



Une Politique pour le Système d'Information

Descartes – Wittgenstein – (XML)

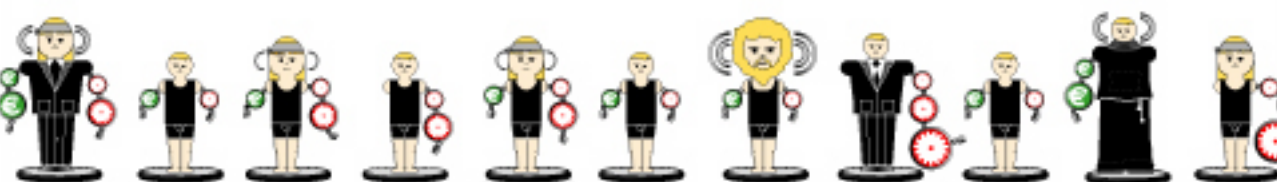


Table des matières

Préface	5
Le Cahier des Charges soumis au doute	6
Légende	17
Introduction	19
Les hommes	20
La frontière	21
La communication par pattern	22
L'incrémental	23
Le test	25
La fin de l'introduction	26
Les Hommes	29
Les opérationnels leaders	30
Natures et valences des individus	32
Qui sont vraiment les profils rares ?	37
Nature des individus et risques projet	44
Comment les profils multivalents participent-ils à la maîtrise des risques projet ?	50
Conclusion	54
La frontière	57
Rationaliser	60
Innover	66
Conclusion	73
La communication par pattern	77
La notion de pattern	78
Une structure pour cataloguer les patterns du SI	79
Documenter un pattern	83
L'analyse par pattern	95
La conception par pattern	99
Identifier et diffuser les patterns	106
Conclusion	107
L'incrémental	111
Faire l'économie du cahier des charges ?	112
Progression incrémentale cadencée	114
Conception incrémentale	121
L'incrémental s'applique aux fondations	133
Peut-on vendre des produits sur plan ?	139
Assembler rapidement sans sacrifier la qualité	143
Tests et progression incrémentale	144

Les Tests	151
Une révolution Copernicienne	152
Valoriser les tests	158
(Faire l')Economie des tests (?)	162
Une démarche outillée	168
Conclusion	179
Piloter la zone rationalisée	183
Le « Run » : enfin une vraie usine informatique ?	184
Perspective Client	190
Perspective des processus métier	195
Perspective financière	205
Perspective Apprentissage & Développement	210
Conclusion	214
Piloter en zones « chaos »	217
La dynamique « chaotique »	218
Perspective Client	231
Perspective Excellence opérationnelle	239
Perspective Finances	254
Perspective Apprentissage	259
Conclusion	260
La politique du SI en action	263
Une politique du SI et les architectures orientées service (SOA)	264
Une politique du SI et l'alignement stratégique, la gouvernance	265
Une politique du SI et l'informatique à la demande	266
Une politique du SI et l'Open Source	267
Une politique du SI et l'offshore	270
Conclusion	275
Bibliographie commentée	279
Index	285
Crédits	291
A propos d'OCTO Technology	292
Bibliographie OCTO Technology	293

Introduction

Le Système d'Information est une matière jeune mais déjà complexe. Optimiser, transformer, ou innover sont devenus des tâches délicates où le taux d'échec demeure excessivement élevé.

Depuis 1998, OCTO a participé à beaucoup de transformations de grands Systèmes d'Information. Notre positionnement historique est tourné vers la technologie, au service de la productivité des organisations. Cette racine nous la conservons car elle fonde notre capacité à agir. Néanmoins cette même technologie sert trop souvent d'alibi pour refouler des **problèmes plus structurels de gouvernance du SI**, que nous vous proposons d'aborder ici.

Nicolas Reimen nous le rappelle, les mots sont importants, et souvent pourtant, leur sens est détourné. Comme le cahier des charges qui n'existait pas, d'autres « vérités » absurdes structurent les mœurs de nos organisations dès qu'il s'agit d'informatique.

En réponse à ces vérités abstraites et sans matérialité dans le monde réel, nous tenterons de proposer cinq réalités observables, des situations dont chacun pourra recréer les conditions dans son entreprise, et qui permettront de rapprocher la vision des gestionnaires de celle des opérationnels dans un objectif commun de productivité.

Ces cinq piliers sont les *hommes*, la *frontière*, la *communication par pattern*, l'*incrémental* et le *test*. **Ils fondent une politique pour le SI.**

Dans la lignée de Wittgenstein, qui marque la fin du positivisme logique, plusieurs influences marquent cet ouvrage : l'approche structuraliste popularisée par Claude Levi-Strauss, les premiers patterns de l'architecte américain Alexander, le manifeste Agile du cercle de Kent Beck et Erich Gamma, ou la sociologie des univers techniques par Gerald Weinberg. Ces racines, nous les assemblons et les étendons au Système d'Information pour en tirer des conclusions à contre-courant de la pensée unique : externalisation, grands projets, urbanisme par la loi, profils à haute valeur ajoutée, gouvernance ...

Les hommes

Architecture, urbanisme, ouvriers du code .. la métaphore du BTP dans le monde informatique n'a rien produit que de faux clichés inspirés de Le Corbusier - rectitude des frontières, urbanisme glacial ...- ou propres à délocaliser des populations entières « d'ouvriers ». Or on ne construit pas des logiciels comme on construit des maisons. En BTP, tous les acteurs d'un projet sont unis par un lien invisible qui procède de leur compréhension uniforme de la finalité de ce qu'ils bâtissent, cette finalité étant stable dans le temps : vivre au chaud, optionnellement à la lumière ou protégé du bruit.

Dans le logiciel, ce *cœur* n'est pas forcément partagé, et bouge énormément. Vous imaginiez des ouvriers travaillant sur des plans univoques ? Observez plutôt des dialoguistes et des dessinateurs guidés par la trame générale d'un metteur en scène (l'architecte). Vous pensiez rentrer dans votre appartement au départ des peintres ? Retournez plutôt le sablier de la construction et patientez pendant qu'on teste votre duplex pour une durée équivalente.

Hier l'informaticien était majoritairement monovalent, tout à son affaire

informatique et sa complexité. Aujourd'hui les bons ingénieurs expérimentés sont largement bivalents, ouverts au métier : ce sont les effets naturels de nos gains de productivité, notamment liés à la standardisation des univers technologiques - IP, le web, le relationnel, l'objet - et la démocratisation de l'accès à l'expertise et l'expérience grâce aux communautés Internet.

Or les organisations n'ont pas évolué. Elles ont même amplifié les cloisonnements et la bureaucratie autour des logiciels. Ces opérationnels accomplis sont malheureux et improductifs dans des structures où la ligne de spécification précède la ligne de code.

La noblesse des métiers opérationnels de l'informatique n'a pas inspiré les gestionnaires du SI, embués par une vision statique héritée du BTP, ils les imaginent encore à l'usine ... donc déjà très loin.

La frontière

Il existe deux natures très différentes de projets dans le SI : ceux qui tendent à rationaliser et/ou à innover marginalement sur l'existant, et ceux qui expérimentent de nouveaux concepts métier, comme récemment le canal Internet ou la Gestion de la Relation Client. Les projets dans ces deux zones sont tirés par des enjeux bien différents : complexité, planification, respect de structures fortes dans le premier, et rapidité, gestion de l'incertain, pilotage par les délais dans le second.

Bien que tentative courante, adresser simultanément les enjeux de *l'innovation* et ceux de la *rationalisation* est impossible dans la réalité. La première des politiques du SI consistera donc à organiser des univers pour chacun, et d'instaurer un flux permanent entre les deux.

Réaliser des gains de productivité dans les zones de *rationalisation* impose de réduire la taille du SI : moins de code, donc moins de maintenance et moins de charge d'exploitation. Or un bloc de SI est souvent constitué d'une ou deux grosses applications historiques entourées de satellites spécialisés, reproduisant une partie de l'information présente dans les cœurs. Lorsque l'on dépouille un satellite de ces données et de ces interfaces pour le fusionner dans un cœur, il se produit un phénomène d'intégration marginale : $10 + 1 = 10,1$ en termes de coûts et toujours 11 en valeur produite.

Réussir une fusion nécessite cependant deux conditions : la stabilité d'un architecte opérationnel ou d'une génération d'architectes, et la stabilité des fondations de l'application *nova*. L'attraction de la *nova* est en effet directement liée à la confiance que peut inspirer son concepteur. Quant aux

fondations, elles conditionnent la masse critique que la *nova* peut atteindre : les technologies doivent y être éprouvées pour supporter la montée en charge du volume de données, du volume d'utilisateurs, ou du volume de tests. Sans tests automatiques, l'application devra stopper sa croissance puisque l'ajout de nouvelles fonctionnalités imposera au fil du temps des tests de non-régression de plus en plus lourds, induisant l'effet inverse de celui recherché.

Côté *innovation*, notre objectif n'est pas l'intégration marginale, mais précisément l'inverse : s'abstraire des contraintes inhérentes à la structure. Cela consistera en la mise à disposition de ressources informatiques indépendantes et reconnues pour réaliser ces applications « incertaines ».

Mais les innovations qui marchent devront tôt ou tard traverser la frontière pour être réintégrées dans des *novas* à moins de générer rapidement un SI parallèle et faiblement structuré. Le flux d'innovation fertilisera régulièrement le SI rationalisé dès que l'orientation principale de l'application passera de la rapidité et l'incertitude à la sécurité, l'intégration et la maîtrise des coûts.

La communication par pattern

On nous demande souvent notre avis sur des Systèmes d'Information. L'exercice consiste à analyser le travail de centaines de personnes, le patrimoine accumulé sur des dizaines d'années, ce qui s'avère être un travail difficile, sinon vain dans une approche exhaustive. Que faire d'un avis nanométrique dont on ne tirerait aucun levier d'action à un niveau macroscopique ? Pour éviter cet écueil, il faut substituer l'approche linéaire à la recherche ciblée de structures fondamentales et surtout récurrentes. Truisme me direz-vous ? Oui si l'on ne connaît pas à l'avance ces canevas de solution déployés face à des situations vues et revues. A l'inverse posséder un inventaire de ces *patterns*, caractéristiques récurrentes de nos SI, dont la portée s'étend de l'organisation, au fonctionnel et au technique, permet de zoomer là où des solutions élégantes ou au contraire inadaptées se déploient régulièrement.

La présence ou l'absence de patterns dans du logiciel fonde souvent sa qualité intrinsèque. Que diriez-vous d'une application de gestion *multi-entité*, c'est-à-dire capable de gérer plusieurs types de centres d'activité, par rapport à un logiciel *mono-entité* ? Pour deux logiciels qui auraient pu être cartographiés de manière identique (comptabilité, gestion back, gestion front, ..), vous tenez là l'essence de leur différence, et donc de leur valeur. Le logiciel est dit *multi-entité* car on l'a déployé dans des environnements variés, en résolvant de manière identique le problème récurrent de déploiement, typiquement par une capacité de paramétrage par entité.

Les patterns multi-X (multi-langue, multi-géographies, multi-clientèle ..) caractérisent toujours des logiciels rodés et à forte valeur ajoutée : le même code sert des besoins variés. Ce n'est pas pour autant qu'il faille conclure que seuls les logiciels multi quelque chose sont de qualité. Il est parfois souhaitable que des applications soient dédiées à des produits ou des clientèles, ayant leurs contraintes propres, souvent liées à des rythmes d'évolution différents. Deux clientèles, deux entités ont des rythmes de développement différents ? Mieux vaut développer deux applications.

Comme on identifie les bonnes pratiques répétées dans le temps, on peut aussi identifier les bêtises récurrentes qui font nos applications, on parlera d'anti-pattern. Pattern « 0 refactoring » : une erreur de design historique est pérennisée malgré les problèmes récurrents qu'elle pose, par exemple utiliser un numéro de carte SIM comme identifiant du client¹. Pattern organisationnel « Bunkerisation » : les projets se referment sous la pression et reproduisent X données ou services existants au prétexte qu'on ne pourra pas faire évoluer cet existant dans la même chronologie. L'anti-pattern Bunkerisation produit des SI balkanisés, redondants et peu maintenables, c'est pourtant un des plus présents dans nos organisations.

Susciter la distillation de ces patterns dans l'entreprise peut sensiblement améliorer la communication dans la *communauté SI*², et donc agir sur sa productivité.

Au-delà de limiter le volume d'informations nécessaire pour appréhender ce qu'est ou n'est pas le SI, le niveau de granularité du pattern fonde aussi les capacités d'action des dirigeants : renforcer certains patterns, corriger certains anti-patterns. Aller plus fin sert finalement peu : indépendamment d'être fausse, la carte au 25000^{ème} du SI n'aura pas beaucoup de lecteurs.

Le concept englobant de *pattern*, concentré d'expérience ou d'ignorance humaine, est essentiel pour comprendre et piloter son SI, car il fonde sa culture.

L'incrémental

Il existe un proverbe qui dit qu'un bon conseiller doit connaître ses proverbes. Derrière un proverbe il y a toujours une situation humaine récurrente, ce que nous avons décidé de désigner par *pattern*.

Rome ne s'est pas faite en un jour dit le proverbe.

Comment s'est faite Rome d'ailleurs ? Un grand monsieur est venu avec un

¹ Comme le nom propre ou l'adresse, la carte SIM ne permet pas d'identifier durablement une personne physique, qui peut se marier, déménager ou perdre son portable. Or le SI d'un opérateur doit pouvoir regrouper l'ensemble de ces interactions avec ses clients, ce qui nécessite d'utiliser un identifiant plus stable.

² Direction informatique, maîtrise d'ouvrage, innovateurs ...

plan détaillé des rues, des égouts, des parcs, des maisons, des immeubles et des lieux publics, on a téléphoné à Accenture, et ils ont construit Rome. Non bien sûr. Sinon Rome ressemblerait à Disneyland.

Au lieu de cela, Rome s'est construit sur des récurrences qui ont imprégné les différents incréments – dont certains destructifs - qu'a subi la ville : un fleuve, un bois, une pierre particuliers, des toits, des palais, des champs, des ponts, des bouts d'égouts, des canons, pas de métro, ...

Dans les SI de gestion, ceux qui automatisent des portions de processus humains, la complexité ne s'approprie pas avec un bazooka. Cette tentative de l'aborder d'avance (up-front) est prétentieuse puisque nos processus sont par essence flous, imparfaits et évolutifs avec le temps. Quand elle ne conduit pas à l'échec pur et simple, la démarche en *cascade* produit du logiciel lourd, déstructuré et donc difficilement maintenable.

Autant la métaphore du BTP est inutile pour décrire la construction de Systèmes d'Information, autant celle de la *littérature* nous aidera à mieux exprimer la réalité de nos métiers. Un livre peut s'écrire de deux manières : chapitre par chapitre, puis une dernière passe pour corriger et raccorder l'ensemble, ou bien dans sa globalité à un niveau général, puis rédaction des chapitres dans un ordre quelconque. Il est des livres dont on perçoit l'architecture, d'autres non. Les premiers vous marquent, c'est le cas par exemple chez Kundera, grand écrivain situationniste, dont la prose n'est pas logorrhée informe, mais structure mettant en scène des personnages pour mieux exprimer une situation qui se reproduira, un *pattern* : l'insoutenable légèreté de l'être, l'impossible retour d'exil, ...

Ainsi se résume l'essence de l'approche incrémentale. L'appliquer au logiciel signifie être capable de se lancer dans le code en acceptant de ne pas avoir écrit son histoire dans le détail, mais seulement dans son ensemble. Pour cela, il faut maîtriser la *peur du brouillard*. Car le réflexe de la *cascade* (cahier des charges complet, spécifications détaillées complètes, ..) révèle la volonté de diminuer cette peur en chassant le *brouillard*. Pourtant cette stagnation sur l'écrit contribue à l'effet inverse, elle augmente la peur en faisant enfler dans le détail un objet que souvent plus personne ne peut comprendre dans son ensemble.

A l'opposé, la culture *incrémentale* propose de réaliser le logiciel par incréments, c'est-à-dire en résolvant rapidement une situation simple, mais de bout en bout. Puis une deuxième situation qui le complexifiera, et ainsi de suite. A chaque itération, le logiciel propose des fonctionnalités intègres, qui nous permettront de mettre en *situation* un utilisateur et récolter régulièrement son feedback.

Cette approche ne peut se concrétiser que dans un environnement où le contrat

fait place à la confiance, où l'on pense partenaire quand on parle de fournisseur, dans la même logique de nos confrères industriels et leur principe reconnu du Keiretsu, intégration d'entreprises autour d'un objectif commun, d'intérêt partagés qui se déroulent sous une direction commune.

Le test

Le mythe du cahier des charges a la dent dure. L'écrit semble tellement à la fois « vrai » et univoque, sécurisant comme un beau contrat. Or il n'a aucune de ces propriétés. Le vrai, quand il existe, ne s'atteint que par les tests concrets (Wittgenstein).

Si la conception générale est un art complexe qu'il convient d'aborder de manière incrémentale, la conception détaillée des logiciels de gestion est clairement un art vain. Car seul ce qui permet de minimiser l'effort futur de modification du code produit de la valeur. L'interprétation radicale de méthodes comme CMM³ alimentent une bureaucratie en documents à la valeur ajoutée très volatile, aisément déplaçable dans le code. Une batterie de tests opérationnels et du code commenté constituent une assurance bien supérieure au papier, qui n'est jamais utilisé dans la maintenance.

La surface de nos logiciels ne cesse de croître, et, qu'on le veuille ou non, les fonctionnalités qu'ils proposent s'imbriquent copieusement. Et c'est bien ainsi, un logiciel qui ne verrait pas toutes ses parties fortement couplées à un *cœur*⁴ ne serait qu'un tas de code informe, donc faiblement maintenable. Toutes ces fonctionnalités imbriquées doivent être retestées, avec un contexte toujours plus riche. Sur ces phases aval, nous avons peu investi. Je ne parle pas de logiciel miracle, mais d'investissement réel de toutes les forces opérationnelles – de la capture des exigences à la recette –, c'est-à-dire de *développement piloté par les tests*. Du code réutilisable pour tester le code représente le principal gisement de productivité dans la maintenance, et se matérialisera par la réduction sensible des phases d'homologation, synonyme de capacité de production en cycles courts.

³ Capacity Maturity Model : méthode de certification du processus logiciel n'incluant pas de propositions opérationnelles pour l'améliorer.

⁴ Le cœur constitue la structure la plus reproduite dans le logiciel, par exemple une grappe d'objets fondatrice, une implantation systématique de traitements (sécurité, collecte d'informations, validation, trace, diffusion).

La fin de l'introduction

Ce livre a été construit dans une approche incrémentale, et vous venez d'en lire le niveau général : 5 piliers en 5 pages.

La suite va détailler et illustrer ces 5 piliers, puis observer leur mise en place de part et d'autre de la frontière, définissant ainsi une Politique du Système d'Information. Le tout avec des exemples concrets issus de l'expérience d'OCTO, des quizz, une **carte des connaissances** en couleurs, des photos de femmes nues dissimulées entre les lignes de code, ..

Excellente lecture.

Pierre Pezziardi
Directeur Technique

De l'action

Lors d'une classique entrevue commerciale avec le Colonel Serge Groquépi, responsable de la coordination des Gendarmeries du Sud-Ouest, il prit le risque, au cours de la conversation, de lui montrer son petit prototype bricolé hâtivement pour illustrer l'avenir du centre d'appel intégré au Système d'Information. Et, contre toute attente, il fit mouche ! Le Colonel, raidi par l'excitation, lui lâche : « mais ton truc là, ça pourrait enregistrer des rapports d'intervention ? ... »

Du suspens

Caroline dispose d'un savoir-faire informatique et métier assez léger, acquis depuis deux ans sur des projets en tant qu'analyste. Elle a été choisie pour sa fiabilité passée, notamment sa capacité à reporter à ses supérieurs et ses qualités rédactionnelles. Cette première expérience peut façonner son développement futur... A suivre ...

De la pensée académique

Ainsi une frontière marque les contours d'une culture. Les contours historiques du SI sont eux généralement fondés sur les métiers de l'entreprise. Bien qu'il existe des particularités culturelles liées à ces métiers - front-office, back-office, métiers de support - les frontières les plus marquées ne se situent pas à cet endroit...

Du politiquement incorrect

Comment s'est faite Rome d'ailleurs ? Un grand monsieur est venu avec un plan détaillé des rues, des égouts, des parcs, des maisons, des immeubles et des lieux publics, on a téléphoné à Accenture, et ils ont construit Rome. Non bien sûr. Sinon Rome ressemblerait à Disneyland...

Des solutions pour changer

Quand faut-il tester ? Ni avant, ni après, mais pendant, tout le temps. Les tests ne sont pas une phase du projet, ils l'accompagnent durant toutes ses phases. La notion de phase disparaît du paysage ; elle rétrécit au point de se dissoudre dans le processus continu que nous appelons développement itératif incrémental...

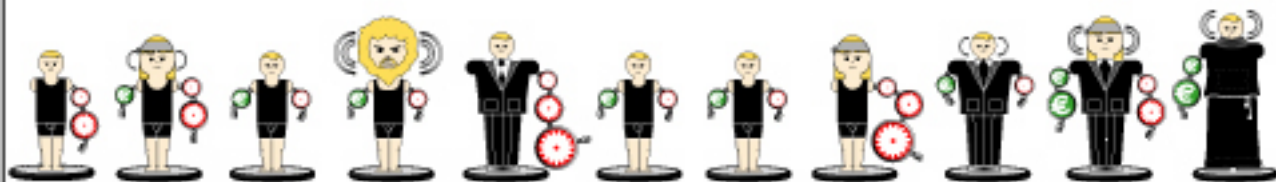
De la technologie

La fusion de fondations est aussi connue sous l'acronyme Enterprise Application Platform (EAP), et poussée par des éditeurs mastodontes comme Microsoft, BEA, Oracle ou IBM. Ils proposent au sein d'une « suite intégrée » la panoplie complète d'outils destinée à gérer la sécurité, les interfaces homme machine, le transactionnel, les batchs, les données, ou encore les échanges. En ce qu'elles veulent embrasser très large, ces initiatives peinent encore à mûrir et s'imposer...

Des sciences (quasi) exactes

La valeur métier va mesurer un coût de remplacement, c'est-à-dire combien coûterait en personnel et en perte de qualité la reprise du travail de l'application non pas à la main, mais sur Excel ou Access. Lorsqu'une telle « manualisation » est impossible, c'est-à-dire pour certains systèmes à visibilité externe (site Internet, service automatisé non-crédible sans informatique, ...), la valeur métier plafonnera à la valeur du marché en question...

Une politique pour le SI, un ouvrage pour toute la famille !



ISBN : 2-9525895-0-X

OCTO Technology, Cabinet d'Architectes en Système d'Information
50, avenue des Champs-Élysées - 75008 Paris
www.octo.com