Le RAD (Rapid Application Development)

Quels outils pour quelle méthode ?

Conçue à la fin des années 80 par James Martin et associés, la méthodologie RAD (Rapid Application Development) a beaucoup fait parler d'elle, essentiellement au travers des outils qui s'y réfèrent.

Quel est le rapport entre la première et les seconds ?
Six ans après l'introduction du terme, quel bilan peut-on en tirer ?
La méthode est-elle dépassée ?
Les outils tiennent-ils leurs promesses ?

La méthodologie

La méthode a été décrite pour la première fois par James Martin en 1991. Bien qu'elle ait pu avoir évolué depuis, c'est cet imposant ouvrage (plus de 750 pages) qui nous servira de référence pour la suite. Remarquons d'emblée que le livre ne décrit pas seulement une méthode, mais surtout une méthodologie, c'est-à-dire une réflexion sur ou autour d'une méthode ou d'un ensemble de méthodes. En particulier, l'ouvrage décrit ce que devrait être la méthode idéale de développement rapide.

Une méthode RAD devrait, d'après son auteur, apporter trois avantages compétitifs à l'entreprise :

- une rapidité de développement (cette caractéristique donne son nom à la méthode) ;
- un faible coût de développement ;
- une application de grande qualité.

Ce dernier aspect est souvent négligé par les vulgarisateurs de la méthode, y compris souvent par les fournisseurs des outils qui s'en réclament, au profit de la rapidité du développement.

Le constat de départ concerne les carences des méthodes actuelles : les applications sont très longues à développer. Lorsqu'elles arrivent à l'utilisateur final, elles ne correspondent plus au besoin, car nous vivons dans un monde où tout s'accélère, et les besoins se modifient au même rythme. De plus, l'utilisateur final est peu impliqué dans la définition des besoins, sans parler de la conception de l'application.
Pour pallier ces inconvénients, pas de recette miracle. Mais un nombre restreint d'idées-forces.

Les principes

Les principes du RAD sont simples et ils découlent du bon sens. S'il y a quelque chose de révolutionnaire dans la méthode, c'est leur formalisation.

Voici les principes de base :

- Tout d'abord, le développement devrait être effectué par de petites équipes, expérimentées et ayant reçu toute la formation nécessaire.
- Le développement doit s'effectuer en faisant appel à une méthodologie formalisée.
- Une équipe de développement RAD doit disposer d'outils puissants qui automatisent le développement dans sa globalité, tant en ce qui concerne les étapes du développement que l'enchaînement de ces étapes.
- L'équipe de développement doit être correctement gérée, par un encadrement encourageant la réutilisation des composants, et attentif aux aspects humains du management de projet.
- Enfin, les utilisateurs finaux devraient s'impliquer dans le processus de développement.

La méthode RAD est donc fondée sur quatre ingrédients de base :

- Les **outils** : de conception, de prototypage, de génération de code, disposant d'un langage de haut niveau (L4G), ces outils étant fédérés par un référentiel et constituant un atelier puissant.
- Les **personnes** : elles doivent être compétentes, expérimentées, correctement formées aux techniques et aux outils utilisés, et, cela ira mieux en le disant, ...motivées ! L'auteur revient souvent sur cet aspect fondamental, et un chapitre entier y est consacré. Ce qui est tout aussi essentiel, c'est que l'utilisateur final soit directement impliqué dans chaque phase du projet.
- Le **management** : une gestion du projet, en particulier en ce qui concerne les aspects humains, est essentielle. Le pire ennemi du RAD, c'est la bureaucratie, dit James Martin. Combien de projets n'a-t-on pas vu dériver ou stagner, ensablés par une bureaucratie toujours plus lourde ?
- La **méthodologie** : Rien ne se fera sans une méthodologie, à la fois bien formalisée et souple. L'auteur recommande que la méthode soit consignée, non sur un document qui remplirait la moitié d'une armoire, mais dans un « hyperdocument » que chacun pourrait parcourir rapidement et de façon non linéaire en fonction de ses besoins et de son rôle sur le projet. Ce document doit être adapté d'un projet à l'autre, toujours affiné, et consultable par tous. Seul un document électronique disponible sur un réseau peut convenir.

Les quatre phases

Un projet RAD se découpe en quatre phases :

1. **Phase de définition des besoins**. Elle se matérialise par des sessions de travail appelées **JRP** (Joint Requirements Planning ou définition conjointe des besoins).
2. **Phase de « conception utilisateur »** (user design). Elle se caractérise par les sessions **JAD** (Joint Application Development ou développement conjoint d'application), qui est un des signes distinctifs du RAD.
3. **Phase de construction**, mêlant les spécifications détaillées et le codage.
4. **Phase de finalisation** et mise en place (en anglais : cutover).

Signalons brièvement deux points essentiels :

- tout d'abord, ces quatre phases constituent un processus itératif. Le « cutover » débouche à nouveau sur une nouvelle définition des besoins, tant que cela s'avère nécessaire ;
- d'autre part, ces phases peuvent, dans une large mesure, être parallélisées.

Les JRP

Le principe de la « définition conjointe des besoins », autrement dit JRP, est simple, mais pourra sembler quelque peu révolutionnaire à certains : sélectionnez les meilleurs éléments, à la fois chez le maître d'œuvre et chez le maître d'ouvrage, et faites les plancher ensemble, dans une salle de réunion sans téléphone et le plus loin possible de toute source de distraction. Ces personnes doivent travailler ensemble, définir ensemble les besoins du système à réaliser.

Lorsque l'auteur parle des personnes-clés, il ne s'agit ni du management seul, ni des « as de la programmation », mais d'une sélection de l'ensemble des profils impliquées dans le projet. Une de ces personnes clés : l'« executive owner », en d'autres termes celui qui paye pour avoir le produit final (ou son représentant attitré). Il doit s'engager à fond dans le processus.

On s'en doutait un peu, mais mieux vaut le répéter que de laisser planer le moindre doute : l'animateur de ces groupes de travail doit être un communicateur hors pair !

Les JAD

L'idée derrière les JAD, qui constituent la substance de la phase de conception, est d'augmenter la productivité globale du développement en réunissant lors d'une même session les utilisateurs et les concepteurs. Encore une fois, son efficacité repose sur deux facteurs clés :

- le facteur humain : dynamique de groupe et suppression des barrières organisationnelles ;
- l'outillage : référentiel puissant et outils de prototypage rapide, les prototypes étant, bien sûr, réutilisables pour le développement du produit final ;

Les « timebox »

Dans la plupart des projets de développement d'application (si ce n'est dans TOUS les projets), il y a des glissements dans les délais. Cela tient à plusieurs causes. Mais on sait en particulier que 80% des fonctionnalités sont réalisées en 20% du temps, et que les 80% du temps restant sont pris par la réalisation des 20% des fonctionnalités à réaliser. En fait, plus le développement semble s'approcher de la fin, moins on avance vite. Cela ressemble à du polissage de pièces. Plus on avance, plus l'abrasif doit être fin. Et plus l'abrasif est fin, moins on avance !

En RAD, la loi est dure, mais simple : le glissement est interdit !
Pour un habitué du développement classique, c'est une révolution ! Comment peut-on interdire tout glissement dans les délais ? Et si je n'ai pas fini de développer, comment faire ?

La solution, ici aussi, est simple. Plutôt que d'autoriser des glissements dans les délais, on va tolérer un glissement dans la réalisation : réduisez les fonctionnalités, mais quoi qu'il arrive, vous finirez à temps.

N'oublions pas que le RAD, et les outils qui sont supposés le supporter, encouragent le développement par affinements successifs. Pour reprendre l'analogie précédente, si on n'a pas le temps d'obtenir un poli parfait, on va se contenter d'une pièce un peu rugueuse (ou même, dans les cas pathologiques cette fois, d'une pièce à peine dégrossie). Mais on ne va pas se donner du temps pour polir encore et encore.

Un timebox s'achève par une revue, qui décidera s'il faut ou non réitérer une deuxième boucle de la spirale de développement.
Le développement itératif ne rend pas seulement le « timeboxing » possible. Il le rend nécessaire. Car le danger des « fonctionnalités rampantes » (je raffine encore, encore et encore) menace de faire dériver un projet à l'infini.
Le management d'un développement de ce style est délicat. Une petite équipe (deux personnes en moyenne) va être mise sous pression pendant une durée fixée (en général 60 jours). Les actions à réaliser alors dans cette boîte temporelle[1] doivent être correctement estimées.

Critique de la méthode

Outre les aspects « outillage », abordés plus loin, la méthode comporte un certain nombre de points faibles.

L'auteur insiste sur la réutilisabilité. Or, pour qu'un composant soit réutilisable, il doit être utilisable dans des contextes différents par des applications différentes, ou par des modules différents. Ceci implique soit une réflexion en amont de sa construction, soit une « généralisation » de ce composant a posteriori pour l'ouvrir à d'autres utilisations que celle pour laquelle il a été conçu.
Si les délais sont très courts, c'est bien ce travail de fond qui sera le premier sacrifié. Si un développeur est pris par le temps, on ne voit pas quelle raison le pousserait à chercher à rendre un composant réutilisable à plus long terme, donc à accomplir une tâche dont il ne verra jamais les fruits.

Il n'y a pas de miracles. La réutilisabilité demande un investissement à long terme. Elle doit être gérée. Elle doit faire partie des objectifs de l'équipe de développement. À moins que le développement consiste en un assemblage de composants préexistants, conçus et développés en dehors du contexte du RAD. Et nous retombons alors dans la problématique de la conception et de la réalisation par objets.

[1] *Je ne sais pas s'il existe un terme français pour « timebox ». Si ce n'est pas le cas, je me permets d'en suggérer un seul : délai-capsule.*

Et les outils?

Après cette brève description de la méthodologie, on peut se poser une question : quels rapports peut-on trouver avec les outils qui, à tort ou à raison, se réclament du RAD? À vrai dire, peu de choses.

Ceux que l'on appelle habituellement les « outils RAD » ont en commun un certain nombre de caractéristiques :

- ils sont le plus souvent « orientés objet », ou se présentent comme tels ;
- ils comportent un outil de construction d'IHM, ou de génération automatique de l'IHM ;
- le langage utilisé pour la programmation est un langage de haut niveau, ou un langage dit « de quatrième génération ».

Les trois caractéristiques précédentes encouragent un cycle de développement en spirale. La gestion des liens logiques entre les objets de l'interface graphique et les variables utilisées dans la programmation est entièrement automatisée. L'outil encourage un dialogue permanent avec l'utilisateur final, grâce aux possibilités de maquettage et de prototypage.

Le maquettage et le prototypage

Les outils de développement de nouvelle génération sont, du point de vue du maquettage/prototypage, bien plus puissants que tous ceux que l'on pouvait trouver sur le marché lorsque le livre de James Martin est paru.

Si les « outils RAD » méritent leur qualificatif de « RAD », c'est bien sur cet aspect-là. Les outils « modernes » permettent de construire quelques dessins d'écrans en une heure.
De là à construire l'interface homme-machine en temps réel, en réunissant informaticiens et utilisateurs dans une même salle pendant quelques jours, il n'y a qu'un pas. Si les recommandations de James Martin sont respectées, en particulier en ce qui concerne les aspects humains, ce pas peut être franchi.

L'orientation objet

James Martin ne préconise pas l'orientation objet dans son ouvrage[2]. Tout au long de la description de la méthode est maintenue la séparation entre données et traitements, ce qui va bien entendu à l'encontre de l'approche par objets (principe d'encapsulation).
Par ailleurs, James Martin précise que le RAD n'a de sens que par ce qu'il s'inscrit dans un cadre méthodologique plus large, constitué par l'Information Engineering (qui n'est pas une méthode objet).
Cependant, l'orientation objet est actuellement l'approche la plus prometteuse pour favoriser la réutilisation.

Ce que serait un outil RAD idéal

Lorsque James Martin parle d'outils, il cite IEF, Cohesion et AD/Cycle (nous sommes en 1990). Il ne s'agit pas là d'outils que l'on qualifierait spontanément de « RAD ». Néanmoins, certains aspects du RAD sont mieux traités par ces outils que par les outils RAD des années 95. En particulier, ils sont intégrés, ou tendent vers l'intégration totale, autour d'un référentiel puissant.

Quel serait alors le portrait de l'outil RAD idéal ? On peut tenter de l'esquisser en faisant la liste des fonctionnalités nécessaires à la mise en œuvre de la méthode:

- référentiel puissant, prenant en compte la totalité du cycle de vie ;
- ce référentiel permet un travail en équipe, souple, efficace, non bureaucratique ;
- le référentiel doit être facile à utiliser, même par des personnes y faisant rarement appel (directeur de projet, consultants externes, utilisateur non informaticien) ;

---

[2] *Le terme d'objet est effectivement utilisé à quelques reprises, mais dans le sens d'objets du référentiel. Le fait que le référentiel soit orienté objet n'implique en aucun cas que la conception de l'application se fasse par objets. Par ailleurs, James Martin est très critique vis-à-vis des techniques objet.*

- la gestion de projet devrait être intégrée à l'outil. Elle devrait prendre en compte la gestion du temps, l'enchaînement des étapes, le parallélisme entre étapes, l'itération du cycle de développement, la communication entre individus ;
- l'outil devrait favoriser la réutilisation des composants.

Parmi les outils actuels, combien y en a-t-il qui permettent de répondre « oui » à tous ces critères ? Chacun a son idée sur le sujet.

Personnellement, je me permets les remarques suivantes :

- Seule l'orientation objet permet d'obtenir un niveau industriel de réutilisabilité. La réutilisation par « bibliothèques de routines » a montré ses limites. Quand à la réutilisation par « copier-coller », elle nous mène droit à la catastrophe organisationnelle. Cependant, n'oublions jamais que la réutilisation doit résulter d'une volonté stratégique, et qu'elle a un coût.
- Le référentiel puissant et « intelligent » préconisé par James Martin est introuvable sur la plupart des outils de développement rapide.
- James Martin insiste sur les aspects humains du développement. On ferme parfois les yeux sur une évidence : aucun outil ne réglera les problèmes humains. Même si certains outils, ou l'absence de certains outils, ne peuvent que les envenimer.

Adéquation des outils à la méthode

Le tableau suivant résume les différences entre les caractéristiques du RAD, telles qu'elles sont décrites par James Martin, et la prise en compte de ces caractéristiques en termes de fonctionnalités correspondantes dans des outils RAD.

Aspects	**Méthode RAD de James Martin**	**« Outils RAD »**	**Remarques**
Personnes	De très bon niveau, très motivées, dynamiques ; jouent un rôle important dans le projet	Pas de prise en compte	
Référentiel	Puissant et « intelligent » (fait appel à des techniques d'intelligence artificielle)	Disposent souvent d'un référentiel, mais en général bien moins puissant que celui préconisé	Le référentiel est un point clé pour James Martin
Conception données / traitements	Entité-relation et fonctionnelle ; séparation données / traitements	Entité-relation parfois, mais, de plus en plus souvent, outils orientés objet	
Outils	Intégrés (I-CASE) prenant en compte tous les aspects	Rarement intégrés ; spécialisés sur une partie du cycle	
Cycle de vie	4 phases, elles-mêmes décomposées en étapes ; démarche itérative	Pas de référence au cycle de vie, mais la démarche itérative est favorisée	
Gestion du temps	Timeboxing (blocage des étapes dans le temps)	Pas de référence au temps	
Réutilisation	Importante, mais n'est pas décrite clairement	Possible	L'orientation objet, souvent favorisée par les « outils RAD » permet une réutilisation effective
Construction de l'IHM	Importante	En général, très bien prise en compte	L'aspect le plus positif des outils modernes

Conclusion

Les équipes décrites par James Martin s'appellent SWAT : Skilled With Advanced Tools. En français : compétentes et armées d'outils de pointe. On parle souvent des outils RAD. Et si on parlait un peu plus des personnes ?

De bons outils, nous l'admettons volontiers, sont indispensables. Rares, sans doute. Chers aussi, mais il suffit d'y mettre le prix pour les avoir. Le prix de six mois, un an de salaire par poste. Exorbitant ?

Ce qui est bien plus rare, cher, précieux, c'est une équipe de projet capable de s'embarquer dans une telle aventure. Ce qui est d'une valeur inestimable, c'est un chef de projet capable à la fois de communiquer son enthousiasme, d'animer une équipe, de comprendre le besoin du client, et de secouer sa propre organisation pour y puiser les ressources qui lui manquent.

Car le RAD, à bien lire l'auteur du livre du même nom, c'est avant tout une affaire de personnes et d'organisation. La forme des bureaux et de la salle de réunion y joue un rôle aussi important que la structure du référentiel. La motivation des développeurs y joue un rôle aussi primordial que le générateur d'interface. Vouloir faire du RAD et n'y voir que des outils, c'est vouloir courir un 400 mètres à cloche-pied ! ▲

Yves Constantinidis

Bibliographie

James Martin, Rapid Application Development, Macmillan, 1991
Le livre de référence en matière de RAD, par le père de la méthode. En anglais

Jean Hugues, Bernard Leblanc, Chantal Morley
RAD, Une méthode pour développer plus vite, InterEditions
Un ouvrage clair et concis. En français

Bertrand Meyer, Object success; A manager's guide to object orientation, its impact on the corporation and its use for reengineering the software process, Prentice Hall
Ouvrage intéressant pour ceux qui s'intéressent à l'orientation objet. Ne traite pas spécifiquement du RAD. Accessible aux non-spécialistes. On y trouve en particulier une réflexion intéressante sur la réutilisation, et sur les aspects humains de la conduite de projet, en particulier dans le cas des développements par objets.