Rapport sur le travail de Fouad Allaoui

[1 octobre 2006 au 25 mai 2007]

Dans le cadre de l'ODL SmartTools

Intégration de SmartTools dans Eclipse

Par Didier Parigot
Juin 2007

1. Introduction

1.1 Objectif de la mission :

Fournir, à une large communauté de développeurs de plugins Eclipse, nos outils (également sous forme de plugins) pour automatiser le plus possible leur processus de développement. En nous appuyant sur la dynamique de cette plate-forme, permettre une diffusion plus large de notre approche de fabrique logicielle, par une instanciation sur le modèle d'environnement de programmation proposé par Eclipse.

1.2 Objectif de la première année :

Mettre en œuvre une intégration simple de l'approche de génération de plugins dans l'environnement Eclipse, en essayant de traiter la chaîne complète de ce nouveau processus de développement. C'est-à-dire :

- Comprendre comment effectuer une distribution (chargement d'un ensemble de plugins avec les outils de mise à jour d'Eclipse), avec un manuel d'utilisation, des exemples de base, afin de standardiser le plus possible l'intégration dans Eclipse de notre approche;
- Proposer des environnements de programmation sous Eclipse pour que l'utilisateur puisse écrire les spécifications associées à un plugin (nos divers petits langages de description). Puis associer à ces environnements, un ensemble de menus pour accéder aux fonctionnalités de génération de notre approche.
- Intégrer cette démarche dans l'approche classique de programmation de plugins (Plugin Développement Environnement, PDE) en spécialisant les « Wizards » et en produisant des plugins standards.
- Utiliser les outils de déploiement de plugins pour valider les plugins générés par cette fabrique de plugins.

Ce processus de développement est résumé dans les trois figures ci-dessous :

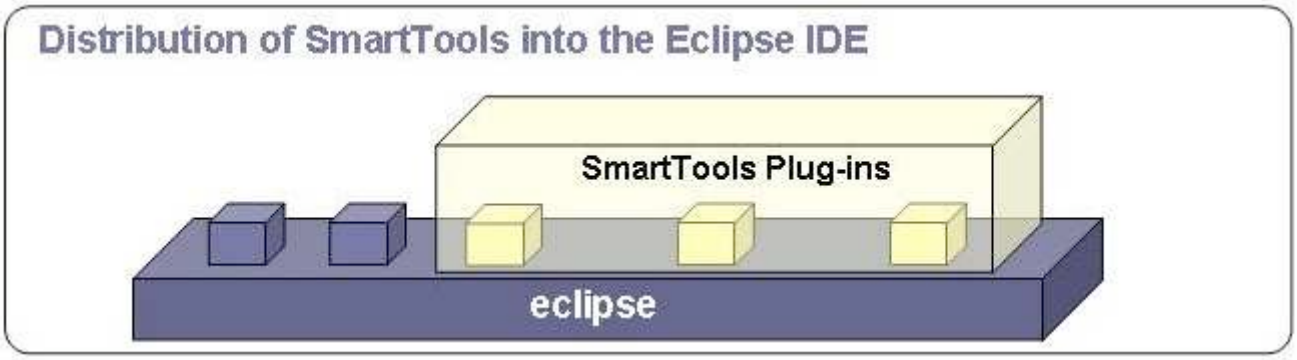

Figure 1 : Distribution de SmartTools.

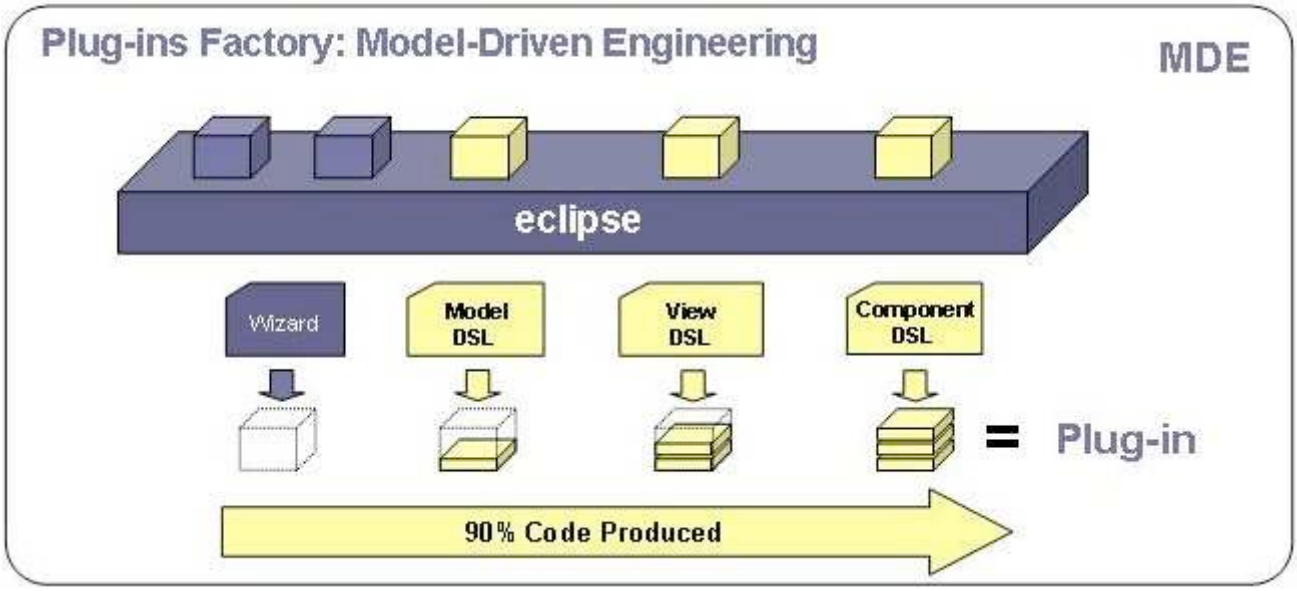

Figure 2 : Environnements et processus de génération de nos petits langages.

Figure 3 : Déploiement des plugins produit par notre fabrique.

1.3 Une synthèse du travail effectué

Nous allons donner d'abord une synthèse du travail effectué sous la forme d'une séquence d'étapes principales. Par la suite, voir la Section 4, nous donnerons plus de détails techniques pour chaque étape. Les étapes principales du travail de Fouad Allaoui ont été les suivantes :

- Etape de mise en place du développement de SmartTools sous Eclipse. Cette étape a constituée à effectuer un ensemble de transformations sur SmartTools, pour préparer les étapes suivantes et permettre un développement sous l'environnement

Eclipse (vu comme un simple IDE pour Java) de l'ensemble des composants de SmartTools.

- Etapes d'intégration de nos trois composants Absynt, Cosynt et CDML en élaborant leur environnement minimal. Cette étape a constituée à prendre comme exemple ces trois plugins pour initier (manuellement) cette intégration et comprendre comment les plugins génères par SmartTools pouvaient s'intégrer à la plate-forme Eclipse.
- Etapes d'intégration de nos outils de génération automatique de plugins. Cette étape a constituée à généraliser l'étape précédente et permettre d'effectuer les phases de génération au siens même d'Eclipse (sans passer par un processus externe).
- Etapes de distribution de notre approche (ensemble de plugins à charger). Cette étape a consistée à comprendre comment utiliser les outils de distribution d'Eclipse, et mettre en place une distribution standard à Eclipse de SmartTools .
- Etapes de démonstration sur un exemple très basique. Cette étape a consistée à élaborer une démonstration complète avec les diverses étapes de ce nouveau processus de développement de plugins.
- Etape de diffusion de notre approche avec l'élaboration de posters, de documents, et des présentations au Salon Linux et EclipseCon'07. Cette étape a consistée à concevoir un ensemble de document (posters etc…) pour présenter et expliquer cette approche de génération automatique de plugins.
- Etape l'intégration de notre architecturé orientée service (SOA) pour l'assemblage et la communication entre plugins génères par SmartTools.

2 Conclusion

2.2 Remarques par rapport au planning prévu

Le travail de Fouad Allaoui correspond quasiment aux tâches de la première année (voir en annexe le rappel du planning de la proposition) avec les trois remarques suivantes :

- Nous avons pu très rapidement montrer (au Salon Linux'07, février et à EclipseCon'07, Mars) une trame du produit que nous souhaitons obtenir au final. Bien sur, il nous été impossible d'élaborer, aussi rapidement, un (ou des) exemple périment pour cette démonstration.
- Les tâches 3 et 5 spécifiques à Eclipse (utilisation plus poussé des outils d'éditions pour obtenir de vrai éditeur incrémentaux) sont moins avancées que prévus et devront être poursuit et approfondis dans les mois qui viennent.
- L'intégration de notre architecturé orientée service (SOA) a été avancée par rapport au planning (la tâche 2 de la deuxième année). Cette intégration de notre SOA permet d'assurer un fonctionnement sans un développement spécifique (code source à l'Eclipse) et donc une meilleure indépendance entre Eclipse et SmartTools.

Une grande partie du travail de Fouad a été de comprendre, avec une certaine difficulté, le fonctionnement d'Eclipse et d'effectuer de nombreux tests pour valider notre démarche et les solutions mise en œuvre. Cela explique un peu pourquoi les tâches (3 et 5) s'appuyant essentiellement sur la technologie Eclipse soient moins avancées. Puis surtout elles sont moins prioritaires pour valoriser notre démarche (obtenir des environnements de qualité pour nos DSLs).

3. Planning des tâches pour la deuxième année:

Tâche 1 : Finalisation de la version 1.2 avec l'intégration de notre SOA

- Suppression des objets graphiques et simplification du composant GUI de SmartTools (`Glayout`) pour interfacer notre gestionnaire de composants (CM) avec Eclipse.
- Finalisation de la création des vues graphiques (problème de fermeture et d'activation entre deux lancements d'Eclipse). Comme Eclipse mémorise l'état de la dernière session, cela nous impose de fermer toutes les vues graphiques SmartTools lorsqu'une session est terminée.
- Environnement Eclipse pour le langage de déploiement (langage World) et le langage de construction (assemblage) de vue graphique (Lml) nécessaire pour utiliser notre SOA**.**
- Finitions sur l'ensemble de nos environnements avec les vues graphiques associées (les outils d'éditions).
- Distribution (nouvel ensemble de plugins) avec notre gestionnaire de plugins (CM) et les deux DSLs , LML et World.
- Version sous Windows.
- Tests de cette nouvelle version (phase de distribution, de génération et de déploiement)
- Elaboration d'exemples plus complexes (e.g. connexion à une base de données avec l'éditeur des réponses) en dehors des exemples standards de SmartTools.

Objectif : mettre en place un ensemble de démonstrations pour montrer l'ensemble des possibilités de notre approche avec notre SOA (assemblage de plugins).

Tâche 2 : Travail de diffusion de cette version 1.2

- Soumettre des présentations (ou démonstration) à EclipseCon Europe (Octobre 07) et EclipseCon '08 ou eXchange (workshop sur les technologies Eclipse à OOSPLA'07), et au Salon linux'08.
- Diffusion à diverses communautés (OSGi par exemple**),** avec l'élaboration d'exemples de démonstrations et de tutoriaux ciblés.
- Mettre en place une démonstration exécutable sur note site web.
- Documentation, Site Web, et Liste de diffusion.
- Mettre notre logiciel sous un Plugins Center.

Objectif : Publicité et diffusion cette version 1.2

Tâche 3 : Exploitation de nos outils sémantiques à base de visiteurs

- Environnement pour notre langage Xprofile (profile des visites sémantiques à base de cette technique de visiteur), et d'autres langages de transformation.
- Mettre à disposition ces techniques de description sémantique.
  - Documentation.
  - Menu, « Wizard » d'aide à l'utilisation.
  - Elaboration d'exemples de démonstration (hors du cadre standard de SmartTools).
- Effectuer des tests sur ces techniques de programmation par aspects (sémantique extensibles).

- Distribution des nouveaux plugins.

Objectif : Préparer le terrain pour nos projets en soumission (ANR RNTL et appel FP7) sur BPEL (langage de workflow sur les Web Services) qui devront utiliser ces techniques de description sémantique (programmation par aspects au dessus de la programmation par visiteurs).

Tâche 4 : Exploitation de notre version distribuée avec notre SOA

- Approfondir l'utilisation de la bibliothèque R-OSGi (Remote OSGi, composants OSGi distribués) sur nos RCPs.
- Mettre en œuvre des applications utilisant cette version distribuée de notre SOA.

Objectif : Valoriser nos techniques d'assemblage de composants et de visualisation graphique qui sont bien adaptées à ce contexte distribué.

4. Chronologie du travail effectué, mois par mois, par Fouad Allaoui pour atteindre nos objectifs

Octobre 2006 :

- Prise en main de l'outil SmartTools, avec les concepts et son développement existant (100 000 lignes de code java) et le processus de génération (externe à Eclipse)
- Prise en main sous gforge du projet de développement SmartTools
  - Mise sous gforge de l'ensemble des sources de SmartTools.
  - Ajout des Copyright, des variables de SVN etc…
  - Utilisation des outils de gforge (tâche, etc…)
- Passage à java 1.5 pour l'intégration des objets swing dans Eclipse (passage obligatoire) pour utiliser nos vues graphiques, et mise à jour de nos bibliothèques externe pour être conforme à java 1.5 et Eclipse.
- Passage à la norme de composant OSGi (mise en conformité de l'ensemble des fichiers Manisfet des composants SmartTools, fichier de description de fichier JAR)
- Passage sous Eclipse (sous forme d'un projet Eclipse) de nos composants, et nombreux tests « bash » de génération (reconstruction complète de SmartTools).

Résultat : Mise en place du développement de SmartTools avec nos notions de composant (plugins) et de génération dans Eclipse.

Novembre 2006 :

- Intégration de nos composants Absynt, Cosynt, CDML et construction d'environnements minimaux pour ces trois petits langages :
  - Test sur la génération de plugins manuellement pour ces trois plugins
    - Test sur la construction des arbres abstraits ;
    - Test sur l'appel des parseurs et construction des vues SmartTools ;
- Mise en place des diverses vues graphiques et des actions (pour l'appel de génération exécute)

    - Editeur générique et Vue générique pour les plugins générés pour partager le maximum de code Java et facilite l'utilisation de notre approche ;
    - Actions et menus de base pour nos DSLs, comprendre le mécanisme de vue d'Eclipse avec les notions de point d'entrée (extension d'un plugin)
    - Vue graphique « Out-line » automatique : la vue graphique des arbres abstraits lue par nos parseurs générées automatiquement ;
- Mise en place de la phase de génération de plugins au sein d'Eclipse.
    - Comprendre le chargement des classes Java avec l'approche OSGi et Eclipse, chaque plugin a son propre chargeur de classes (ClassLoader) .
    - Comprendre l'environnement PDE (environnement de développement de plugins Eclipse) pour la gestion des dépendances vis-à-vis d'autre plugins (dépendances décrites dans un fichier particulier d'Eclipse pour chaque plugins « plugins.xml »).
- Mise en place de l'activeur générique dans le contexte OSGi et d'Eclipse (phase de lancement des plugins).
- Construction des plugins de base (bibliothèque), st-jar pour les bibliothèques externes et st-core pour les bibliothèques internes à SmartTools.

Résultat : Première phase d'intégration de nos composant Absynt, Cosynt et CDML

Décembre 2006 :

- Réorganisation des répertoires (projets, plugins) pour prendre en compte la mise en place d'une distribution avec sa documentation :
    - Création de plugins spécifiques à l'intégration (éditeur générique, manuel, exemples, site de distribution locale etc…)
    - Déplacement des fichiers de ressources (contrainte de nommage des bundles OSGi) et renommage de certains plugins.
- Mise en place d'une démonstration pour le Salon Linux
    - Nombreux tests sur les phases de génération, édition, de génération des vues graphiques, pour avoir une démonstration complète de la démarche SmartTools (Distribution, Spécification, Génération, Déploiement, Utilisation)
- Début d'une documentation, en utilisant des outils spécifiques d'Eclispe (le langage toc d'Eclipse), comme un manuel en ligne ou des « Cheat Sheets » pour aider l'utilisateur dans ces premiers pas.
- Distribution sous forme d'une « Feature » (ensemble de plugins à charger)
    - Comprendre le fonctionnement de cet outil « Feature ».
    - Nombreux tests pour vérifier le bon fonctionnement.
    - Tentative de réduction du plugin `st-jar` (les bibliothèques externe de SmartTools) car Eclipse contient déjà une partie des même bibliothèques.
- Elaboration d'un « Wizard » d'utilisation de l'approche de génération automatique de plugins.

Résultat : Première mise en place de notre approche de génération automatique de plugins

Janvier 2007 :

- Poursuite de la Documentation et de la Distribution (« Features »)
    - Création d'un site Web avec une page pour le chargement, une courte présentation de « Plugins Factory » et une page sur la démonstration.
    - Perfectionnement sur les outils de documentation d'Eclipse.

- Elaboration de plusieurs posters, d'une présentation et d'un « flyers » sur cette première intégration, pour le salon linux. Les figures 4, 5 et 6 ci-dessous donnent un aperçu de ce travail.
- Nombreux tests sur la démonstration pour le Salon Linux pour s'assurer de son bon fonctionnement ;
- Elaboration d'un jeu icônes pour la fabrique SmartTools car dans Eclipse, beaucoup d'objets (un type de fichier, un menu, une vue graphique etc.) demande la présence d'une icône;
- 3 jours au salon linux.
  - Mise en place d'une première liste d'industriels intéressés par l'approche

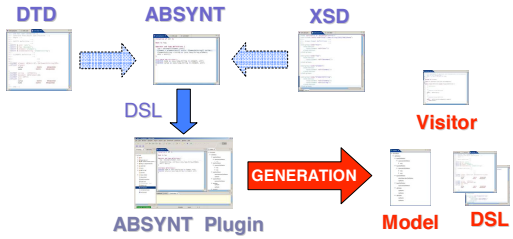

Figure 4 : Schéma de génération pour Absynt (grammaire)

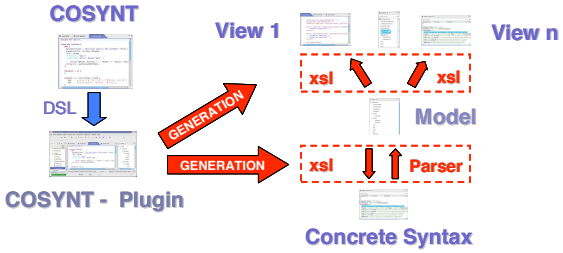

Figure 5 : Schéma de génération pour Cosynt (parseur et vue graphique)

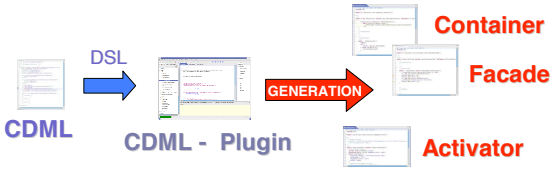

Figure 6 : Schéma de génération pour CDML (Conteneur et Activateur)

Résultat : Première démonstration de notre intégration

Février 2007

- Préparation de la Conférence EclipseCon 2007
- Finalisation des posters pour EclipseCon'07, et le flyer en englais
- Quelques essais avec le frame-work EMF (environnement UML d'Eclipse), pour éventuellement mettre en place notre traducteur UML vers Absynt ;
- Mise en place d'un site (présentation et site de chargement de la « Feature »)
  - Mieux comprendre le fonctionnement du mécanisme de mise à jour de cet outil
  - Test sur cette distribution en mode « update »
- Construction d'un « Wizard » avec effet de bords pour la mise à jour d'un plugin déjà construit et extension pour accepter des langages spécifiés en UML.
- Mise en conformité des descriptions de plugins (en particulier le fichier `plugins.properties`) pour le déploiement avec l'export d'Eclipse.

- Création d'un Plugin contenant des exemples (pour la ou les démonstrations à venir)

Résultat : Première étape de diffusion de notre approche

Mars 2007

- Semaine à EclipseCom '07
- Début de la Mise en place de notre Architecture Orientée Service (SOA)
  - Comprendre le fonctionnement de nos petits exemples de Rich Client Platform (RCPs) qui utilisent cette SOA (RCP-Basic et RCP-Base-Donnée). La figure 7 montre un exemple d'assemblage de plugins dans le cadre particulier des RCPs d'Eclipse.
  - Suppression des appels directs au code métier de nos composants, par utilisation direct de nos messages en passant par les conteneurs de nos composants.
  - Lancement de notre gestionnaire de composant (CM) pour la création et l'interconnexion entre nos composants dans le fichier de configuration d'Eclipse (les composants actifs au lancement).
  - Associer à chaque composant SmartTools un objet Eclipse (en général une vue Eclipse) pour interagir au sein d'Eclipse avec le composant.
  - Création de l'objet Eclipse associé à notre composant GUI (Glayout) pour réaliser l'interface entre notre gestionnaire de composant (CM) et Eclipse.
- Mise en place de l'ouverture d'un document (éditeur) avec notre SOA, en passant par la création du composant SmartTools (avec son conteneur associé), et à l'aide des messages de notre SOA (Open File).

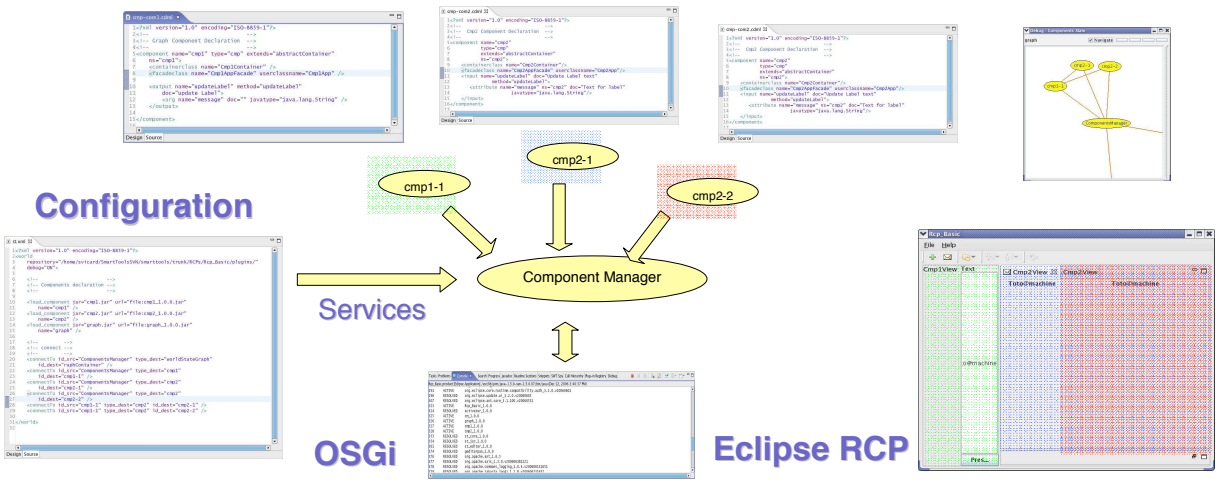

Figure 7 : Exemple de notre SOA au-dessus d'une RCP d'Eclipse

Résultat : Finalisation de la première version et début de l'intégration de notre SOA
Fouad a pris 15 jours pour un séjour linguistique au USA.

Avril 2007

- Finalisation de la mise en place de notre Architecture Orientée Service
  - Mise en place de l'ouverture d'une vue graphique (message `Request View` pour SmartTools) en passant par notre SOA.

    - Lancement de notre gestionnaire de composants à la demande (pour éviter les problèmes de synchronisation au démarrage des plugins actifs) à l'aide d'un menu spécifique.
- Intégration de notre composant de « Log » pour la trace des envois de message entre nos divers composants pour nous aider dans cette mise en place de notre SOA;
- Mise en place de la synchronisation des vues Eclipse pour montrer dans Eclipse les avantages de notre SOA (connexion entre les plugins automatiquement générés à l'aide de notre langage de composant, CDML).
    - Comprendre le fonctionnement de base d'Eclipse pour la synchronisation entre un éditeur et les vues Eclipse (graphiques) associées.
- Mise en conformité de nos exemples de « Rich Client PlatForm » avec les nouveaux activateurs de plugins (associant un composant SmartTools à un objet Eclipse).

Résultat : première phase d'intégration de notre SOA

Mai 2007

- Finalisation de la synchronisation des vues Eclipse
    - Comprendre la notion de multi-vue d'Eclipse (comme la vue Out-Line) qui permet de montrer dans une même vue Eclipse, différente vue (de même type) avec un mécanisme de sélection automatique (mise au premier plan) en fonction de l'éditeur sélectionné.
    - Mise en place d'une notion de multi-vue pour un type de vue graphique de SmartTools equivalent a l'Out-Line d'Eclipse.

Résultat : première phase de synchronisation des vues SmartTools avec l'éditeur Eclipse associé.

5. Annexe : le planning de la proposition

Une proposition d'un planning des tâches avec [durée, phase]:
Pour la première année :

1. Apprentissage de l'approche SmartTools; [1 mois, phase 0]
2. Intégration basique de nos composants pour rendre l'ensemble de nos services disponibles à travers l'environnement Eclipse. S'assurer que les services de génération de notre composant « absynt » s'intégrer bien à la plate-forme Eclipse avec les parties de traitements sémantiques « xprofile »; [1 mois, phase 1]
3. Approfondir l'utilisation des services de notre composant « cosynt » (génération de parseur/scanner) pour d'obtenir de vrai éditeur incrémentale « à la Eclipse »; [1 mois, phase 1]
4. test sur nos trois composants [15 jours, phase 2]
5. Intégration de notre démarche de construction automatique de plugins en utilisant le plus possible les divers services (éditions, message d'erreur etc..) de la plate-forme Eclipse; [1 mois, phase 3]
6. phase de distributions des trois plugins de base [15 jours, phase 2 et 3]
7. Intégration des services de notre composant « cosynt » en terme de construction automatique de vue graphique; [1 mois, phase 1]
8. Documentations et instrumentations de nos trois plugins et de notre démarche de fabrique logicielle; [3 mois, phase 2 et 3]

9. Effort de diffusion de notre approche à travers des présentations/démonstrations. [3 mois, phase 3]

Pour la deuxième année : en poursuivant les tâches 8 et 9

1. Approfondir nos plugins en terme d'extension de services pour permettre d'étendre nos composants graphiques; [1 mois, phase 4]
2. Utilisation de l'architecture par composants dirigée par les services pour une plus grande souplesse; [3 mois]
3. Utilisation de la génération automatique de vue graphique pour une exportation de composant (plugins) à travers le réseau; [2 mois]
4. Utilisation de notre approche de programmation par aspects pour une adaptation dynamique des composants. [2 mois]

Les tâches 8 et 9 et celles de la deuxième année devront être précisées et découpées en sous tâches.