
	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

Évolution du modèle de projet


Livrable du au titre du projet	COCLICO
Lot	3 : Adaptation des outils et méthodologies
Tache	2 : Évolution du moteur de modèles
Livrable	L3.2.3: Un rapport d'étude sur la possibilité de suivre l'évolution du modèle de projet au cours du temps

Rédacteur(s)	Vérificateur(s)	Approbateur(s)
Giulio IANNAZZO	Codendi Team	

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11


Documents applicables

Documents de références (pour information)

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11


Gestion des versions

N° de version	Date	Auteurs	Modification apportées
1.0	24/02/11	Giulio Iannazzo	Version initiale
1.1	10/02/11	Giulio Iannazzo	Intégration feedbacks équipe

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

Sommaire


1	Objectif.....	5
2	Le modèle de projet (état actuel).....	6
3	Évolution.....	7
	De projet modèle à projet parent.....	7
	Éléments mis à jour par défaut.....	7
	Éléments mis à jour sur option.....	8
	Éléments qui ne peuvent jamais être mis à jour.....	8
4	Idées pour l'implémentation	9
	Plugin projet modèle.....	9
	Référence et copie.....	9
5	Évolutions ultérieures.....	10
	Outils de suivi.....	10

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

1 Objectif

L'objectif de ce document est de décrire brièvement le mécanisme de modèle de projet existant sur Codendi, et de se concentrer sur une analyse de comment ce système pourrait évoluer.

On est en particulier intéressés à mettre en évidence comment les changements sur le modèle pourraient être répercutés sur les projets générés à partir de celui-ci.

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

2 Le modèle de projet (état actuel)

Actuellement les définitions du projet qui sont hérités du modèle, sont :


- Les services activés
- La configuration des outils de suivi (le schéma et les droits)
- Les groupes utilisateurs

De plus une partie des données sont copiées :

- l'arborescence et le contenu des documents
- les paquetages dans le service fichiers
- les pages wiki
- la structure des forums
- les références

Ne sont pas copiés :

- les outils de suivi dont la copie pour les nouveaux projets n'est pas activée
- les données des outils de suivi (artefacts)
- les messages dans les forums
- les liste de diffusions
- les sondages
- les news
- Hudson jobs
- Git
- Subversion

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

3 Évolution

De projet modèle à projet parent

Le fait de pouvoir répercuter les modifications du modèle implique d'étendre le concept même de modèle. Il ne serait plus un simple outil pour faciliter la création des nouveaux projets, mais il deviendrait un vrai projet parent, où on pourrait factoriser toute la configuration commune à plusieurs projets de la même famille. Dans ce sens les projets dérivés ne seraient que spécialisations du projet parent.

La situation idéale consisterait à fournir, pour chaque définition du projet provenant du modèle, la possibilité de choisir s'il s'agit pour le projet fils d'une simple occurrence du projet parent ou d'une spécialisation ; seulement dans le premier cas une mise à jour dans le projet parent serait répercutée dans le projet fils.

On aurait ainsi la possibilité d'exploiter au maximum le mécanisme d'héritage.

Dans le cas où un élément est défini en tant qu'héritage il serait configuré dans le projet fils en lecture seule, pour éviter tout conflit.

Dans le cas d'éléments où l'héritage ne concerne que la structure et la configuration, les données ne seraient pas en lecture seule.


Par exemple, pour un outil de suivi sont hérités la structure des champs et les droits qui les concernent. Ces éléments, si l'héritage est activé, seraient en lecture seule, alors que les données ne le seraient pas.

Par contre, dans le cas des documents ou des pages wiki où le contenu même peut être hérité, le blocage en lecture seule pourrait concerner les données aussi. L'utilisateur aurait ainsi la possibilité de garder la relation, et donc d'hériter toutes les modifications faites dans le document parent, ou bien de rompre la relation et de faire évoluer indépendamment le document fils.

Bien évidemment on peut imaginer de pousser le niveau de configurabilité très loin, mais, pour éviter que la finesse dans le choix des options ne se fasse pas au détriment de l'utilisabilité, il faudra prévoir un comportement par défaut qui soit le plus près possible des habitudes de la plupart des utilisateurs.

Éléments mis à jour par défaut

Il s'agit ici d'identifier les éléments de configuration, ou les données et/ou métadonnées pour lesquels l'héritage des modifications (donc la mise à jour quand l'élément parent est mis à jour) représenterait un comportement souhaitable par défaut.

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

Ces éléments seraient :


- groupes utilisateurs
- services activés

Éléments mis à jour sur option

- la configuration des outil de suivi
- l'arborescence et le contenu des documents
- les pages wiki
- les paquetages dans le service fichiers

Éléments qui ne peuvent jamais être mis à jour

- Dépôts scm (Subversion, CVS, Git)

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

4 Idées pour l'implémentation

Plugin projet modèle

Une implémentation du mécanisme de projet modèle décrit, en accord avec les principes architecturaux recommandés pour les évolution de Codendi pourrait être de regrouper toutes les API concernées dans un plugin. Le choix de cette architecture permettrait d'implémenter cette fonctionnalité dans une autre forge.

En installant le plugin projet modèle on ajouterait à la plateforme les interfaces pour générer un projet à partir d'un modèle et un modèle par défaut (fonctionnalité qui est actuellement fournie dans le core), plus la possibilité de configurer dans les projet fils les éléments qui seraient hérités du modèle (et donc amenés à évoluer de manière synchrone).


Le plugin ajouterait des hooks pour déclencher des événements à chaque modification des éléments d'un projet (seulement les éléments qui peuvent être hérités pour être précis).

Chaque projet fils s'inscrirait à ces événements, et à chaque modification du père répercuterait les modifications dans les éléments configurés.

Référence et copie

Les éléments provenant du modèle sont actuellement copiés dans un nouveau projet, lorsque dans l'implémentation proposé ils seraient insérés dans le projet fils en tant que copies ou références, en fonction de l'activation de la répercussion des modifications.

Ce choix est particulièrement sensible dans le cas d'éléments qui peuvent évoluer souvent, et dont le traitement peut être complexe et lourd, par exemple les documents ou les pages wiki.

	Titre du document : Évolution du modèle de projet
	Référence : L3.2.3
	Version du 10/02/11

5 Évolutions ultérieures

Outils de suivi

Pour aller jusqu'au bout de la logique de factorisation des éléments communs à plusieurs projets dans un projet parent, on pourrait imaginer de rendre les outils de suivi entièrement héritables.

Dans cette hypothèse aussi les données (les artefacts) et toutes les modifications les concernant seraient répercutées sur les projets fils, et pas seulement le schéma et les droits.

On pourrait imaginer par exemple qu'on veuille créer et maintenir centralement certains outils de suivi communs à une famille de projets appartenant au même portfolio.