# Descrizione dello sviluppo del plugin

## 1 Introduzione

Questo documento non è necessario per utilizzare il plugin TTEP e relativo help, nè è parte integrante della documentazione di sviluppo. Invece ha come obiettivo la descrizione delle fasi principali dello sviluppo dei plugin, i tool utilizzati e l'ambiente di lavoro in maniera sintetica e informale.

## 2 Hardware/software

Plugin realizzato su:

- pc con 768 MB di ram, CPU centrino 1,5 Ghz
- Linux (con test di compatibilità anche su Windows Xp)

Sistema di versionamento del progetto:

- automatico
- utilizzando svn con repository remoto fornito da Assembla.com, indirizzo del progetto
  http://www.assembla.com/wiki/show/b5O0SY0smr3BFzab7jnrAJ , indirizzo del repository svn
  http://subversion.assembla.com/svn/ttep (è quindi possibile vedere come si è evouto lo sviluppo del codice essendo conservate tutte le versioni)
- il progetto è diventato pubblico in corso d'opera per cambiamento delle politiche di fornitura del servizio del sito

- nel progetto pubblico NON è (e non è mai stato) compreso codice di TwoTowers o comunque codice non inerente il plugin stesso

Versioni di Eclipse utilizzate:

- inizio del lavoro su Eclipse Europa
- passaggio in corso d'opera ad Eclipse Ganymede
- la versione più recente, Galileo, non è stata utilizzata nello sviluppo
- update costanti e tuning del file di configurazione (altrimenti frequenti crash di Eclipse)

Tool di supporto usati:

- Plugin di Eclipse per:
  - modelli uml (Omondo -> eUml2 -> Umlet)
  - latex (Texlipse)
  - svn (Subversive)

# 3 Sviluppo del progetto

Fonti di informazioni maggiormente utilizzate:

- documentazione ufficiale di Eclipse
- paper, tutorial, documenti presenti nel sito ufficiale di Eclipse
- mailing list specifiche per lo sviluppo di plugin Eclipse
- Google code (code.google.com)
- vari siti internet che trattano lo sviluppo di plugin Eclipse.

Lunga fase di studio di Eclipse, sia iniziale, sia durante lo sviluppo. Da utente:

- comprensione della terminologia (es. prospettiva, vista, workspace, workbench...)

- logiche di funzionamento del programma
- padronanza della gestione del tool

Da sviluppatore:

- studio dell'architettura generale
- comprensione della logica dei plugin
- individuazione delle componenti dell'architettura su cui intervenire per raggiungere l'obiettivo
- scelta (non sempre a priori) tra diverse metodologie di implementazione di funzionalità (es. menù actionSet vs command)
- controllo dell'evoluzione della piattaforma Eclipse al fine di non produrre un plugin obsoleto ancor prima del rilascio (es. evitato l'utilizzo di classi/metodi o pattern deprecati)

Studio del tool TwoTowers:

- funzionamento generale
- architettura generale
- logica di funzionamento del codice sorgente
- individuazione dei punti di collegamento su cui agganciare il plugin