

Quelle place pour la maintenance aujourd'hui ?

Alors que les entreprises viennent de vivre avec l'an 2000 et l'Euro les deux plus grandes opérations de maintenance que l'informatique ait jamais connues, le thème de la maintenance n'occupe qu'une place modeste dans la presse spécialisée et les séminaires professionnels. Faut-il en conclure que la maintenance n'est plus une préoccupation majeure des responsables informatiques ? La réalité du terrain vient contredire cette apparente indifférence. En effet, la maîtrise de la maintenance devient des plus critiques pour les raisons suivantes :

- Les coûts de maintenance, déjà très importants, continueront de croître et deviendront d'autant plus visibles que le prix des matériels diminuent.
- La rapidité d'adaptation des entreprises à l'environnement économique est un facteur clé de réussite. Cette flexibilité est conditionnée par leur capacité à faire évoluer rapidement leurs applications stratégiques, c'est à dire à maîtriser la maintenance évolutive.
- Les entreprises doivent également maîtriser la maintenance adaptative si elles veulent ouvrir leurs applications existantes (saisie de commande, suivi des livraisons,...) au client final via des infrastructures de type Internet.
- En ouvrant leurs applications à l'Internet les entreprises affichent leur niveau de qualité. Elles doivent donc maîtriser leur processus logiciel, notamment leurs activités de maintenance (corrective, adaptative, perfective et évolutive).

Soulignons de plus que la problématique de la maintenance ne concerne pas uniquement les vieilles applications qu'il suffirait de remplacer pour éliminer le problème. En fait, dès qu'une application nouvelle est mise en production elle entre dans le cycle de maintenance.

En s'appuyant sur des études et sur l'état de l'art en matière de génie logiciel, cet article présente les solutions qui dès aujourd'hui permettent de relever les défis de la maintenance.

Faits et chiffres

Deux études illustrent bien la problématique de la maintenance :

La première étude [1] identifie les 3 facteurs les plus négatifs rencontrés dans la maintenance :

1. L'absence de processus de maintenance logicielle (73% des cas).
2. Le manque de documentation à jour (68% des cas) qui se traduit par sa non utilisation.
3. Le manque de temps pour mettre à jour la documentation (54% des cas).

Une deuxième étude [2] montre que les coûts de maintenance augmentent dans le temps au fur et à mesure que les multiples modifications apportées au code rendent les interventions de plus en plus complexes.

Solutions

Pour améliorer la qualité du code, pour réduire les délais et les coûts, les trois approches les plus efficaces sont :

1. La mise en place d'un processus de maintenance.
2. La documentation ou redocumentation automatique des applications.
3. La rénovation des applications.

Avant d'aborder ces approches en détail, il est important de souligner que leur succès ne réside pas seulement dans leurs qualités intrinsèques mais également dans leur acceptation par les développeurs. En effet, ces derniers sont fondamentalement des artisans, au sens noble du terme, qui rejettent les solutions imposées. Le succès des solutions repose donc en grande partie sur l'écoute du terrain et l'apport de solutions concrètes aux problèmes quotidiens (approche « bottom-up »).

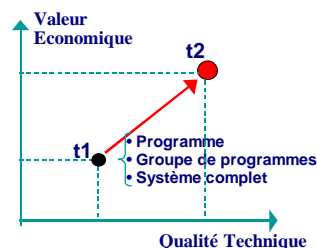
Processus de Maintenance

Pour mémoire, rappelons que la mise en place d'un processus de maintenance suppose la maîtrise de deux activités clés en matière de génie logiciel :

1. La gestion des configurations logicielles.
2. La gestion des changements.

Ces deux activités, bien connues, ne sont pas approfondies ici. Par contre, il est important de mettre l'accent sur un autre aspect, relativement peu mis en pratique : la métrologie. En effet, on ne peut pas maîtriser un processus si l'on ne dispose pas de mesures. Au-delà des indicateurs élémentaires de type jours x hommes ou nombre de lignes de code modifiées par personne, il est important de disposer d'indicateurs concernant :

- la qualité technique de chaque composant
- la valeur économique de chaque composant
- l'évolution des mesures dans le temps



Ces mesures doivent pouvoir être pondérées en fonction du contexte applicatif et être agrégeables verticalement en terme de groupes de programmes et de système applicatif, ainsi qu'horizontalement suivant des thèmes tels que : portabilité, fiabilité, maintenabilité...

Une fois identifiés les domaines de non-qualité, un modèle décisionnel doit être disponible pour proposer les actions correctrices.

La Redocumentation

La documentation des applications a toujours été le maillon faible de l'ensemble des livrables d'un projet applicatif. Ceci est confirmé par une étude de l'IEEE qui fait ressortir que 95% des logiciels ont une documentation obsolète et incomplète. L'étude [1] montre que dans 89% des cas les développeurs en sont réduits à utiliser le code source comme principale source de documentation. Il en résulte que la majorité du temps de maintenance est passé non pas à modifier le code mais à rechercher l'information, à la lire et à essayer de la comprendre. Une documentation véritablement utile dépasse les simples commentaires décrivant une ligne de code ou un groupe d'instructions. Elle contient l'ensemble des informations nécessaires à la maintenance de l'application, notamment :

- L'organisation de l'application : propriétaire, découpage fonctionnel ou organisationnel, etc.
- L'ensemble des composants mis en jeu : JCL, programmes, fichiers et bases de données, transactions, etc.
- Les relations à l'intérieur d'un composant et entre composants.
- Le dictionnaire des données.

La richesse du contenu informationnel n'est cependant pas le seul critère d'une bonne documentation. Elle doit également être à jour, diffusée à tous, et facile à exploiter, ce qui suppose :

- Des automatismes de mise à jour et d'accès partagé.
- L'adaptation au modèle cognitif de l'utilisateur.
- Une navigation aisée dans un composant et d'un composant à un autre.
- Une métaphore d'utilisation familière.
- Une capacité dynamique d'analyse d'impact intra ou inter-composants.

La connaissance étant localisée dans les codes sources et chez les experts, un système de redocumentation doit donc offrir un automate d'analyse des sources et une infrastructure de capture de la connaissance des experts.

Aujourd'hui, la technologie permet de retrouver et d'exploiter automatiquement toute la connaissance contenue dans les codes sources. A l'intérêt de l'automatisation il faut ajouter que les codes sources offrent l'avantage d'être à jour, puisque exploités quotidiennement.

A titre d'exemple, la redocumentation automatique [3] d'une application d'environ 5 millions de lignes de code dans le domaine de l'assurance génère environ 500 000 pages HTML de documentation exhaustive. Ceci met en évidence que :

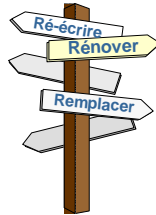
- Sans outil de génération automatique un tel volume de documentation ne peut ni être créé ni maintenu à jour dans le cadre de délais et de coûts raisonnables.
- Sans navigation hypertexte, une telle somme d'informations est inexploitable.

La capture de la connaissance des experts, appelée " assignation sémantique ", ne peut pas être totalement automatisée. Cependant elle est facilitée d'une part par l'interface humaine d'un outil de documentation doté de fenêtres de saisie de cette connaissance, et d'autre part par la technologie des bases de données objets permettant de stocker et de gérer automatiquement cette connaissance. En effet, un commentaire n'est qu'un des attributs d'un objet informatique.

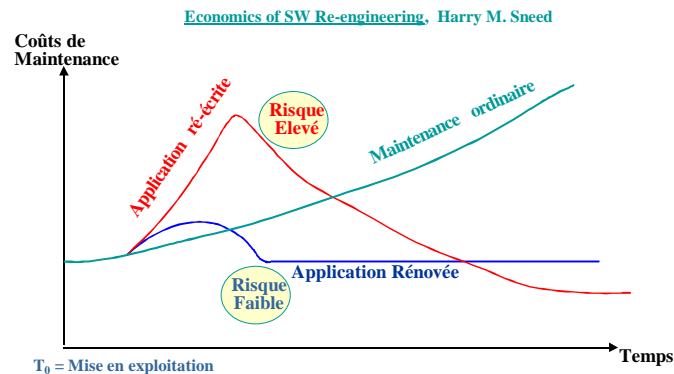
La Rénovation

L'objectif premier de la rénovation est d'améliorer la maintenabilité du code pour diminuer les coûts de maintenance. Au-delà de cet objectif, la rénovation a des implications sur la stratégie d'évolution des applications. En effet, lorsqu'une application devient trop difficile à maintenir le management a deux options :

- Réécrire l'application.
- Remplacer l'application par une autre application ou un progiciel par exemple.



Ces options, parfois inéluctables, ont l'inconvénient d'être risquées, car de type " big bang ", et coûteuses. La troisième voie, la rénovation, rendue possible par les technologies actuelles, permet, à investissement modéré, de diminuer rapidement les coûts de maintenance, d'allonger la durée de vie de l'application et donc d'en diminuer le coût global. En outre, la rénovation est une solution à faible risque car elle autorise les marches arrières, elle peut se découper en activités de taille raisonnable, et elle peut être en grande partie automatisée.



La rénovation englobe les techniques de redocumentation, de restructuration, de restauration et de reengineering. La redocumentation a été présentée plus haut. La restructuration recouvre un ensemble de techniques très automatisées telles que :

- Suppression du code mort et des variables non utilisées
- Réduction du nombre de GO TO
- Suppression des GO TO arrières
- Restructuration des données
- Suppression des fichiers inutiles
- Réduction des «Dominance Tree» (réduction des IF imbriqués, par exemple)
- Alignement de la précision des données numériques
- Normalisation du code et des noms

Dans une étape ultérieure on peut aborder les opérations de restauration de la signification des fonctions et de ré-ingénierie qui demandent plus d'interventions humaines mais qui sont

grandement assistées par les outils décrits plus haut et par d'autres techniques automatisées telles que le " slicing " permettant de regrouper les lignes de code suivant la nature des traitements effectués (affichage, accès aux données, etc.), et de les assigner à des fonctions (notion de " Function Mining ").

Un projet de rénovation doit commencer par un diagnostic initial. Il repose sur des mesures de la qualité du code et de la valeur économique des composants, et permet de choisir :

- Le sous-ensemble des composants à plus forte la valeur économique dans le système applicatif.
- Pour ce sous-ensemble, les techniques de rénovation les plus pertinentes.

La métrologie technique et économique des composants et des applications mentionnée précédemment joue donc également un rôle clé en matière de rénovation. Non seulement la métrologie permet de cibler les actions de rénovation, mais de plus elle permet de suivre l'évolution de la qualité dans le temps et donc de corriger les dérives éventuelles par des actions ponctuelles.

Technologies et Architectures

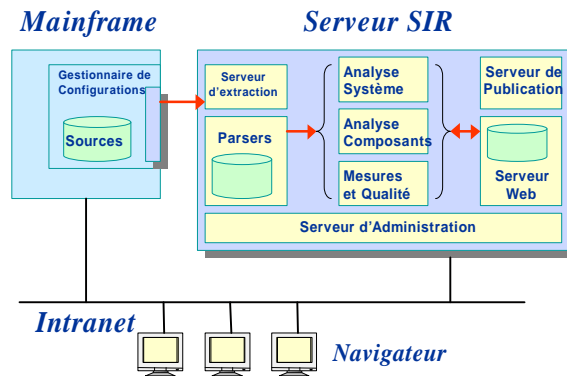
La redocumentation et la rénovation reposent sur des technologies d'analyse de code source, de manipulation de ce code, de stockage et de diffusion de l'information.

Pour l'analyse et les transformations, les outils travaillant directement sur le code source ont en général des capacités limitées. Les outils les plus évolués utilisent les techniques issues de la recherche sur les compilateurs. En effet, le compilateur est le seul automate qui sache interpréter correctement et de façon exhaustive la grammaire et la syntaxe d'un code source. Pour cela il construit une représentation abstraite du code source organisée sous forme d'arborescence très adaptée à la construction des control flow et data flow et à la propagation de modifications.

Le stockage sous forme d'objets répond à la nature des objets programmatiques très interdépendants. Il permet d'enrichir l'information et de l'interroger au moyen d'un langage formel.

En matière de diffusion et de visualisation de l'information, la technologie Internet, à base de serveurs centralisés de fabrication de pages HTML et XML, de serveurs de pages et de postes de travail dotés de navigateurs est très adaptée à la documentation des applications. Elle présente plusieurs intérêts majeurs :

- La notion de serveur central permet de contrôler l'intégrité de la documentation, son couplage avec la gestion des configurations logicielles, et de contrôler les accès.
- Les navigateurs présents sur tous les postes de travail éliminent l'installation et la maintenance de code propriétaire sur chaque poste.
- Le support de la navigation hypertexte et hypergraphe facilitent la navigation. XML permet d'exploiter toute la richesse des informations capturées.
- La métaphore Web réduit la formation au plus à quelques heures.



Une architecture sur les modèles client/serveur et Internet permet de créer des serveurs de documentation sans impact sur les mainframe ni sur les postes de travail. De plus, les serveurs peuvent être répartis sur plusieurs machines afin d'obtenir la scalabilité désirée.

On peut s'étonner que les solutions présentées ici ne soient pas largement répandues. Sans doute y a-t-il plusieurs raisons :

- Très peu de sociétés maîtrisent la technologie des compilateurs.
- Il faut en plus maîtriser la production de masse nécessaire au traitement de systèmes applicatifs de plusieurs millions de lignes de code.
- Peu d'acteurs ont la capacité à assembler et maîtriser l'ensemble des technologies mentionnées.
- Jusqu'à un passé relativement proche, la puissance de calcul et les gigaoctets de mémoire nécessaires empêchaient la mise sur le marché de solutions économiquement viables.
- La technologie Internet qui est la base de la diffusion de l'information est relativement récente auprès des équipes de maintenance.
- Les fournisseurs historiques d'outils de documentation se trouvent handicapés par leur architecture ancienne, orientée poste de travail, utilisant des structures de fichiers inadaptées et des interfaces de présentation propriétaires.

Paradoxalement, ce sont donc les derniers entrés sur ce marché qui ont pu capitaliser immédiatement sur cet ensemble de technologies et offrir des solutions bien architecturées, évolutives et économiques.

Des bénéfices mesurés

Les solutions préconisées dans cet article ont fait l'objet de mises en œuvre opérationnelles qui se sont révélées riches en enseignements. Par exemple :

1. La mise en œuvre d'un outil de redocumentation automatique et de navigation hypertexte [3] a permis de ramener le temps de certaines recherches à 1h30 contre 5h avec la documentation traditionnelle.
2. Dans un autre projet [4], l'amélioration du processus de maintenance, la restructuration de code applicatif et l'amélioration de la documentation, ont permis de satisfaire 9% de demandes de services en plus, en y consacrant 18% de jours x hommes en moins.

Dans le domaine qualitatif, il faut souligner que doter les équipes de maintenance d'outils efficaces et à fort contenu technologique contribue à redonner de l'intérêt à leur métier. Au management ils redonnent le contrôle sur le processus de maintenance et permettent d'engager une réelle politique qualité.

Conclusion

Constatant que la majorité des développeurs est affectée à des tâches de maintenance, constatant l'explosion du nombre d'applications mises en service, leur complexité croissante, la variété des langages et des composants utilisés, la pression sur les coûts et sur la qualité, on comprend que les solutions présentées ici ont un bel avenir car, en économisant quelques pour-cent du temps, elles ont un effet de levier important et immédiat sur les budgets informatiques et sur la qualité.

Ces solutions sont les premiers instruments de la lente industrialisation du développement et de la gestion des patrimoines logiciels qui seront décisifs dans les compétitions qui se jouent en ce début de XXIème siècle.

Références :

- [1] M. J. Castro Sousa et H. Mendes Moreira, *A survey on the Software Maintenance Process*, O-8186-8779-7/98 IEEE.
- [2] H. M. Sneed, *Economics of Software Re-engineering*, Journal of Software Maintenance : Research and Practice, Vol 3, N°3, Sept. 1991.
- [3] SIR, logiciel de redocumentation de la société NGSET.
- [4] G. Visagio, *Process Improvement Through Data Reuse*, IEEE Software magazine, July 1994.