Modèles et approches de la définition d'un prototype de logiciel ou d'environnement pédagogique informatique

Guy Vaudrin

Introduction et logique d'intervention

L'inévitable choix d'un modèle ou d'une approche (que ce soit conscient ou inconscient) pour l'élaboration des étapes préliminaires à la présentation d'un projet visant la production d'un logiciel, répond habituellement aux impératifs les plus immédiats, soient ceux correspondant aux réflexes élémentaires de s'assurer des disponibilités de ressources financières et humaines nécessaires à la réalisation du projet. Cela est particulièrement vrai, dans les disciplines traitant de l'évolution de la connaissance et de son transfert par le biais de la pédagogie.

Le sujet qui vous est présenté aujourd'hui, est habituellement absent à la présentation de ces projets. La raison en est très simple. Pourquoi s'attarder au niveau de stratégie et de planification de l'utilisation des outils et des techniques qui nous sont mis à notre disposition. Le plus important est la pertinence et la justesse du devis pédagogique présenté et l'informatique, vu sous cet angle ne devient qu'accessoire à sa réalisation.

Et pourtant... À l'heure où le virtuel est devenu une marque de commerce au moment où l'interconnectivité touche la planète dans ses moindres recoins, le saut quantique ressenti par chacun avec un tantinet de vertige mal contrôlé nous amène à pressentir un besoin urgent de « shifter » vers d'autres modèles de pensée, vers d'autres façons d'agir. Alors, pourquoi pas, avec une petite dose d'irrationnel, se laisser tenter vers l'interchangeabilité des cultures liées aux techniques. L'informaticien qui devient un peu plus pédagogue et le pédagogue qui devient un peu plus informaticien. En ne se limitant pas uniquement au contact avec la technique de la discipline connexe (ce qui est inévitable dans son utilisation) mais surtout dans le plongeon de la culture qui la sous-tend, en abordant les fondements de la science qui l'a créée.

Ce qui m'amène aujourd'hui, pour les pédagogues intéressés par le sujet, à traiter d'une des premières abstractions qu'a connu l'informatique et qui a servi comme base de cadre conceptuel de développement des logiques programmées, soient les modèles de processus de développement de logiciels.

Nous allons débuter avec la chute d'eau avec une étape cascade, puis suivra les prototypes au cours desquels apparaîtra le moteur interactif, puis la spirale pouvant traiter les mégaprojets avec flexibilité et le développement itératif qui en est le générique, et finalement un court aperçu du développement orienté objets.

Après avoir fait le tour du propriétaire sur les modèles de processus de développement de logiciels, nous allons transposer cette conférence en ateliers et comme des enfants avides de découvertes, nous allons démantibuler tous ces objets pour en soutirer les moindres composantes.

Identifiés, classifiés et réorganisés en sous-modèles, le brassage vigoureux et la manipulation créative de ces idées nous permettra d'initier une étude qui pourrait s'intituler « Modèles et approches itératives, interactives et proactives de création d'environnements pédagogiques informatiques ».

La communication se terminera sur quelques commentaires généraux et les pistes de recherche que l'auteur suggérera, pour mieux exploiter le collectif des productions déjà réalisées et par conséquent favoriser un meilleur retour sur l'investissement effectué dans ce domaine.

Le modèle de développement de la chute d'eau

Dans les années 60 et 70, toute l'importance du développement est mise sur la planification et le contrôle et généra des coûts et des délais exagérés (Basili et Miusa, 1991). Ce modèle a été largement utilisé pour les projets complexes et de grande envergure avec l'idée principale d'exprimer clairement ce qu'il faut faire avant de développer. Ses avantages sont donc principalement qu'il force le développement à être structuré et à être facilement gérable. Le processus génère une série de documents qui peuvent être à tout moment utilisés pour vérifier et maintenir le système. Le paradigme Entrée-Programmation-Validation-Sortie (traduction libre de Radice et al., 1995) en est la principale caractéristique. En dépit du concept de la chute d'eau qui suggère une progression unidirectionnelle du processus, les phases de design et de codification sont implantées interactivement avec des étapes de vérification.

Le modèle de développement de la chute d'eau est une approche disciplinée de développement de logiciels. Lorsqu'on utilise ce modèle, l'attention doit se concen-

trer principalement sur les délivrables intermédiaires (document de design, règles d'interfaces, stratégies des tests...) plutôt que sur les séquences d'activités pour chaque phase de développement. C'est donc plus une gestion de micro-projets qu'une gestion de méga-projets.

Le prototypage

Dans le modèle de la chute d'eau, les réquisitions doivent être bien définies une fois pour toutes puisque ensuite on passe au design et à la codification sans y revenir. Dans le cas où les réquisitions changent significativement au cours du développement, on voit bien que ce modèle risque de créer de sérieux problèmes à la finalisation du projet. Pour palier à ce type de problèmes, on a donc conçu une approche de développement par prototypes permettant ainsi une meilleure interaction entre les phases de conception et d'utilisation. Un prototype est une implantation partielle, physique et logique du produit en présentant toutes les interfaces externes. Les utilisateurs potentiels peuvent alors fournir un feedback à l'équipe responsable du développement avant que commence le développement de la version finale du produit. Par cette approche, les utilisateurs et les concepteurs peuvent mutuellement se clarifier les réquisitions et les interprétations et assurer ainsi une bonne évolution du projet.
Le facteur critique du succès est le déroulement effectué au centre du processus qui raffine à chaque cycle le design et la construction du prototype. Différentes technologies peuvent être utilisées pour faciliter l'interaction entre la conception et l'utilisation du prototype telles que la programmation réutilisable et les langages de quatrième génération avec interface d'exploitation graphique. Le prototypage est bien approprié pour les petits projets mais cette approche peut s'avérer très utile à l'intérieur de certaines phases de projets de plus grande envergure.
La technique du « Rapid throaway prototyping approach » popularisé par Gomaa et Scott (1981) est largement utilisée dans les cas d'items à hauts risques là où il est difficile au début de connaître toutes les dimensions du problème. Le prototype est donc utilisé par les concepteurs comme source première d'information.
Finalement, on peut ajouter le prototypage évolutionniste qui est utile lorsqu'on connaît bien les besoins mais que les demandes sont prioritaires (Hough, 1993).

La spirale

S'appuyant substantiellement sur le prototypage et la gestion des risques, le modèle de la spirale (développé par Boehm, 1988), raffine le modèle de la chute d'eau en étant plus flexible. Ce modèle est encore en essai, le plus complet étant le TRW - Software Productivity System, décrit par Boehm.
On remarque dans ce modèle que chaque niveau d'élaboration de chaque portion du projet est soumis aux mêmes séquences d'étapes correspondant à un cycle de la spirale. À la première étape de chaque cycle, nous identifions les alternatives qui peuvent engendrer l'implantation de la portion du produit et les contraintes imposées sur les applications de ces alternatives. Dans une deuxième étape, les alternatives sont évaluées en fonction des objectifs et contraintes identifiées.
On a donc une approche orientée-risque où le prototypage est un outil important. À cela s'ajoute pour chaque cycle, des simulations, l'élaboration de modèles et la fixation de niveau à atteindre.
La spirale est, par rapport à la chute d'eau, plus flexible et, par rapport au prototypage, est capable de traiter des projets complexes divisés en portion de produits. Mais par contre cette démarche est exigeante en ressources expertes surtout à l'égard de l'analyse et la gestion des risques. Si on identifie mal les risques à prioriser, le projet peut se retrouver avec de graves problèmes.

Le modèle du processus du développement itératif

Chacun des trois modèles qui précèdent contiennent dans leurs phases des éléments itératifs. La chute d'eau a des itérations design-Inspection et code-inspection, le prototypage a en son cours l'itération design-évaluation qui constitue le moteur de cette approche, et finalement la spirale qui est en fait un cas spécifique de processus itératif. Par contre, le modèle du processus du développement itératif est un modèle générique sous lequel peuvent exister plusieurs formes variées de modèles dont celui de la spirale.
L'approche d'amélioration itérative (Basili and Turner, 1975) débute avec les demandes et développe un sous-ensemble du produit qui satisfait les besoins exprimés par les utilisateurs et qui fournit une expérience d'apprentissage pour les développeurs. Basé sur l'analyse de chaque produit intermédiaire, le design et les besoins exprimés sont interactivement modifiés par des séries d'itérations qui en fin de ligne, fournit un système répondant à la fois à l'évolution des besoins exprimés et l'amélioration du design basé sur la dynamique rétroaction-apprentissage.

Le modèle du processus du développement orienté-objets

L'approche basée sur le paradigme objet a été introduite dans les années 80 et a d'abord été caractérisée par la phase programmation et codification du développement des logiciels. Ce qui nous amène au coeur du problème des développements basés sur cette approche. Chaque langage 00 a sa propre définition de ce qu'est l'approche 00. Il y a eu bien sûr certaines tentatives, particulièrement dans les bases de données, de standardiser au minimum les définitions des éléments fondamentaux de l'architecture sur laquelle on pourrait baser cette approche. Mais, encore aujourd'hui, il existe très peu d'information à propos du processus de développement orienté-objet et ce qui existe peut être utilisé pour les petits projets qui n'ont pas nécessairement besoin de processus de développement (Stephen H. Kan, 1995). Il est hasardeux de parler d'une approche commune favorisant l'interconnexion et la connectivité des différents projets qui sont basés sur l'00. Il y a donc de ce côté un besoin urgent d'abstraction et de mathématisation de l'approche, pour en faire un modèle aux projets, quelque soit leur envergure.
Cette approche continue et continuera, pour plusieurs années, à avoir un effet majeur dans la communauté du développement des logiciels. Elle se distingue des programmations traditionnelles qui séparent les données de la programmation alors que cette approche considère un objet comme un ensemble de données et d'opérations qui peuvent agir sur un ensemble de données et sur un ensemble d'opérations.
Branson et Herness (1992) ont proposé un Processus Outil Objet pour les mégaprojets basés sur huit étapes centrées sur l'observation, des séries d'inspections, un ensemble de technologies et des règles régissant le prototypage et la vérification. Ces étapes se divisent en trois phases logiques : l'analyse offrant une représentation concise des demandes des utilisateurs et précisant la plate-forme d'implantation (environnement matériels et logiciels).

Phase 1
Étape 1 Modélisation du système essentiel
Étape 2 Dérivation des classes essentielles

Phase 2
Étape 3 Modification du système essentiel avec les contraintes
Étape 4 Dérivation des classes essentielles additionnelles
Étape 5 Hiérarchisation des classes
Étape 6 Définition des interfaces

Phase 3
Étape 7 Finalisation du design
Étape 8 Implantation de la solution

S'ajoute à cette méthodologie.

Généralités des modèles

Qu'ils se réalisent en série ou en parallèle, avec itération ou non et de façon interactif ou unidirectionnel, avec ou sans les utilisateurs, tous les développements sont soumis aux mêmes généralités, soient :

*	l'expression des besoins	Analyse
*	la représentation du développement à réaliser	Design
*	la production du développement	Codification
*	la validation	Testing
*	la mise en oeuvre	Livraison

Aussi, il est nécessaire de tenir compte des caractéristiques suivantes du projet :
envergure
domaine
discipline
contraintes générales
conditions critiques
...