Organisation de grands projets de développement

Emmanuel CHENU
emmanuel.chenu@fr.thalesgroup.com
http://emmanuelchenu.blogspot.com

Cette note porte sur l'organisation d'un grand développment logiciel éventuellement multisite et sur plusieurs années. L'objectif est d'identifier des propositions d'organisation dont la mise en oeuvre vise à minimiser le gaspillage dans un tel contexte de développement. Toutes les propositions sont accompagnées d'une justification. Il s'agit d'une synthèse basée sur l'expérience acquise et sur plusieurs ouvrages et articles portant sur le développement logiciel.

LES DIFFERENTES MANIERES D'ETRE UN GRAND PROJET

Un projet de développement peut être considéré grand par sa durée *(sur plusieurs années)*, par la taille de l'équipe, par la compléxité du problème à résoudre, par la complexité des solutions proposées ou par la complexité des contraintes[1].

LES PERILS DE LA GRANDE ORGANISATION MULTISITE

Plusieurs sources s'accordent pour affirmer que l'équipe idéale de développement est composée de sept plus ou moins deux membres réunis sur un même site[2]. En dehors de ce contexte idéal, les équipes souffrent des mêmes freins à l'efficacité: perte de communication dans l'équipe, moins de réactivité et non-homogénéité du logiciel[3]. Les propositions d'organisation identifiées ont pour objectif de pallier à ces difficultés.

OBJECTIF: PRIVILEGIER LA COMMUNICATION

La communication est ce qui importe le plus au sein d'une équipe de développement. Les problèmes rencontrés lors d'un projet sont souvent dus à des manques d'échange[4]. Alors, il s'agit d'organiser le développement de manière à inciter une communication intense et en continue au sein de l'équipe y compris avec le client.

Tous les membres le l'équipe auront beaucoup à apprendre. Pour certains, il faudra apprendre le domaine du produit à développer. Pour d'autres, il faudra apprendre le langage ou les pratiques de développement logiciel. Ainsi, l'organisation doit faciliter l'apprentissage, notamment en privilégiant la communication au sein de l'équipe.

La stabilité de l'équipe est un objectif – mais elle ne sera pas assurée. Il faut donc faciliter l'intégration de nouveaux membres et ne pas perdre des connaissancs acquises par l'équipe. La encore, la communication doit être privilégiée pour captialiser ce que l'équipe apprend et pour le transmettre à de nouveaux intervenants.

Consigner toutes les décisions avec leur justification.
Certaines décisions importantes et justifications associées sont consignées et datées dans:

- Un plan de développement lorsqu'elles concernent
  - l'organisation et le processus;
  - la stratégie de test.
  - les pratiques de développement, ateliers et outils.
- Une description de la conception lorsqu'elles concernent l'architecture.
- Une main courante pour toutes les autres décisions.

Cette pratique permet de conserver les raisons qui ont amené les choix. Un tel historique argumenté facilite l'intégration de nouveaux intervenants et évite la perte d'information lorsque des membres quittent l'équipe.

L'équipe de développement partage un espace commun.

1 Voir [BecAnd]
2 Voir [SchwaBee] et [Chaos]
3 Voir [Broo] et [SchwaBee] et [BecAnd]
4 Voir [Broo] et [SchwaBee] et [BecAnd] et [HunTho]

Cet espace peut être organisé en plusieurs bureaux. Il doit contenir des bureaux adaptés au travail en équipe.

Une telle organisation incite la communication, le formation des membres de l'équipe, le travail en binômes et la management visuel du projet.

L'équipe est pluridisciplinaire[5].
Toutes les compétences nécessaires à la réussite du projet doivent être intégrées à l'équipe. L'équipe ne doit pas être organisée par spécialité. L'organisation de la comptabilité du projet ne doit pas avoir de répercussion sur la collaboration de l'équipe.

Cette pratique facilite la diffusion des connaissances au sein de l'équipe.

L'équipe se réunit brièvement tous les jours à heure fixe et en lieu fixe.
L'objectif est que l'équipe communique son avancement, ses engagements et les obstacles qu'elle rencontre[6].

Cette pratique permet de gagner en visibilité et en réactivité.

Partager l'information.
L'information doit être consignée et partagée au moyen d'outils pragmatiques, légers et ergonomiques.

Les outils lourds et difficiles à utiliser sont vite abandonnés par leurs utilisateurs. Les outils peuvent être informels (comme des post-its) ou plus formels mais légers et accessibles comme des wikis.

OBJECTIF: PRIVILEGIER LE RETOUR RAPIDE D'INFORMATION

Le changement est inévitable et continu lors d'un développement logiciel. Plutôt que de gaspiller de l'énergie à contenir le changement, de nombreuses sources proposent de s'y adapter[7]. Un retour d'information rapide et en continue sur le développement permet alors de piloter le travail de manière réactive et de l'adapter au changement.

Le développement itératif et incrémental, le pilotage par les tests et l'intégration continue sont des pratiques fournissant un retour rapide d'information. Etant déjà largement appliquées, elles ne serot pas traitées davantage.

Management visuel.
L'avancement et les difficultés du projet sont affichés dans l'espace de travail.

Cette pratique facilite la communication et le retour d'information.

La réunion qotidienne fournit un retour rapide d'information sur l'avancement et les problèmes rencontrés.

Production quotidienne d'indicateurs.
Des indicateurs d'avancement et de qualité du logiciel sont automatiquement synthétisés quotidiennement.

La synthèse des indicateurs peut être produite par le script d'intégration continue. Ils fournissent un état objectif et à jour de la santé du projet et du logiciel. Les indicateurs doivent être communiqués en les affichant dans l'espace commun de travail et doivent être commentés lors de la réunion quotidienne afin de proposer d'éventuelles actions correctives. Ainsi, ils contribuent à la réactivité de l'équipe.

Retrospectives régulières.
Régulièrement, l'équipe se réunit brièvement pour prendre du recul sur ses pratiques et rechercher des axes d'amélioration.

Cette démarche fournit un retour d'information régulier sur la manière de travailler de l'équipe. Elle est une base de l'amélioration continue des pratiques de développement.

OBJECTIF: PRIVILEGIER LA SYNCHRONISATION

5 Voir [BecAnd] et [SchwaBee]
6 Voir [SchwaBee] et [Yip] et [BenBosMedWil]
7 Voir [BecAnd] et [SchwaBee] et [Broo]

Les grandes équipes de développement – éventuellement multisites - perdent souvent en efficacité car elles ne sont pas synchronisées: elles travaillent sur des bases de code et des versions de code différentes. L'objectif est donc de s'organiser de manière à ce que tous sur synchronisent très régulièrement sur une unique base.

Partager une unique base d'artifacts.
Code, tests, documents, plans, justifications et autres données sont gérés dans une unique base partagée par toute l'équipe.

Il faut éviter toute dupplication d'information[8].

L'équipe se synchronise aussi régulièrement que possible sur l'unique base d'artifacts.
Le travail est délivré aussi souvent que possible. Le travail est re-synchronisé aussi souvent que possible.

L'intégration continue est une application de ce principe.

OBJECTIF: DEVELOPPER EN FLUX TENDU

Le gaspillage croit avec la dimension des équipes et avec la perte de communication liée au travail multisite. Afin de minimiser ce gaspillage, il s'agit alors de développer en flux tendu de la spécification à la vérification, notamment en appliquant un cycle itératif et incrémental avec intégration continue[9].
Pour développer en flux tendu de manière fluide et rapide, il faut décrire le flux complet de travail en intégrant tous les métiers, identifier le principal goulot d'étranglement, le réduire puis réitérer la démarche régulièrement au fil du développement[10].
Cette démarche peut être outillée au moyen d'un plan du flux de travail et d'un Kanban[11].

OBJECTIF: PRIVILEGIER L'ADAPTATION

Le changement est inévitable et continu lors d'un développement logiciel[12]. Ce constat est désormais acquis pour la construction du produit puisque la construction incrémentale d'une architecture émergente se standardise. Il reste à transformer ce constat en pratiques pour l'organisation qui produit le logiciel.
La loi de Conway prédit que les organisations qui concoivent des systèmes les réalisent à l'image de ses propres structures. Conway remarque qu'une première conception est presque immanquablement pas la bonne. La conception doit changer et s'adapter. Il en est donc de même pour l'organisation qui produit la conception: la première n'est pas la bonne et elle doit changer pour s'adapter[13].

On a compris qu'il ne fallait consigner dans un plan de l'architecture que des principes et les décisions importantes et stables pour laisser une liberté de flexibilité. De même, il ne faut consigner dans les plans d'organisation que les principes, les décisions importantes et stables pour permettre l'expérimentation et la l'adaptation.

L'organisation doit s'adapter au long du développement.

Les retrospectives régulières offrent un cadre pour réfléchir à l'amélioration continue de l'organisation du travail.

OBJECTIF: PRIVILEGIER L'HOMOGENEITE DU PRODUIT

Afin de faciliter les inévitables modifications et l'intégration de nouveaux intervenants, le logiciel doit sembler être écrit par une seule et très compétente personne[14].Il s'agit de préserver l'intégrité conceptuelle du produit[15]. Cette homogénéité est très souvent perdue au fil des années de développement, lorsque l'équipe est répartie sur plusieurs sites, lorsqu'elle dépasse la taille idéale (7+/-2) ou lorsqu'elle est même

8 Voir le DRY (Don't Repeat Yourself) dans [HunTho]
9 Voir [Pop1] et [Pop2]
10 Voir [Pop1] et [Pop2]
11 Voir [Pop1] et [Pop2]
12 Voir [Broo] et [SchwaBee]
13 Voir [Broo]
14 Voir [BecAnd]
15 Voir [Broo] et [Pop1]

partiellement renouvellée. Il s'agit donc d'entretenir un standard de développement[16] et de vérifier qu'il est continuellement appliqué, notamment grace au travail en binômes[17]. Ce standard doit évoluer, tout comme l'architecture du produit et l'organisation qui le construit.

OBJECTIF: MAITRISER LA MONTEE EN CHARGE

Désormais la montée en charge des capacités d'un système se fait couramment de manière itérative et incrémentale. Il devrait en être de même pour l'organisation qui produit le système.
Il s'agit de construire un premier incrément fonctionnel du système avec un premier incrément d'organisation comprenant un premier incrément d'équipe de dimension idéale (7+/-2). Cette première organisation a pour objectif de produire une première version utilisable du produit et de mettre en place une permière version des pratiques et des outils. Ce premier incrément doit fournir un retour rapide d'information sur le produit et l'organisation afin de les adapter pour une montée en charge progressive du produit et de l'organisation[18].

Construire l'organisation (et l'équipe) de manière itérative et incrémentale.

De nombreuses sources affirment que la taille idéale d'une équipe est de sept membres plus ou moins deux[19]. Si cette taille critique vient à être dépassée, il convient de constituer plusieurs équipes, chacune respectant la règle 7+/-2[20]. Chaque équipe pluridisciplinaire doit respecter les pratiques de développement – y compris les pratiques de communication et de retour rapide d'information. En plus d'être déployées au sein de chaque équipe, ces dernières sont également déployées entre les équipes[21]. Ainsi, chaque équipe se réunit quotidiennement puis des représentant de chaque équipe de réunissent pour se syncroniser.

Une découpe en sous-équipes ne doit pas se faire sur des critères de spécialité ou d'architecture du produit (comme une équipe par couche par exemple) mais en terme de forte cohesion fonctionnelle et de faible couplage, comme pour la modularisation du produit[22]. Le faible couplage entre les équipes n'a pas pour objectif de réduire la communication, mais plutôt de réduire les zones d'intervention commune. Les équipes devront donc être réorganisées au fur et à mesure que l'architecture du produit émerge.

Organiser les équipes en petits groupes sur des critères de cohésion fonctionnelle et de faible couplage.

REUTILISER DES PATTERNS

Tout comme il existe des patrons de conception, il existe des patrons d'organisation. Scrum est un exemple d'organisation répandue[23].

BIBLIOGRAPHIE

- [BecAnd] Kent Beck and Cynthia Andres – Extreme Programming Explained, embrace change – ISBN 0-321-27865-8
- [BenBosMedWil] Jean-Louis Bénard, Laurent Bossavit, Régis Medina et Dominic Williams – Gestion de projet eXtreme Programming – ISBN 2-212-11561-X
- [Bro] Frederick P. Brooks, Jr – The Mythical Man-Month – ISBN 0201835959
- [Chaos] CHAOS: A Recipe for Success – The Standish Group International, inc. 1999
- [Cus1] Michael A. Cusumano – How Microsoft Makes Large Teams Work Like Small Teams – SLOAN Management Review/Fall 1997
- [Cus2] Michael A. Cusumano and Richard W. Selby – How Microsoft Builds Software – Communications of the ACM 1997
- [Fow] Martin Fowler - Using an Agile Software Process with Offshore Development -

16 Voir [Pop1] et [Pop2]
17 Voir [BecAnd]
18 Voir [SchwaBee] et [BecAnd]
19 Voir [BenBosMedWil] et [SchwaBee]
20 Voir [SchwaBee]
21 Voir [SchwaBee]
22 Voir [HunTho]
23 Voir [SchwaBee]

http://martinfowler.com/articles/
- [HunTho] Andrew Hunt and David Thomas – The Pragmatic Programmer, from journeyman to master – ISBN020161622X
- [Pop1] Mary et Tom Poppendieck – Lean Software Development, An Agile Toolkit – ISBN 0-321-15078-3
- [Pop2] Mary et Tom Poppendieck – Implementing Lean Software development, From Concept to Cash.
- [SchwaBee] Ken Schwaber and Mike Beedle – Agile Software Development wih Scrum – ISBN 0-13-067634-9
- [Yip] Jason Yip - It's Not Just Standing Up: Patterns of Daily Stand-up Meetings - http://martinfowler.com/articles/