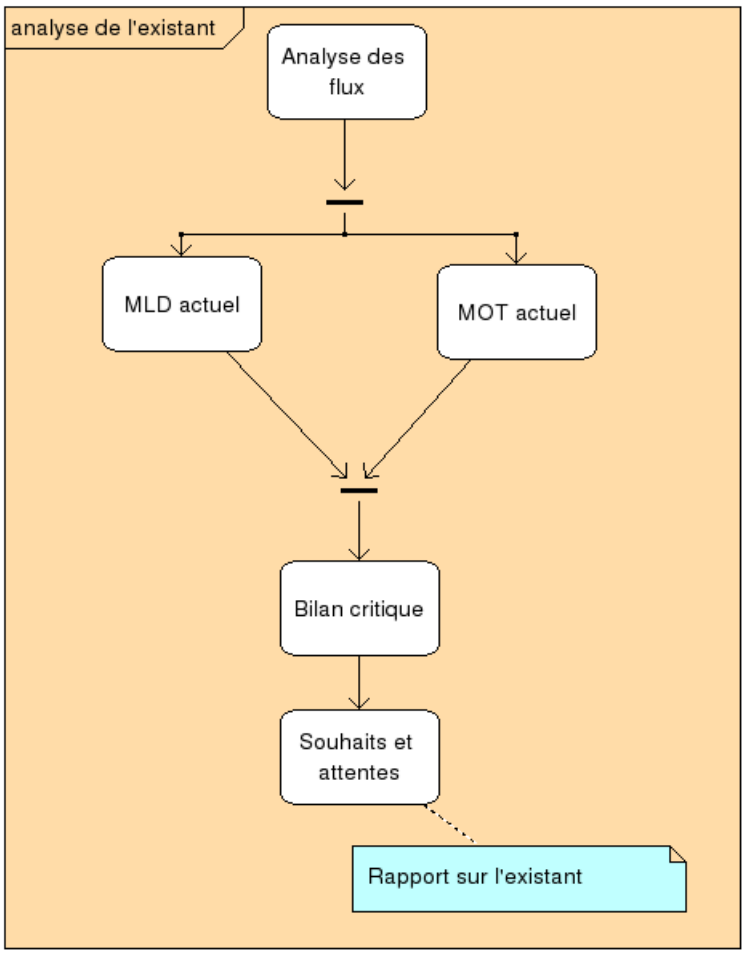
La phase de lancement précise les limites (en temps, en objet et en ressources humaine). Elle planifie les structures de contrôle, répartit les tâches, fixe un cadre budgétaire.

1. Objets et limites du projet : le projet est situé dans son contexte et par rapport au domaine qu'il concerne.
2. Orientation générale : description de ce que le projet est censé faire.
3. Organisation et planification : on précise les ressources humaines (personnes à contacter), on spécifie également un planning et des réunions régulières destinées à évaluer l'avancement du projet.
4. Information : à ce stade l'information doit se présenter sous une forme synthétique afin que toutes les parties prenantes disposent d'un document de référence utile pour réaliser toute l'analyse.

Le travail de lancement se termine par un document interne présentant une synthèse utilisable de la première approche. Il est destiné aux collaborateurs afin de leur donner une vue générale. Les commanditaires peuvent également s'en servir pour valider la description donnée de la problématique.

11.2.2 Phase d'analyse de l'existant

Souvent excessivement développée, cette phase consomme trop de ressources et jette le discrédit sur la méthode.



©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

1. Analyse des flux : mettre en évidence les acteurs concernés et les flux d'information, dans une vision structurelle. Ce schéma présente une importance capitale puisque tous les traitements réalisés par le projet doivent s'articuler sur cette description.

2. Analyse du modèle organisationnel actuel : on identifie les différents traitements réalisés, en prenant en compte l'existence des différents postes de travail. On veillera à ne pas détailler trop ces traitements. Dans certains cas simples (un seul poste de travail), le MOT apporte peu d'éclaircissement par rapport au diagramme des flux.

3. Analyse du modèle logique des données actuel : les fichiers informatiques sont analysés et quantifiés (volume, fréquence de création, modification).

4. Bilan critique de la situation actuelle : comparer points faibles et forts, les confronter avec les attentes. Mise en évidence des dysfonctionnements techniques (matériel non adapté, obsolète ou absent), organisationnels (lourdeurs, insatisfactions ou procédures « occultes ») et de gestion (en quoi le système est-il en décalage par rapport aux choix de gestion ?).

5. Souhaits et attentes à satisfaire : tant au niveau des gestionnaires que des utilisateurs finaux.

Un document, le rapport sur l'existant, permet de valider le travail et de le confronter aux avis des différentes parties prenantes.

11.2.3 Phase de conception des solutions

1. Objectifs et contraintes du futur système : prise en compte des demandes (avec mise en évidence des contradictions), reformulation et synthèse. Détermination de sous-objectifs.

2. Construction des MCD et MCT futurs : en général le MCD futur est proche du MCD actuel. Le MCT est construit en prenant en compte le schéma directeur et le MOT observé, dans lequel on va faire abstraction des ressources. Note : pour le modèle de données, il n'est pas nécessaire de préciser toutes les propriétés.

3. Présentation des MC futurs : la présentation permet de vérifier que les modèles conceptuels concordent avec les objectifs des gestionnaires. Leur approbation est nécessaire pour passer à la phase suivante.

4. Construction de MOD et MOT futurs : niveau fondamental en étude préalable. On va affecter des ressources au MCT défini précédemment (postes, tâches, phases). Spécifier pour le MOT ce qui doit être mémorisé (éliminer tout ce qui sera traité manuellement).

5. cohérence MOD/MOT : la cohérence ne s'attache pas aux détails (notamment, toutes les propriétés ne sont pas encore définies). De nombreux problèmes peuvent se déceler à ce niveau.

6. Synthèse et validation : on reprend les solutions qui viennent d'être dégagées et on les reformule dans des termes accessibles aux gestionnaires.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

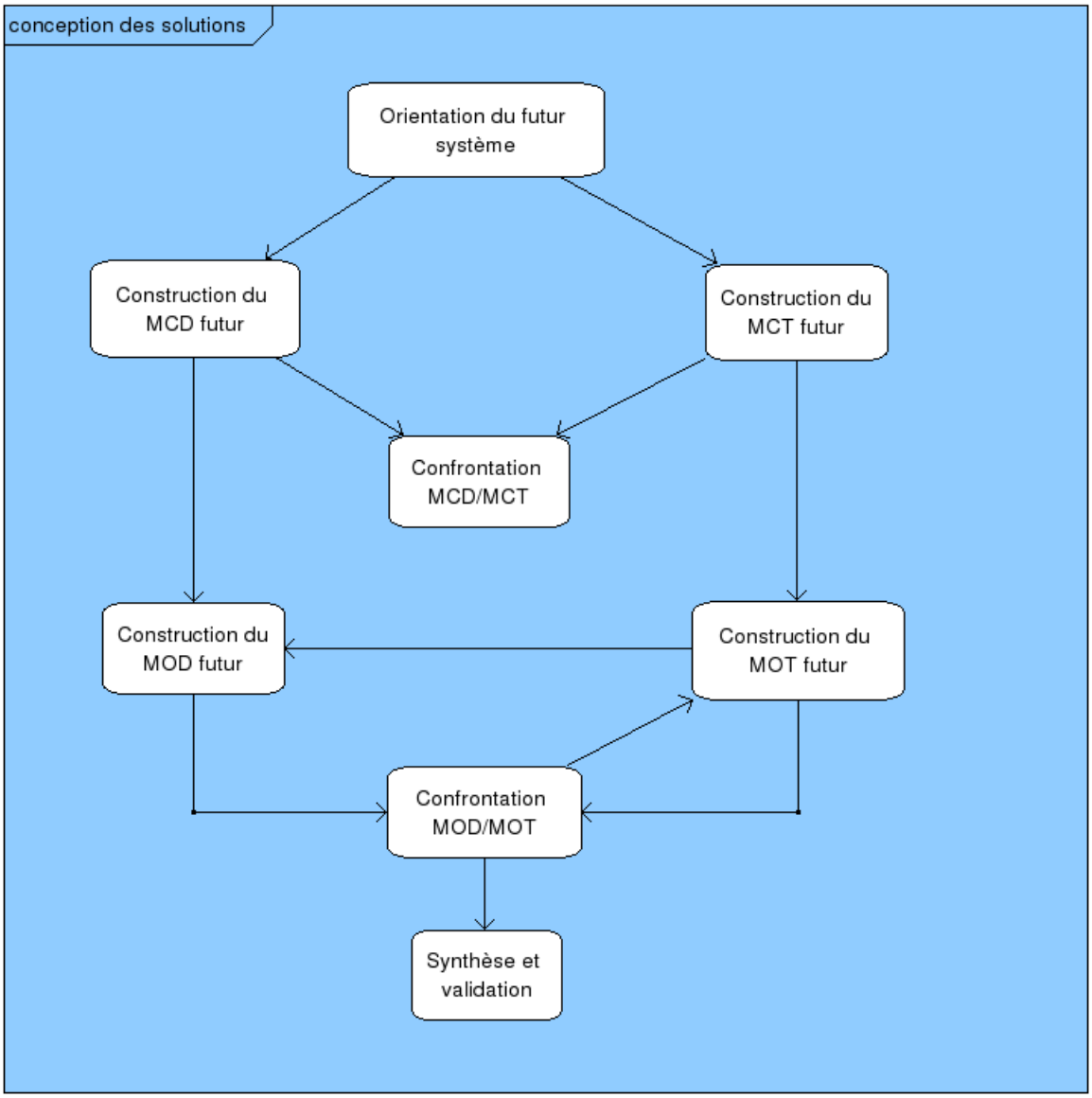
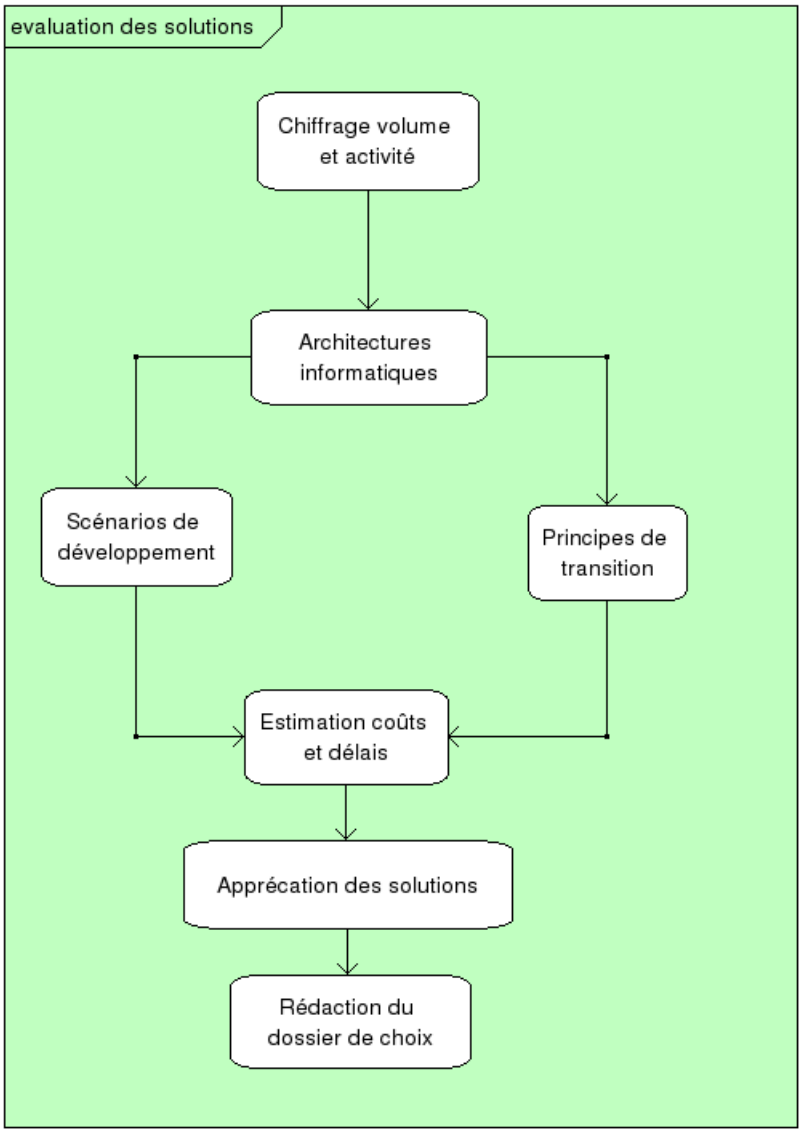


FIG. 11.2 – Conception des solutions

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

11.2.4 Phase d'évaluation des solutions



1. Chiffrage du volume et de l'activité : essentiellement destiné à permettre l'étape suivante : on va tente de chiffrer la taille des données, l'intensité et la fréquence des trafics d'informations.
2. Précision des architectures informatiques : on définit les besoins en termes de nombres de « terminaux » et serveurs, capacités requises (mémoire et traitement), logiciels de base (OS, SGBD).
3. Principe de transition : dès ce stade, il faut prévoir comment passer du système actuel au nouveau système. Il n'est pas concevable de fermer l'entreprise pendant la mutation. Pour des gros projets, il faut prévoir une installation progressive. Parfois un fonctionnement en double s'impose pendant une période.
4. Scénario de développement : on évalue les ressources humaines et matérielles du développement, on prévoit éventuellement des sous-projets et des équipes pour les réaliser.
5. Estimation des coûts et délais : ces deux points sont sans doute les plus délicats. Il s'agit de les évaluer correctement sous peine de voir le projet engloutir des fortunes, non prévues initialement.

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

6. Appréciation des solutions : il faut pouvoir convaincre les décideurs de ce que la solution proposée, avec ses coûts, apporte une amélioration à la situation initiale.
7. Rédaction du dossier de choix : le dossier de choix reprend sous une forme compacte, la solution ou les solutions envisagées et doit permettre aux décideurs d'accepter l'une des solutions.

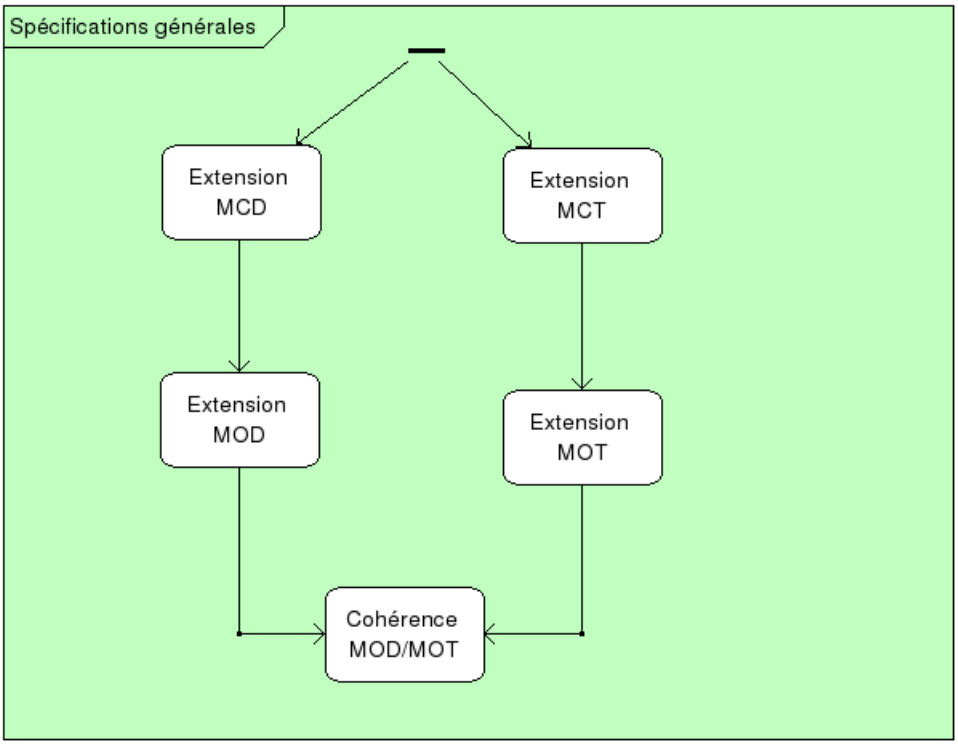
A ce stade une décision va intervenir :
- on choisit une des solutions
- on demande un complément d'étude
- on reporte ou abandonne le projet.

11.3 Étude détaillée

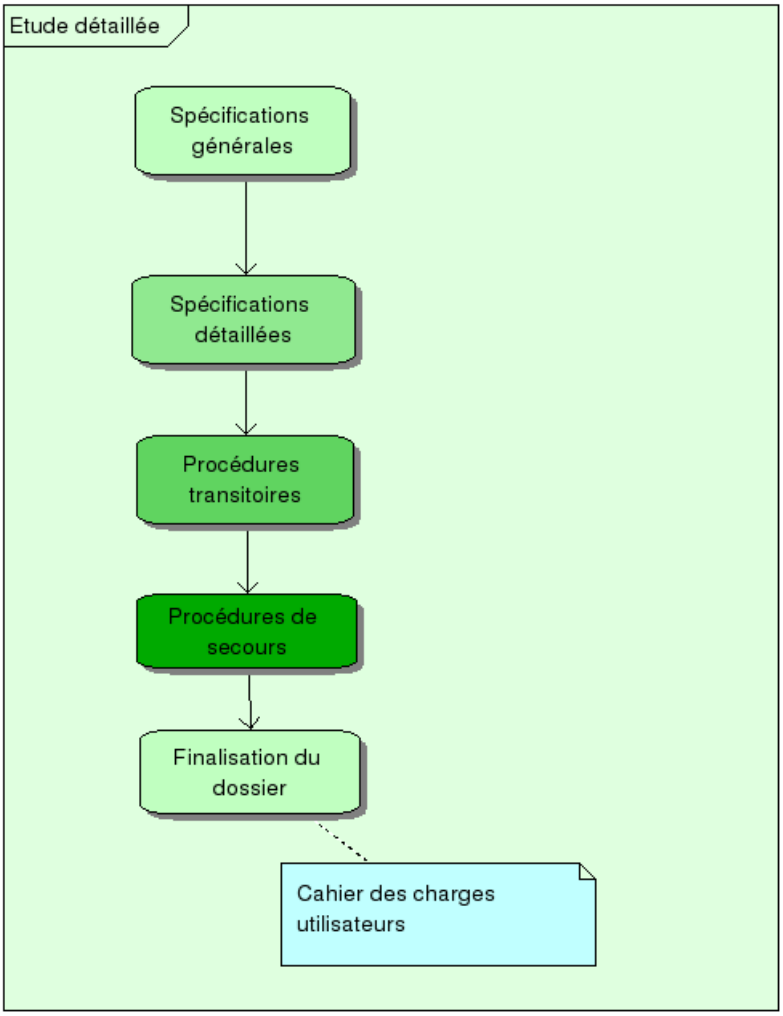
Elle concerne la solution retenue et entreprend de donner les éléments permettant la réalisation du projet.

L'étude détaillée correspond chronologiquement à ce que d'autres auteurs appellent l'*analyse fonctionnelle*, qui spécifie le *quoi faire*. Dans l'optique de Merise, on préfère mettre l'accent sur l'étude du système d'information, considéré en soi, et indépendamment de l'outil informatique. C'est dire qu'à ce niveau les dirigeants de l'entreprise ont un rôle capital à jouer, puisque maintes décisions vont devoir être prises. Elle se compose de cinq phases :
- phase de spécifications générales
- phase de spécifications détaillées
- phase de spécifications des procédures transitoires
- phase de spécifications des procédures de secours
- phase de finalisation du dossier

11.3.1 Phase de spécifications générales



©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET



1. extension du MCD : reprise du modèle de l'étude préalable, mais cette fois, il est complété par tout ce qui a été mis de côté (notamment la liste exhaustive des propriétés).

2. extension du MCT : ici encore, on ajoute les traitements qui avaient été écartés, comme marginaux ou rares.

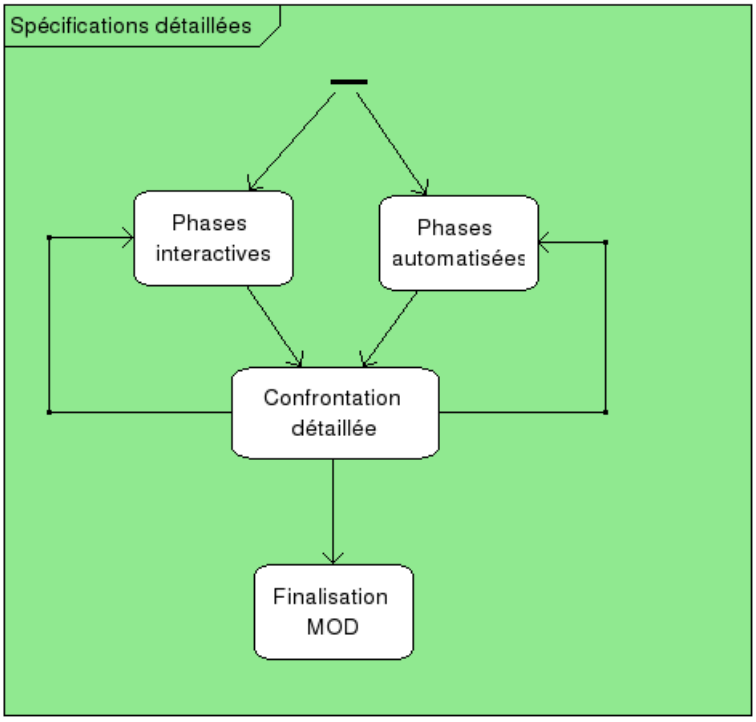
3. extension du MOD : Il tient compte des extensions du MCD et de ce qui est exigé par la description plus précise du MOT (MOD locaux).

4. extension du MOT : On entre complètement dans les détails des tâches, des postes, de leur répartition entre l'homme et la machine.

5. une vérification de la cohérence des MOD et MOT doit se faire à la fin de cette phase.

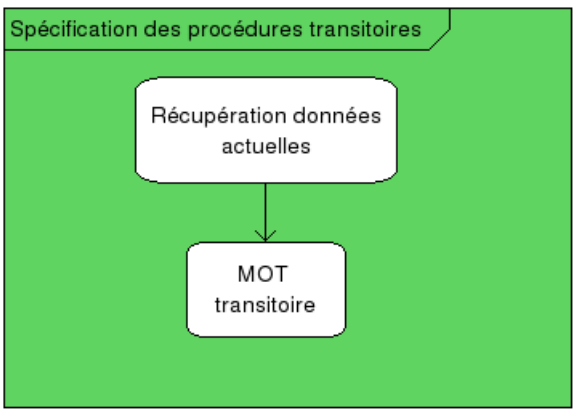
©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

11.3.2 Phase de conception détaillée



1. tâches conversationnelles et automatisées : Pour les tâches conversationnelles, grand luxe de détails, notamment dessin des écrans et des états, précision des différents contrôles de cohérence des données, spécification des algorithmes, action sur les données, messages d'erreur... Pour les tâches automatisées, on donnera les mêmes précisions, mais sans devoir tenir compte des terminaux.
2. validation détaillée : Il s'agit de vérifier la cohérence entre les deux modèles, notamment par l'utilisation des sous-schémas de données.
3. enrichissement du MCD et du MOT : la validation a révélé des manques et on corrige les modèles de manière à les rencontrer.
4. finalisation du MOD : le MCD final est transformé en MOD.
5. révision des estimations et du scénario : au vu de ce qui a été spécifié, le scénario et les estimations sont revus.

11.3.3 Phase des procédures transitoires

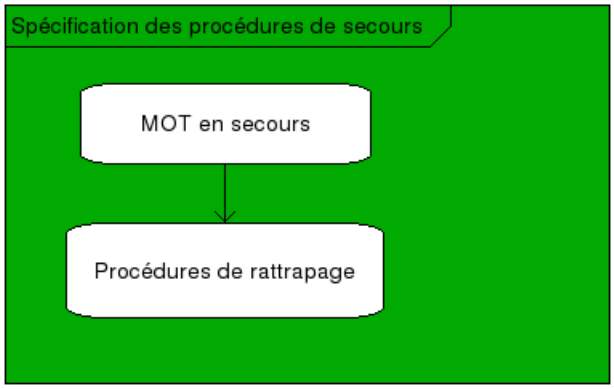


©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Cette phase concerne les projets qui remplacent des solutions existantes. Ils nécessitent des traitements de modifications des données, avec éventuellement un étalement dans le temps et différentes phases.

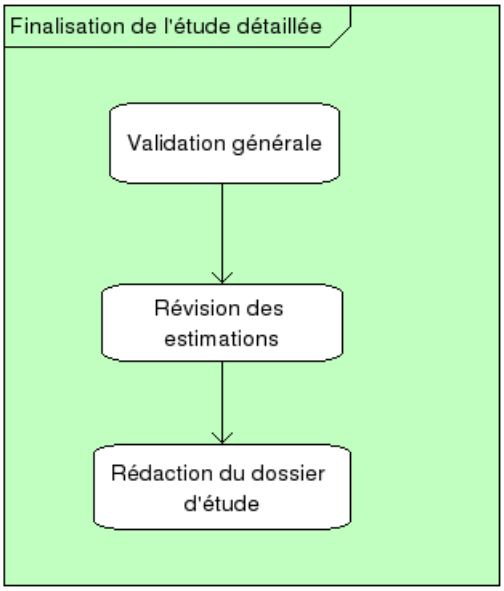
1. procédures de conversion et/ou saisie et chargement
2. MOT transitoire : si la transition nécessite un travail de longue haleine, il peut s'accompagner de programmes de conversion qui devraient faire l'objet d'une analyse.

11.3.4 Phase de procédures de secours



1. MOT pendant la défaillance : prévision d'un modèle appauvri en cas de panne (éventuellement sans informatique).
2. Procédures de rattrapage : comment faire pour réintroduire les données perdues pendant la panne ?

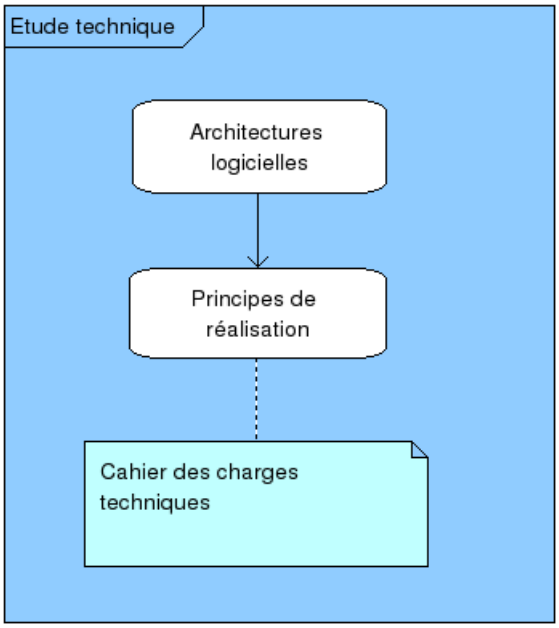
11.3.5 Finalisation et production du cahier de charge utilisateur



La nature contractuelle de ce document est fondamentale. Il faut donc qu'il y ait un accord sur ce qui va être fait. Dans certains contextes, un cahier de charge mal conçu peut entraîner des déboires financiers et/ou des procès. Il faut parfois aussi se prémunir contre la mauvaise foi du client.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

11.4 Étude technique



Le stade suivant, parfois appelé *analyse organique*, serait du domaine du *comment faire*. Dans la perspective de Merise, il faudrait reformuler autrement : avec quels moyens informatiques. A ce stade, l'avis du client n'est plus pris en compte que pour des choix budgétaires, l'étude détaillée a dû normalement lui permettre d'effectuer les choix qui lui importaient. L'étude technique est souvent réalisée par l'équipe qui assure le développement.

11.4.1 Conception logique de l'application

1. conception du MLD et du MLT : On va tenir compte des technologies disponibles pour définir, par site, le MLD. On tient compte de l'architecture du matériel, de la répartition organisationnelle et des contraintes de confidentialité et de sécurité. Pour le MLT, on parlera d'états et d'écrans, d'unités logiques de traitement, provenant d'une décomposition logique des tâches du MOT. Notons que les écrans eux-mêmes peuvent faire l'objet de décompositions en éléments réutilisables (mêmes fonctionnalités).
2. cohérence entre données et traitement avec retour sur l'étape antérieure : Les sous-schémas conceptuels trouvent ici leur équivalent logique, permettant de vérifier la cohérence des deux approches MLD et MLT.
3. optimisation du MLD : Certaines contraintes entraînent la nécessité de revoir le MLD en vue d'une optimisation des traitements.

11.4.2 Spécifications physiques du logiciel

1. spécification des fichiers ou bases de données (MPD) : Les données sont cette fois décrites de manière opérationnelles, directement exploitable (par exemple en langage SQL).
2. spécification des modules d'accès : Selon le cas, il s'agira de sous-programmes, de bibliothèques à inclure, d'objets...

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

3. architecture physique des programmes transactionnels et batch : largement tributaire de l'environnement choisi, celle-ci fera appel à toutes les ressources disponibles : objets dynamiques et statiques, spécifications des transactions, grilles de dialogues,...

11.4.3 Préparation de la réalisation

Il s'agit de l'élaboration d'un cahier des charges destiné au développeur, reprenant tous les éléments utiles pour la réalisation du programme.

- préconisation des outils de réalisation
- règles de programmation et de documentation (structuration du code, nommage et commentaire, documentation du code)
- principes de qualification (tests unitaires et d'intégration, jeux de données, gestion des versions)
- organisation et planification de la réalisation (répartition entre équipes, planning de production)

11.4.4 Production du cahier de charges réalisateur

Ce document, qui n'est pas destiné au client, contient en fait tout ce qui ce qui doit être réalisé. Dans certains cas, la réalisation d'un logiciel sera faite par une équipe appartenant à une autre entreprise (c'est le cas de certains sous-traitants de pays du tiers monde). Il faut qu'à ce stade le document produit soit auto-suffisant.

11.5 Réalisation du logiciel

Ce point ne concerne plus l'analyse. Nous le laisserons donc sous une forme schématique.

- Organisation production
- Écriture des programmes et documentation
- Qualification des codes unitaires
- Intégration du logiciel
- Qualification interne

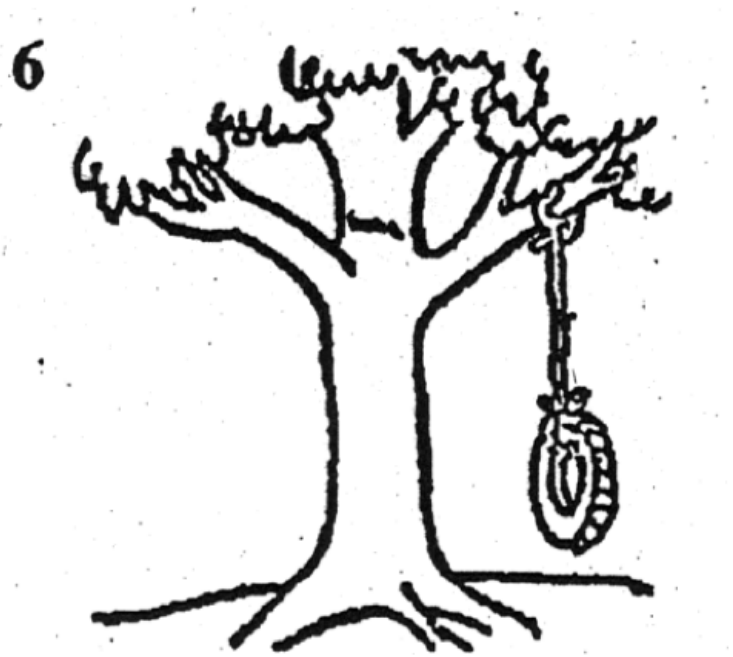
11.6 Mise en service

- Planification de la mise en service
- Mise en place des ressources
- Préparation du lancement
- Mise en exploitation
  - recette avant lancement
  - lancement effectif
  - période probatoire
  - recette définitive

11.7 Maintenance

- Prise en compte d'une demande
- Décision
- Réalisation de la modification

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET



Ce qu'il fallait

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Chapitre 12

La méthode rapide de Merise

Je suggère ici un retour sur une méthode rapide inspirée de Merise. Il pourra aider lors de la rédaction des mémoires[1] .

12.1 Présentation

12.1.1 Nécessité d'une nouvelle méthodologie

Au cours du temps, la méthode Merise, conçue initialement dans le courant des années 80, s'est vue critiquée et remaniée, en fonction de l'évolution des techniques. Parallèlement à des évolutions récentes[2], le besoin s'est fait sentir d'une démarche simplifiée pour s'appliquer dans des cas non prévus par la méthode initiale. Cette démarche alternative est venue répondre à des besoins nés dans des projets mettant en oeuvre

1. de nouveaux environnements :
   - le développement de la micro-informatique et spécialement des interfaces graphiques « à la Windows » ;
   - les SGBD relationnels
   - l'émergence d'Internet et des intranets.
2. de nouvelles exigences :
   - implication des utilisateurs
   - équipes plus compactes
   - maîtrise des coûts.

Parmi les nouveaux concepts récemment émergés, il convient de souligner le *RAD*. Le concept de *Rapid Application Development* implique de nouveaux outils spécifiques (Visual BASIC et certains outils de Visual Studio .net, Delphi, Kylix, C++Builder, Power Builder, Windev, pour ne citer que les plus courants) et une nouvelle manière de travailler. Les utilisateurs sont plus impliqués que dans d'autres projets, ils n'ont pas nécessairement une vision claire de leurs besoins et ils ont un budget limité (on fera donc ce qu'il est possible de réaliser sans sortir d'une enveloppe).

[1] Mes notes s'inspirent de la lecture du chapitre 16 de la 4me édition de *Ingénierie des système d'information : Merise* de NANCY, ESPINASSE et autres (Éditions Vuibert 2001) consacré à « La démarche rapide de Merise ».

[2] La méthode présentée dans mon cours d'*Analyse, principes et méthodes*, se réfère à la version finale et actuelle de Merise.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

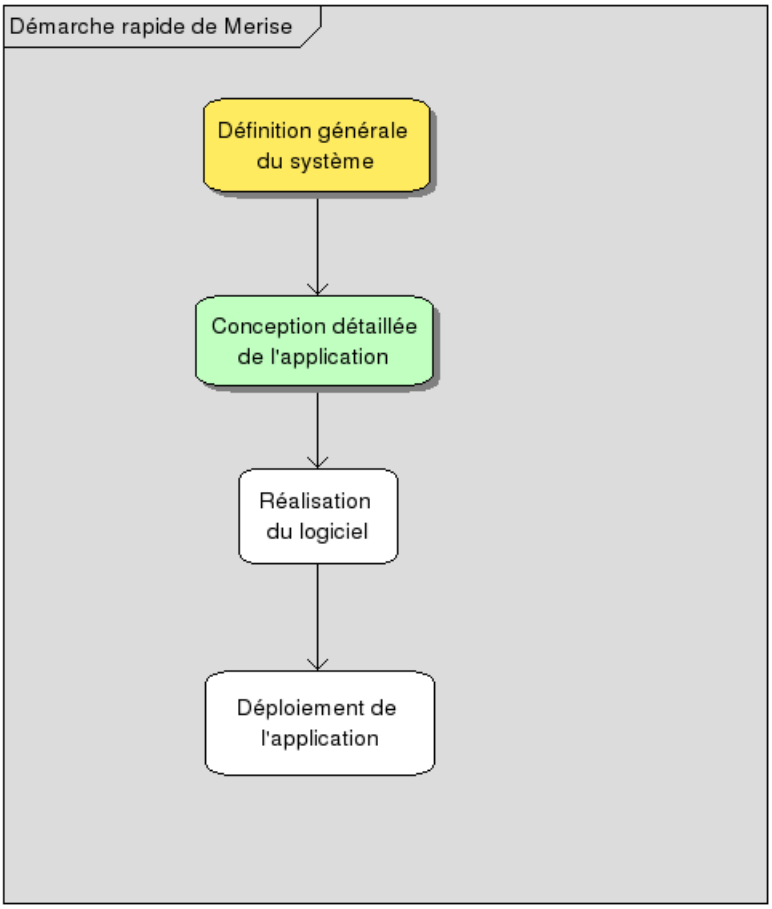


FIG. 12.1 – Modèle simplifié et phases de la définition générale du système

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

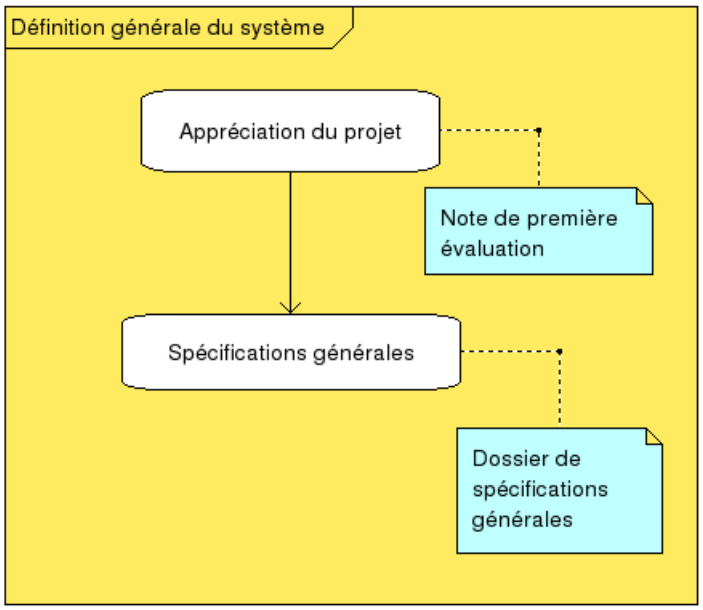
12.1.2 Caractéristiques générales

- La démarche rapide ne renie pas les techniques employées par la démarche classique, elle s'en inspire donc fortement.
- L'utilisation des méthodes de maquettage/prototypage favorise la définition des spécifications, en collaboration avec les utilisateurs (éviter l'effet tunnel).
- Les démarches issues du RAD débouchent souvent sur une approche itérative de la conception, notamment par la livraison de versions successives, de plus en plus riches en fonctionnalités. Cette démarche doit rester distincte de la démarche exploratoire, parfois employées dans des projets de recherches. Si, extérieurement, et notamment pour l'utilisateur, l'évolution du programme semble manifester un certain empirisme, la méthode doit s'appuyer sur des modèles clairement définis. Procéder autrement débouchera forcément sur un système instable et incapable d'évolution à long terme.

12.2 Étapes

12.2.1 Définition générale du système

Il s'agit ici de mesurer l'étendue et la complexité du projet (voire sa faisabilité en terme de budget et d'applicabilité d'une démarche rapide). On distingue deux phases, produisant chacune un document final.



Phase d'appréciation du projet (→ note de première évaluation)

Obligatoirement courte, cette première phase rappelle l'étude préalable de la méthode classique. Elle définit les caractéristiques générales du projet dans son environnement et étudie sa faisabilité. Le document final est destiné au maître d'ouvrage.

1. positionnement du projet : on tâchera de répondre à des questions concernant l'étendue et la complexité du projet, ses objectifs, la bonne connaissance du domaine, la disponibilité des utilisateurs, l'existence de logiciels équivalents.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

2. délimitation des fonctions à informatiser : pour déterminer plus facilement le périmètre fonctionnel du projet, on pourra faire usage de deux diagrammes des flux : l'un montrant les échanges du système avec les acteurs de son environnement, l'autre explicitant le fonctionnement général du système.
3. évaluation des enjeux : tous les points repris précédemment font l'objet d'une note de synthèse permettant d'évaluer les enjeux du projet, mis en relation avec la demande initiale. Notons qu'à ce stade, il n'est pas encore question de proposer de solution.

Phase de spécifications générales (→ dossier de spécifications générales)

Il s'agit de spécifier définitivement :
– les données à mémoriser ;
– les fonctions à informatiser ;
– les éventuelles interactions avec d'autres systèmes.

Le dossier résultant reprendra une synthèse du document précédent plus les deux modèles qui vont être cités ainsi qu'une évaluation des besoins en matériel et en logiciel.

On proposera essentiellement deux modèles :

1. Modèle conceptuel des données : il est hors de question de bâcler le travail à ce niveau (la qualité de la base de données en dépend). On peut éventuellement omettre provisoirement quelques propriétés.
2. Macro-modélisation organisationnelle des traitements : l'allégement par rapport à la méthode complète portera sur les points suivants :
   – ne pas nécessairement répartir les traitements en postes de travail
   – accentuer l'examen des phases (niveau de détails peu élevé)
   – ne pas donner trop de détails.

12.2.2 Conception détaillée de l'application

Il est possible d'envisager une conception détaillée suivie immédiatement d'une réalisation pour chaque module de l'application. L'utilisateur sera fortement impliqué dans l'évaluation de la conception grâce à la technique du maquettage. Le but de la maquette est de permettre une définition précise du contenu et de la forme de l'application à livrer. Chaque maquette sera montrée aux utilisateurs et révisées en fonction de leurs remarques, qui pourront porter sur ses fonctionnalités ou son ergonomie/aspect.

On pourra distinguer deux types de maquettes : une maquette jetable, réalisée avec un logiciel léger, soit une maquette réalisée avec l'environnement de développement final, mais ayant des fonctionnalités incomplètes ou simulées. Chaque méthode a évidemment ses avantages et ses inconvénients.

Phase de développement de la maquette

1. modélisation des données : la maquette s'appuie évidemment sur un modèle logique. On veillera cependant dans cette phase de conception à garder un modèle conceptuel comme référence de base pour procéder à des modifications et/ou des enrichissements.
2. conceptions des dialogues : les différents dialogues seront réalisés dans la maquette. On veillera à bien distinguer trois aspects :
   (a) *design* de la présentation : l'utilisation des différents éléments du dialogue doit évidemment correspondre aux fonctionnalités désirées et appliquer les règles de l'ergonomie.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

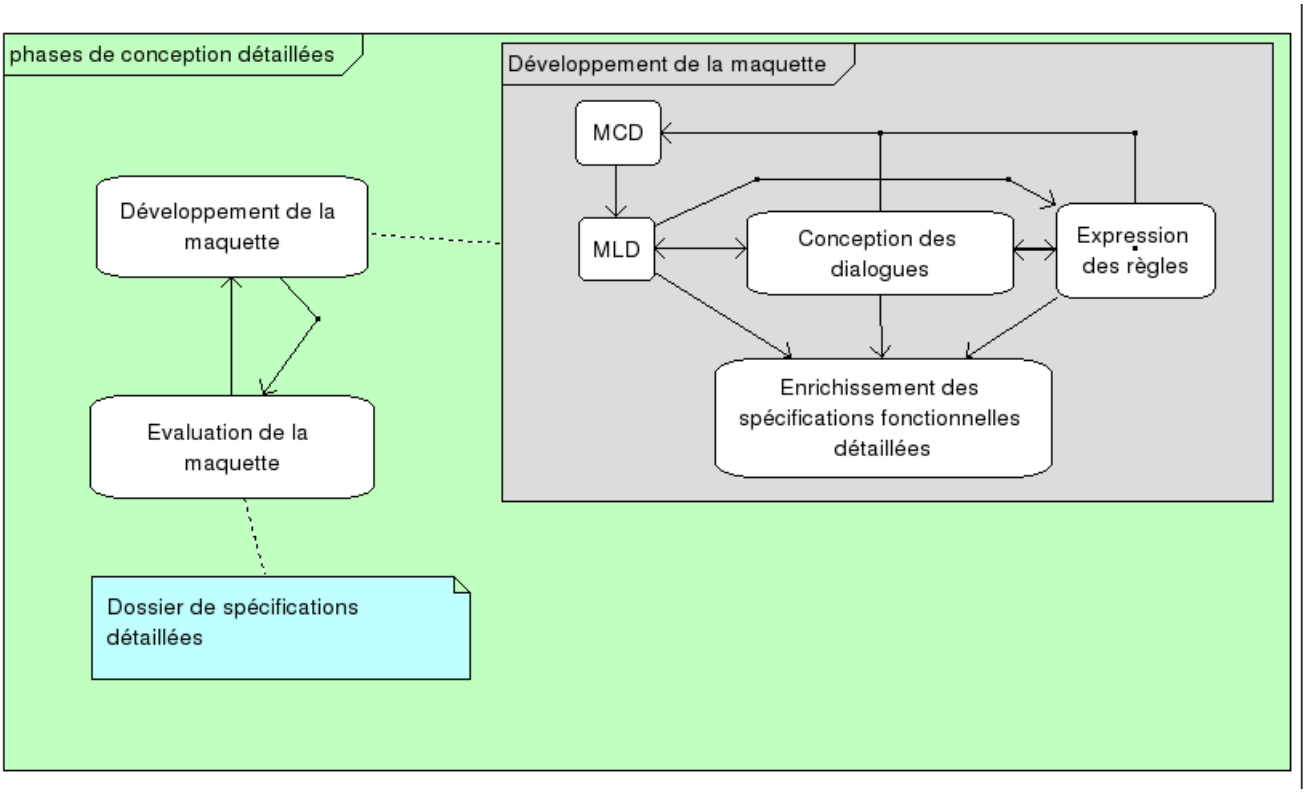


FIG. 12.2 – Modèle simplifié et phases de la conception détaillée

(b) expression de la logique du dialogue : à l'intérieur du dialogue, des valeurs calculées, des contraintes, des séquences doivent se définir.

(c) définition des enchaînements : à l'aide de boutons et menus, on passera d'une fonctionnalité à l'autre[3].

3. Expression des règles : toutes les règles de gestion doivent être explicitées. L'intégration dans une maquette n'est pas toujours possible, ni directement testable. On les présentera à l'aide d'un formalisme précis, permettant une lecture non ambiguë par le client (ce qui exclut un langage de programmation).

4. Enrichissement des spécifications fonctionnelles détaillées : la maquette ne peut pas tout intégrer. Il est donc tout à fait nécessaire de faire valider les modèles de données et de traitements par le client (pas nécessairement l'utilisateur final). Un dossier devra donc rassembler toutes ces informations.

Phase d'évaluation de la maquette

1. présentation de la maquette : le contexte de réalisation de cette présentation conditionne fortement la fiabilité des enseignements qu'on peut en tirer. La maquette sert à présenter le produit et ses spécifications, non à tester le futur programme. L'utilisateur doit être mis au courant de la finalité de cette présentation et du contexte futur d'utilisation de la fonction testée (test modulaire).

[3]L'expérience montre que c'est un point difficile de la conception : beaucoup de programmes présentés à l'épreuve intégrée font l'impasse sur la définition des enchaînements et on se trouve face à une succession d'écrans plus qu'à une véritable application. On peut considérer à ce moment que les enchaînements sont contrôlés par l'utilisateur et non pas le programme, avec les dysfonctionnements que cela peut entraîner.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

2. formalisation des amendements : les remarques, critiques et demandes doivent être scrupuleusement notées et prises en compte dans des versions successives (si des changements importants sont introduits, ils nécessitent une nouvelle présentation).

12.2.3 Réalisation du logiciel

La réalisation du logiciel n'apporte pas de révolution par rapport à la démarche traditionnelle de Merise. Il faut néanmoins apporter quelques évolutions rendues nécessaires par les modifications de la conception :

- la prise en compte de la maquette dans le processus de réalisation (soit par une simple reprise fonctionnelle si l'environnement de développement est différent, soit une récupération pure et simple, avec affinement et remplissage des fonctionnalités laissées à l'état d'ébauche).
- adoption d'une démarche en spirale : on travaille par modules avec à chaque fois un cycle conception/réalisation.

Deux tâches de cette étape sont particulièrement importantes : l'optimisation de la base de données et la prise en compte des spécificités de l'environnement.

12.2.4 Déploiement de l'application

Cette partie de la démarche ne diffère pas tellement de la démarche classique, si ce n'est par son ampleur moindre. On fera si possible un déploiement progressif (par installation sur des sites pilotes), en prenant soin de tenir compte des réactions des premiers utilisateurs et sans négliger la phase d'apprentissage et de réalisation de la documentation[4], avant la réception définitive.

[4]Une partie de la documentation sera intégrée au logiciel, grâce aux possibilités offertes par l'interface. Cette obligation de « coder » la documentation prouve qu'elle doit être envisagée dès la phase de conception et non laissée pour la fin, si on en a le temps.

©2008-2009 -JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Chapitre 13

Cycle itératif et incrémental

On constate souvent qu'un système qui fonctionne constitue l'évolution d'un autre système qui fonctionnait déjà. Cela suppose des capacités d'évolution dont les logiciels, par nature fragiles, se montrent souvent incapables. On peut cependant utiliser une approche objet qui favorise l'évolution[1].

13.1 Cycle de vie linéaire

Dans cette optique, le développement d'un projet s'envisage comme la réalisation d'une série d'étapes qui mènent de l'élaboration du cahier de charges à la réalisation. Le *modèle en tunnel* en est la caricature : on part d'une demande initiale formulée rapidement, on poursuit avec un développement sans concertation avec les utilisateurs et on termine par la livraison, si tout va bien, d'un logiciel totalement inutilisable. Le *modèle en cascade*, qu'on retrouve par exemple dans la méthode Merise, propose des possibilités d'évaluation et de concertation à différents niveaux.

Malheureusement, la seule évaluation vraiment pertinente ne peut se faire qu'au terme du codage (les autres évaluations se font toujours sur des documents écrits). Il arrive donc fréquemment que les tests fonctionnels obligent à revenir sur la phase d'analyse, ce qui entraîne la nécessité de relancer les processus de conception et de codage. A ce moment, le modèle n'est plus conforme à la pratique du terrain.

13.2 Cycle de vie itératif

Le cycle de vie itératif suppose une implication de l'ensemble des participants. Il s'agit de reprendre les quatre opérations fondamentales ANALYSE-CONCEPTION-CODAGE-INTÉGRATION *n* fois jusqu'à l'obtention d'un résultat satisfaisant pour toutes les parties. Loin d'encourager la « bidouille », ce procédé peut donner des résultats souvent meilleurs que les méthodes linéaires par l'existence de réalisations concrètes, qui encouragent aussi bien l'utilisateur que le développeur. Le mot d'ordre est la progressivité.

On va donc reproduire le modèle en cascade, mais avec chaque fois une planification précise reprenant :

- les objectifs
- les tests à réaliser pour évaluer que les objectifs ont été atteints.

[1] Ce paragraphe reprend pour l'essentiel des idées de l'exposé du chapitre 4 de Pierre-Alain MULLER, *Modélisation objet avec UML*. Éditions Eyrolles.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET

Dans un premier temps, on va partir des cas d'utilisation simple, en proposer la réalisation et l'évaluation. Le travail débouche sur la livraison d'un prototype. Sur base de la réaction des utilisateurs, on va peu à peu ajouter des fonctionnalités nouvelles et corriger les fonctionnalités dont la réalisation présentait des défauts à l'itération précédente.

Il est vrai que cette méthode de travail paraît reproduire la méthode essais-erreurs tant décriée. Il ne faut cependant pas s'y tromper, ce n'est pas la bidouille à laquelle nous avons tous plus ou moins succombé un jour :

- les objectifs sont clairement définis, ainsi que les moyens de vérifier s'ils sont atteints ;
- on part toujours des cas d'utilisation les plus importants, de manière à assurer en premier les fonctions vitales du projet ;
- les versions successives ne remettent pas en question le travail réalisé précédemment (sauf pour ce qui concerne la correction des erreurs relevées) : elles le complètent par l'ajout de fonctionnalités supplémentaires, elles en poursuivent l'implémentation pour des fonctions qui n'avaient été que simulées[2] ou elles en traitent les cas particuliers mis tout d'abord de côté.

Souvent décrié, le cycle de vie itératif a le mérite de concilier théorie et pratique de terrain. L'expérience montre en effet que les modèles en cascade sont rarement appliqués fidèlement.

13.3 La suite...

En abordant cette méthodologie, nous quittons l'analyse classique et la programmation fonctionnelle. Ces principes sont mis en œuvre dans les méthodes d'analyse appliquées à l'objet. Nous les retrouverons dans différentes approches telles que le *Rational Unified Process* des *tres amigos* qui ont influencé la définition du langage UML ou la méthode *Two Track Unified Process* de Pascal Roques. Il reste beaucoup à apprendre et un autre cours a heureusement été prévu.

[2]On peut parfaitement, dans un premier prototype, simuler des fonctions importantes comme la mémorisation des données. En effet, l'utilisateur n'est pas vraiment concerné par la gestion des bases de données. Cela peut sembler une perte de temps d'implémenter un modèle de persistance tant que le modèle des données n'est pas validé du point de vue de l'utilisateur. Cela suppose évidemment une structuration claire de l'application. L'affichage et le traitement des données sont totalement indépendants de l'archivage.

©2008-2009 - JACQUES THOORENS - HTTP://WWW.THOORENS.NET