



## Fiche projet - EGPI

AFF065-2009-NOT337

\$Rev: 494 \$

2011-04-03 13:22:58+02:00

Sébastien DEVAUX

dexter@detexia.com

**Résumé :** EGPI est une démarche de gestion de projet visant à permettre un développement agile même dans un environnement contraint par un formalisme fort et des règles strictes comme on les rencontre dans les systèmes embarqués et l'aérospatial. Elle repose sur une utilisation systématique d'un système de contrôle de version qui devient lui-même une base de donnée qui concrétise non seulement la gestion de configuration mais aussi la gestion de projet intégrale en passant par la gestion des faits techniques.

**Mots-clé :** gestion de configuration, gestion des faits techniques, gestion de projet

## Table des matières

1 Motivation.....	1
2 Intégration des gestions des faits techniques et de configuration.....	1
3 De la gestion des faits techniques à la gestion de projet.....	2
4 Mise en oeuvre.....	3
Table des URL .....	4

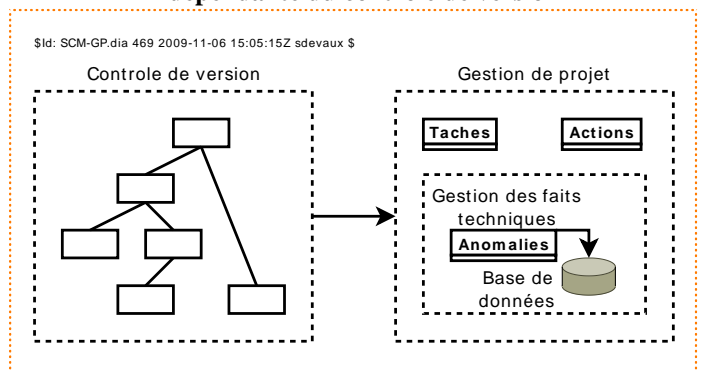
### 1 Motivation

EGPI<sup>[url1]</sup> est né d'un constat : sur des projets complexes et de grande envergure, il devient au fur et à mesure de l'ancienneté accumulé de plus en plus difficile d'en comprendre l'évolution, donc d'en assurer la maîtrise. Même quand les gestions de projet et de configuration sont menées rigoureusement selon les règles de l'art, on constate souvent de grandes difficultés à restituer correctement l'histoire du projet. Ceci pour une raison relativement simple, des éléments historiques sont définitivement perdus car trop d'informations relatives au projet ne sont pas confiées à un système de contrôle de version fiable.

La méthode de développement agile<sup>[url2]</sup> préconise de réaliser ce dont le client a réellement besoin au lieu de réaliser strictement ce qu'il a spécifié. Cela implique qu'on autorise une certaine évolution de la définition du produit au cours de son développement et du périmètre des tâches

qui commandent le déroulement du projet. Ceci n'autorise pourtant pas à faire n'importe quoi n'importe quand, surtout dans des environnements contraints comme l'aérospatial ou l'embarqué. Il est par conséquent nécessaire de bien conserver la trace de tous les changements et de toutes les actions accomplies durant le cycle de développement, c'est à dire non seulement les modifications effectuées sur la définition du produit mais aussi tout ce qui les a motivées.

**Fig.1 : Gestion de projet indépendante du contrôle de version**



### 2 Intégration des gestions des faits techniques et de configuration

La gestion de configuration permet à tout moment de restituer un état antérieur et de le comparer à l'état actuel ou un autre état antérieur. La comparaison de deux états permet d'identifier les

changements qui sont à l'origine d'effets (indésirables ou non) particuliers. Cependant, plus on remonte loin dans le passé, plus il est difficile de se remémorer les raisons qui ont motivé chaque changement.

Ces raisons sont une information capitale pour établir la pertinence de chaque changement, surtout dans le cadre de corrections d'anomalies. C'est pourquoi il est indispensable de conserver la traçabilité entre les changements et leur origine. Cette stratégie est déjà assez bien ancrée dans les chantiers de stabilisation et de chasse au bug. De nombreux outils sont déjà disponibles pour assister et guider les développeurs dans cette tâche. Ce sont les fameux "Bug Trackers", dont la plupart proposent des outils plus ou moins sophistiqués pour interagir avec une gestion de configuration. Pour améliorer la globalité du processus de développement, rien n'empêche de généraliser cette bonne pratique à l'intégralité du cycle de développement. La gestion des anomalies devient alors la gestion des faits techniques.

Un fait technique est une motivation d'accomplir des changements sur les articles de configuration. C'est une fiche dotée d'un identifiant unique qui décrit un but, une action, un travail à accomplir. Les fiches d'anomalies constituent alors une classe particulière de faits techniques.

Un premier niveau d'intégration repose sur la faculté de la plupart des outils de gestion de configuration, d'attacher des attributs ou un journal (log) à chaque action, c'est à dire à toute modification enregistrée sur les objets gérés en configuration. Il suffit d'y inclure l'identifiant du fait technique à l'origine du changement. En examinant l'historique d'un objet, on connaît ainsi immédiatement, par le biais des références aux faits techniques, pourquoi chaque modification a été entreprise. Selon la bonne volonté des utilisateurs ou pourra adjoindre des automatismes plus ou moins contraignants pour encourager, voir forcer, l'association d'un fait technique à tout changement de la configuration. Quand la traçabilité est bien établie, on peut également reconstituer la relation inverse, c'est à dire pour un fait technique donné, retrouver précisément tous les changements faits en son nom.

Les systèmes de gestion de faits techniques classiques reposent pour la plupart sur une base de données relationnelle où chaque fiche y est décrite de façon plus ou moins rigide et on arrive vite à un système qui repose trop sur des conventions d'utilisation de champs commentaires texte libre avec lesquels il est difficile de guider l'utilisateur pour une saisie pertinente et efficace des informations. De plus, un fait technique peut varier au cours du temps, au minimum son état évolue selon son cycle de vie (typiquement il passe de soumis, à en cours, validé, livré, clos...). L'historique de ces changements propres n'est pas toujours bien conservé ce qui est gênant surtout quand des objectifs du fait technique sont revus et remis en cause. Un moyen de résoudre cette limite que les bases de données sous-jacentes ne peuvent prendre en charge est simplement de décrire les faits techniques par des fiches elles même gérées en configuration. Le fait technique devenant un objet de configuration comme un autre, on obtient un deuxième niveau d'intégration qui rend la gestion de configuration auto-suffisante puisque la gestion des faits techniques y est alors totalement incluse. Ce n'est plus un outil annexe qui interagit avec la gestion de configuration avec plus ou moins de bonheur et les risques de désynchronisation inhérents à un tel couplage relativement faible. On conservera éventuellement un système de gestion de base de donnée

classique, uniquement pour accélérer l'indexation des bases de faits techniques très volumineuses.

### 3 De la gestion des faits techniques à la gestion de projet

Après avoir généralisé les fiches d'anomalies en faits techniques en interaction directe avec la gestion de configuration, poursuivons le processus. Dans une gestion de projet courante, on retrouve :

- des jalons : des objectifs précis à atteindre.
- des tâches.
- des risques : des points techniques à surveiller avec plus d'attention.
- des actions qui forment une liste de choses à faire et à ne pas oublier.
- ...

Tout comme les rapports d'anomalies, toutes ces entités ne sont en fait que la description plus ou moins succincte d'un travail à faire. Elles ont des attributs communs :

- Un identifiant
- un titre, une description succincte
- des attributs de planification : contraintes de réalisation et dates d'échéance.

C'est à dire qu'on peut toutes les qualifier de faits techniques si on considère que ces derniers sont l'abstraction la plus large de ces concepts. Si tout est fait technique, plus rien ne s'oppose à ce que toute action sur la configuration soit motivée par un fait technique. De même tout document, ou tout fichier, quel qu'il soit, produit dans le cadre du projet a sa place dans la configuration puisque sa réalisation est obligatoirement commandée par un fait technique.

Les faits techniques sont typés afin de pouvoir continuer à distinguer une action, d'un jalon ou d'une anomalie. On inclura également des relations entre faits techniques :

- des relations de dépendances : la réalisation d'un fait technique est conditionnée par l'évolution d'autres faits techniques.
- une relation hiérarchique : le regroupement de faits techniques par lot facilite la gestion macroscopique du projet. La relation hiérarchique implique une relation de dépendance. Un fait technique ne peut être achevé que lorsque tous ses sous faits techniques le sont.

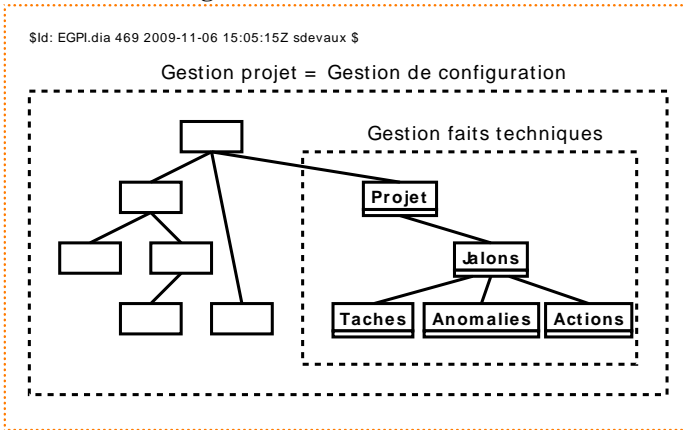
Par défaut on aura toujours au moins trois niveaux hiérarchiques :

- Projet : on a besoin d'un fait technique racine, qui pour une meilleure consistance sera le seul fait technique sans parent.
- Jalons : les objectifs de livraisons intermédiaires et finales.
- Tâches, actions et anomalies : description détaillée d'un travail élémentaire.

Un instantané des relations hiérarchiques constitue la planification courante. Reporter ou anticiper une tâche consiste à réaffecter son jalon parent. Si pour chaque fait technique on a réussi à renseigner ses estimations de coût et de réalisation :

- pour les jalons uniquement date de fin au plus tard (en théorie, seul la date de fin au plus tard du dernier jalon devrait suffire).
- pour les tâches la charge requise.

**Fig.2 : Gestion de projet intégrale sous contrôle de version**



L'exploitation des relations pourra conduire à un calcul de planning. Dans l'idéal il faudrait quand même adjoindre un solveur ayant connaissance de la disponibilité des ressources. A nouveau, l'intérêt de gérer en configuration toutes ces informations est de pouvoir observer leur évolution et se rappeler de façon fiable des états antérieurs et de façons synchronisé avec tous les autres documents du projet (spécifications, documents techniques, plans, sources logiciels, ...). La synthèse de toutes les notions projet (actions, tâches, anomalies) permet en plus d'éviter la surcharge provoquée par la gestion de multiples listes de choses à faire plus ou moins redondantes et réduit alors le risque de voir l'une d'entre elle être oubliée car égarée dans une liste qui n'est pas consultée régulièrement. Leur centralisation en gestion de configuration facilite l'établissement de tableaux de bord par croisement de toutes les informations qu'ils contiennent sans les effets néfastes d'une centralisation excessive puisque par nature un système de gestion de configuration est aussi système de synchronisation et de réplication d'une structure de fichiers.

## 4 Mise en oeuvre

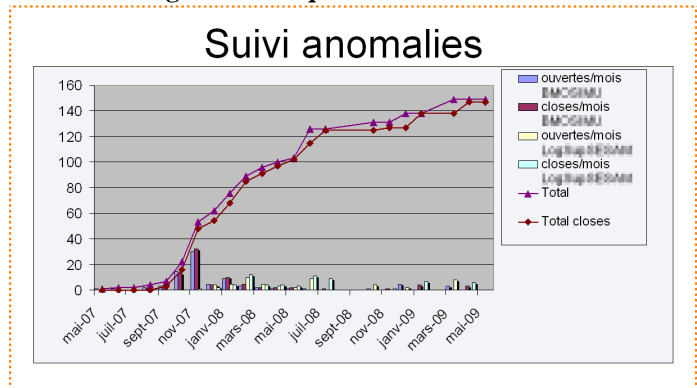
Detexia a choisi de mettre en oeuvre ce principe de la façon la plus légère possible afin de l'éprouver rapidement en usage réel. Cette mise en oeuvre s'est déroulée jusqu'à aujourd'hui en trois phases :

- Phase 1 : expérimentation sans aucun développement.  
Dans le cadre d'une affaire d'ampleur limitée (durée de 4 mois pour 3 développeurs), l'expérimentation a consisté par une application de la méthode par le biais de simples règles communes établies lors de la constitution de l'équipe :
  - Confier l'intégralité des sources et de la documentation projet à un système de contrôle de version. Nous avons choisi l'outil subversion <sup>[url3]</sup>
  - La base de faits techniques a pris la forme d'une simple feuille de calcul qui, comme tout document du projet, est enregistrée dès sa création alors qu'elle est encore vierge de faits techniques sous contrôle de version.
  - Toute action de modification d'un document et d'un fichier requière la création préalable d'un fait technique, sauf pour la base de faits techniques elle même.
  - Tout enregistrement dans le système de contrôle de version (action "commit" ou "checkin")

nécessite un commentaire structuré selon une convention établie au démarrage du projet pour déclarer l'identifiant du fait technique qui justifie les changements soumis.

L'expérimentation a montré le potentiel de la démarche mais a mis en lumière deux difficultés majeures. Premièrement une simple feuille de calcul est insuffisante et peu pratique pour mener une bonne gestion de faits techniques dès que le projet prend un peu d'ampleur et accumule un certain historique. La prise en charge des relations entre faits techniques par de simples annotations est insuffisante. Deuxièmement, la traçabilité n'est pas suffisamment fiable quand elle repose uniquement sur des conventions que seule la bonne volonté humaine est chargée d'appliquer. Enfin si la démarche a permis d'établir malgré tout une traçabilité acceptable, il est devenu évident en observant l'historique d'un article de configuration de connaître les tâches qui ont déterminé son évolution, il est encore trop fastidieux de faire l'inventaire des changements accomplis pour un fait technique particulier, même si l'opération est toujours possible.

**Fig.3 : Statistiques - Tableau de bord**

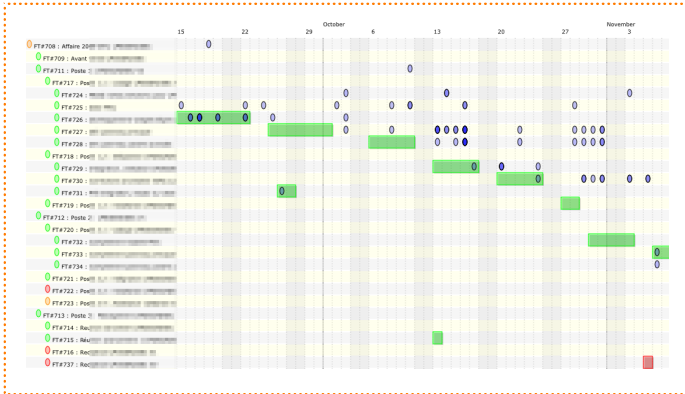


- Phase 2 : amélioration de la fiabilité par une interface utilisateur légère.  
Grâce au succès du premier projet qui nous a permis de remporter de nouvelles affaires qui lui ont succédé, il a été entrepris un développement spécifique axé principalement sur une interface utilisateur proposant des formulaires d'édition de faits techniques et un gestionnaire d'espace de travail qui masque le système de contrôle de version sous-jacent. L'outil produit possède les caractéristiques suivantes :
  - Logiciel multi-plateforme en Java
  - Chaque fait technique est enregistré sous forme d'un fichier XML, toujours sous contrôle de version. Cela permet ensuite l'adjonction rapide d'outils personnalisés à l'aide de feuille de style XSL pour par exemple produire des tableaux de bord (fig.3), des planning (fig.4), ou des fiches de version.
  - Les actions de modification de la configuration propagent une mise à jour automatique des faits techniques associés : les changements accomplis sous couverts d'un fait technique donné sont directement consignés en lui-même ce qui permet de restituer directement la traçabilité dans les deux sens.

Lors de sa mise en oeuvre, l'outil a permis de combler les lacunes du système précédent. La conservation de la

même base de contrôle de version subversion a préservé l'historique du premier développement qui a pu migrer sans difficulté dans le nouveau système. Toutefois, dans sa première réalisation, son utilisation reste encore un peu trop difficile pour un utilisateur non informaticien. Il requière une installation délicate pour certains systèmes d'exploitation afin de pouvoir s'intégrer correctement avec le client subversion.

**Fig.4 : Génération de planning par feuille XSL vers SVG**



- Phase 3 : amélioration de l'interface utilisateur et généralisation de l'utilisation.

Le comportement satisfaisant de l'outil a motivé la réalisation de quelques améliorations visant à faciliter sa mise en oeuvre et son utilisation par un public plus large que des développeurs logiciel.

Le système est en cours de déploiement et de généralisation à tous les projets Detexia, y compris la gestion interne de l'entreprise qui est traitée de la même façon qu'une affaire.

## Table des URL

[url1] *EGPI* : <http://dexter.detexia.net/egpi/>

[url2] *méthode de développement agile* : [http://fr.wikipedia.org/wiki/M%C3%A9thode\\_agile](http://fr.wikipedia.org/wiki/M%C3%A9thode_agile)

[url3] *subversion* : <http://subversion.tigris.org/>