LA RACHE, UNE MÉTHODOLOGIE RÉALISTE MAIS FORMALISTE

Tome 1 - Simuler une conduite de projet efficace tout en continuant à travailler à la Rache

par Sukender

Département de recherche fondamentale de glandouille

IILAR - International Institute of LA Rache

Dirigé par M. Jean CIVE, Président Honorifique IILaR

Révisions du document :

1.1	6 décembre 2005	Ajout d'information sur la modélisation Modification du style
1.0	4 décembre 2005	Article initial

Introduction

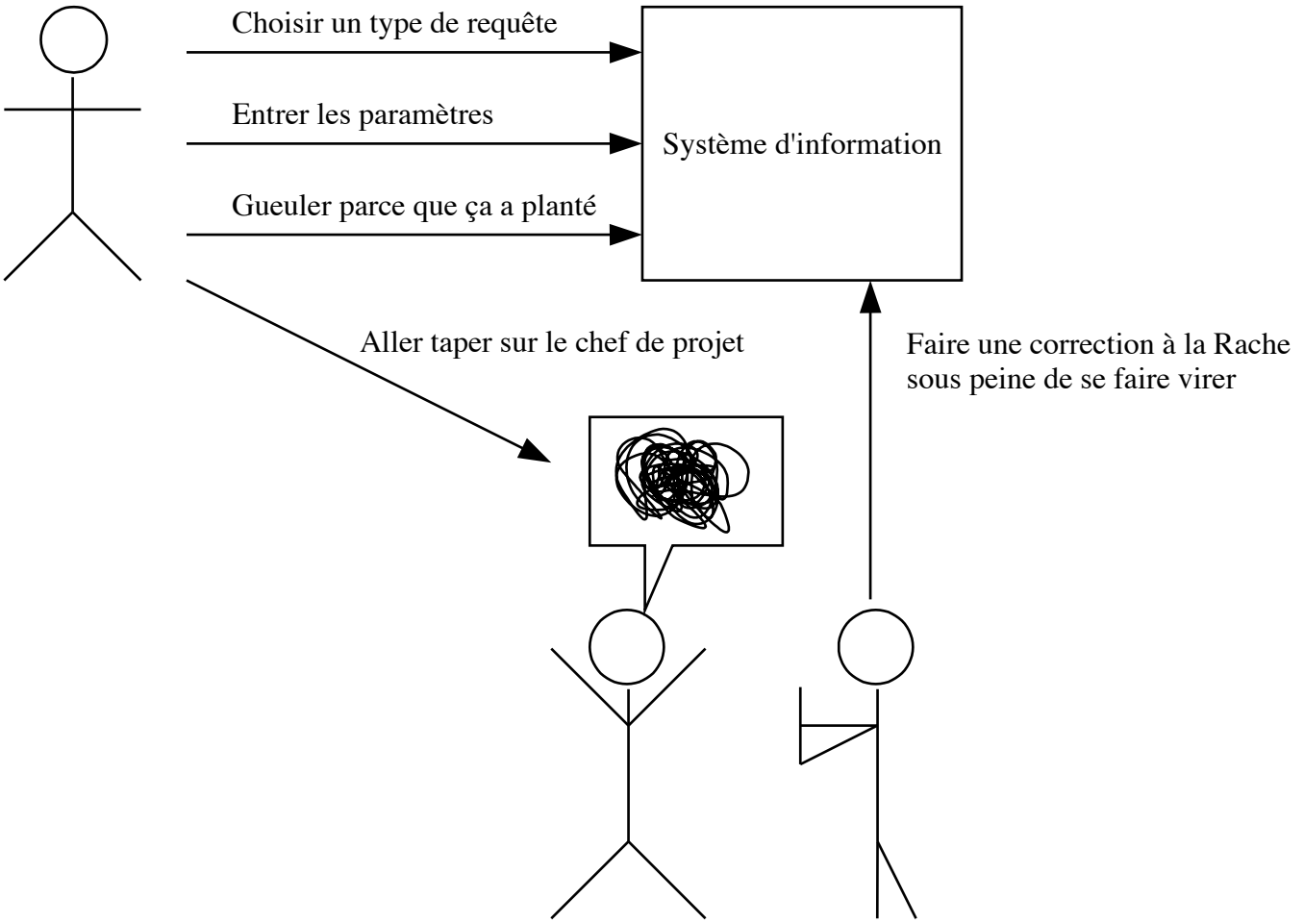
La Rache est une méthodologie très formalisée, contrairement à ce que l'on pourrait croire. En effet, faire un projet à la Rache est une chose, mais faire croire qu'on l'a fait correctement malgré la méthodologie Rache en est une autre. Toute la phase de formalisation structurelle du projet à la Rache est basée sur le concept et l'essence même du Pipotron® : des mots compliqués, si possible américanisés et vides de sens, et des phrases alambiquées pour en mettre plein la vue. Que l'on soit bien clair : du fait de la baisse de confiance induite, il est nécessaire d'imaginer chacune des voies déjà en notre possession[1].

Deux axes majeurs sont à étudier pour un projet à la Rache crédible : la modélisation du problème et la conduite de projet. Nous passerons rapidement en revue la modélisation mais étudierons plus en détail la conduite de projet dans ce tome. La modélisation d'un problème n'étant absolument pas la priorité (puisque c'est en général ce qu'on fait à la fin pour montrer aux supérieurs qu'on a bien bossé), nous l'approfondirons dans un prochain volume.

1. Merci le Pipotron !

MODÉLISATION - URML (PRÉ-REQUIS)

Les nouvelles technologies s'appuient sur le modèle objet. En terme d'analyse et de modélisation objet, URML (Ununderstandable Random Modeling Langage) est aujourd'hui un standard incontournable, stabilisé, industriel, et conforme à la Rache.Voici à titre d'exemple un schéma URML :



Il est tout à fait envisageable de s'appuyer sur d'autres normes et langages, mais nous nous concentrerons ici sur l'URML pour des raisons de facilité d'apprentissage.

Avantages

La norme URML permet de montrer au chef que vous avez compris son problème et que vous êtes l'homme / la femme de la situation. Il permet par conséquent, mais indirectement, une augmentation de votre salaire. Il permet aussi de montrer à vos collègues que vous leur êtes supérieur et qu'ils vous doivent le respect. L'URML est aléatoire (d'où son nom) et ne vous demandera que très peu d'efforts pour paraître crédible et important.

L'URML permet aussi :

- De vous sentir bien
- D'augmenter votre confiance en vous
- D'accroitre vos performances sexuelles
- De gagner au loto
- Et de régler automatiquement l'horloge de votre four lorsque celle-ci retarde

Inconvénients

À l'inverse d'autres normes ou langages, l'URML n'est absolument pas fait pour la communication utile. C'est un language à base de *Stupid Elements To Impress Bosses* (SETIB, ou BITES en français), c'est-à-dire destiné à impressionner et non à communiquer des informations tangibles.

Un autre problème qui se pose avec l'utilisation d'URML survient lorsque quelqu'un tente de comprendre l'un de vos schémas. La norme en elle-même ne répond pas à ce problème (bien que des solutions standards existent) et fera donc appel directement à vos compétences de génération d'explications foireuses.

Conclusion

Bref, vous l'aurez compris, la norme URML, sans être une solution miracle, vous apportera un gain de productivité apparente significatif.

Nous nous attarderons à décrire les régles qui régissent cette norme dans un prochain tome de cette publication.

CONDUITE DE PROJET

Il est important pour une bonne conduite de projet (Entendez "bonne" = "qui masque que vous avez fait un projet à la Rache") de se donner les moyens de faire du vent. Brassez de l'air en ayant l'air soucieux et occupé ; vous aurez ainsi déjà acquis 20% de la méthodologie. Faire bonne impression avec des schémas et des termes inutiles vous garantit le plus souvent +80 à +120% de pognon sur un projet où vous ne masquez pas la vérité. Il en va de même avec les étudiants pour les notes, et pour les développeurs (= non responsables) pour justifier temps et primes supplémentaires.

Découpage en lots

Plusieurs façons de découper un projet existent. Vous devez dès le début en choisir une pour montrer que vous avez déjà une idée de ce que sera le projet final.

- Itératif et incrémental : le projet est découpé en itérations de courte durée (de 10 minutes à 2 heures, en fonction de la durée de la pause café) qui vous empêchent de savoir ce que vous allez faire dans 2 itérations. Vous pouvez grouper les itérations dans de plus grosses. A la fin de chaque grosse itération, vous faites péter le champagne et demandez une augmentation de salaire.
- Centré sur l'architecture : tout système complexe doit être décomposé en parties modulaires afin de garantir que celui qui bosse sur un module ne sache rien du module d'à côté. Vous permettez ainsi aux développeurs de faire 2 ou 3 fois le même travail sans s'en rendre compte et participez ainsi à la réduction du taux de chômage.
- Piloté par les risques : les risques majeurs du projet doivent être identifiés au plus tôt. C'est donc à la phase finale (la recette) que vous allez réagir car c'est à ce moment que les utilisateurs vont gueuler sur les horribles bugs de votre appli. Vous pourrez alors lancer la production d'un patch correctif mal fait, payant et amenant d'autres bugs.
- Conduit par les cas d'utilisation : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Vous aurez ainsi un projet très beau qui fait en apparence plein de choses et un code imbitable, et surtout inmaintenable en dessous. Bravo, c'est certainement la meilleure méthode.

N'hésitez surtout pas à analyser tous les *business cases* en vous conformant au *risk management document* afin d'assurer un *pattern matching* sur les *users behavior tools*.

Phases du projet

Voici le déroulement 'type' d'un projet à la Rache.

Expression du besoin

On vous a dit « Tiens ça serait bien si tu pouvais nous pondre un ptit outil vite fait pour automatiser ce truc chiant sur lequel je bosse 2 heures et demi par jour ». Traduisez « Si tu parviens à me faire gagner 2 heures par jour avec un énorme logiciel de la mort, non seulement je me fais augmenter parce que je bosserai plus vite, mais en plus j'aurai 2 heures de plus pour tromper mon mari / ma femme sans qu'il / elle ne se doute de rien. Ah oui, et toi tu n'auras rien de plus sur ton salaire bien sûr. ».

Vous venez donc avec un bloc notes et un stylo à la machine à café ou à la photocopieuse pour mettre par écrit ce que vous raconte votre demandeur. Vous vous rendez compte qu'il ne sait absolument pas ce qu'il veut concernant le programme, mais il est certain de vouloir quitter le boulot 2 heures plus tôt. Vous l'entendrez probablement dire « Je veux un gros bouton qui fasse tout ». Profitez-en pour définir la charte graphique de votre projet : un gros bouton et c'est tout.

Retournons à nos moutons. Votre demandeur ne sait pas ce qu'il veut, et de toutes façons il n'a pas le temps d'y réfléchir. Vous lui proposez donc des idées, de préférence réalisables « vite fait sur le gaz », mais attention ! Faites-lui bien comprendre que « C'est à mon avis une solution optimale sur le plan du *using process*, et à mon avis ça te fera gagner un temps considérable. Mais c'est pas évident à mettre en place à cause de la structure du système d'information, tu comprends ? ».

Après quelques temps, vous avez, vous et votre demandeur, grifonné au plus 2 ou 3 pages et mis sur papier des schémas illisibles dont vous oublierez la signification d'ici demain puisque vous n'allez pas les remettre au propre ce soir.

Spécifications

Vous écrirez par la suite un cachier des charges qui n'a rien à voir avec la demande. Remplissez celui-ci de nombreux schémas inutiles glanés sur le net, de textes écrits par « copier/coller » et de petits graphiques avec des données pifométriques, le tout pour faire du volume. N'imprimez pas en recto-verso, et utilisez une police 12 points (Le 14 ou plus faisant trop « livre pour enfant », vous risqueriez d'être découvert). N'oubliez surtout pas une grande page réservée à un schéma montrant « le gros bouton », ça fera plaisir au demandeur quand il feuilletera. Pensez à mettre un peu de texte crédible avant et après si jamais il lit un paragraphe ou deux.

En revanche, soignez particulièrement les deux seules parties lues : l'intro et la conclusion. Expliquez en intro que vous avez parfaitement compris le besoin et que vous avez une bonne solution écrite dans ce

cahier des charges. Ne dites pas que la solution est excellente, sinon il y aurait un rique pour que quelqu'un le lise.

En conclusion, expliquez, termes techniques barbares à l'appui, qu'il est parfaitement possible de répondre au besoin à condition de vous laisser du temps et des moyens. Laissez entendre que la partie n'est pas gagnée et que ce sera difficile mais rentable, surtout si on vous augmente.

Enfin, écrivez un sommaire bidon mais avec des termes qui font bien au cas où quelqu'un y jetterait un oeil.

Analyse et conception

Lancez-vous directement dans le développement d'un prototype et montrez-le de temps à autre au demandeur pour qu'il dise ce qu'il en pense. Profitez de cette phase pour faire semblant de travailler et poussez quelques fois des cris du genre « Arrrrgh ! Ça ne sera pas 100% sécurisé si on fait comme ça ! », en froissant et en jettant violemment une feuille de papier. Gardez un air préocupé.

Dessinez de beaux schémas de conduite de projet, comme le cycle en 'Y' (ou 2TUP). Si vous ne savez pas ce que c'est, utilisez un moteur de recherche et recopiez bêtement les schémas trouvés en modifiant quelques mots de la légende. Laissez apparents ces schémas sur votre bureau et arrangez-vous pour que vos supérieurs les voient.

Glandouillez et profitez de cette période.

Développement

Glandez pendant un bon mois, puis ajoutez un splash screen avec écrit « Version 1.0 bêta ». Continuez le développement à la Rache de votre prototype et montrez-le de temps à autre au demandeur pour qu'il dise ce qu'il en pense. Plaignez-vous de ses demandes incessantes pour justifier plus tard une rallonge de temps.

Allez lentement sur le développement de fonctionnalités et faites-les les unes après les autres. Votre demandeur vous demandera sûrement « Ouaaaah ! Mais elle fait ça ton appli ? C'est super ! Ça serait même encore plus génial si tu pouvais faire ça aussi... et puis ça... et puis... ça serait bien... tu comprends ? ». Or si vous développez plusieurs fonctionnalités en même temps, non seulement elles seront buggés jusqu'à l'os et inexploitables, mais en plus votre appli deviendra tentaculaire face à des demandes multiples.

Ecrivez un document d'avancement de projet bidon pour contenter le demandeur.

Tests et déploiement

En fait, commencez par le déploiement et faites utiliser l'appli par voter demandeur, comme vous l'avez tourjours fait depuis l'analyse / conception. Il suffit de faire comme d'hab, toujours dans l'optique d'un projet à la Rache. Si ça ne marche pas, revenez en arrière jusqu'à ce que ça donne une appli qui «à l'air de marcher ».

Support technique

Faites-vous vite muter de service ou démissionnez avant que les utilisateurs ne tombent sur les vrais bugs.

CONCLUSION

Bon, d'accord, puisque vous insistez, je vais quand même conclure.

Faire semblant est un art. Beaucoup de projets se font à la Rache aujourd'hui et finalement, si la méthode tend à se formaliser grâce à l'action capitale de génies de la Rache, de travailleurs archarnés et d'organisations structurées et crédibles comme l'IILAR, il manque à la Rache la formalisation d'une partie essentielle qu'est le camouflage. Aujourd'hui, la simulation de travail est nécessaire pour une bonne image de soi ou de l'entreprise que l'on représente. Mais elle passe avant tout par de bonnes compétences de *pipeau* (C'est-à-dire de génération d'explications pourraves), par des techniques avancées de modélisation comme les BITES de l'URML et des méthodologies concrètes de conduite de projet qui vous donnent l'air de bosser et qui vous permettent de ne rien foutre en étant payé (ou bien noté pour les étudiants).

Du même auteur, probablement à paraître (J'suis pas certain) :

- La Rache, une méthodologie réaliste mais formaliste : Tome 2 – La norme URML
- Le mastering de pipeau par l'exemple (ou « Génération d'explications foireuses »)