ARPA FVG
Agenzia Regionale per la Protezione dell'Ambiente del Friuli Venezia Giulia

Art. Tec-Sci No. 007/2015 (29/10/2015)
ARPA FVG - CRMA

CRMA ARPA FVG

Il sistema di sviluppo collaborativo di modelli e software del CRMA

Dario Giaiotti[1]

Sommario
In questo articolo si illustra il gruppo di applicativi che costituisce il sistema di sviluppo collaborativo dei software e dei modelli del CRMA. Si descrivono i principi fondamentali secondo i quali vengono affrontati i problemi ambientali che sono trattati con strumenti computazionali, inoltre si dettaglia la procedura che permette l'integrazione degli strumenti Git, TRAC e Jenkins in un unico ambiente di sviluppo. [1, 2, 3] .

Keywords
Sviluppo modellistico collaborativo - Git - TRAC - Jenkins

[1] *ARPA FVG - CRMA*
*__Autore di riferimento__: Dario.Giaiotti@arpa.fvg.it

Indice

1. Introduzione

Nello sviluppo e per la manutenzione di modelli, il CRMA si avvale di diverse competenze che sono distribuite tra i colleghi che compongono il Centro. Oltre alle competenze specifiche di ciascuno, vi sono le attitudini e le metodologie personali che sovente si confrontano nell'intento di trovare soluzioni.

Con lo scopo di rendere quanto più sinergico, ma anche documentabile, quindi revisionabile, e riproducibile, il processo che porta alla realizzazione di un sistema modellistico, di un'elaborazione dati o di loro parti, in generale con l'obiettivo di trovare la soluzione computazionale ad un problema, il CRMA adotta un processo collaborativo che si avvale di un ambiente concettuale e computazionale determinato da alcuni elementi caratteristici, presentati in questo articolo.

2. Il processo di sviluppo ed applicazione dei codici numerici

La realizzazione e l'utilizzo di codici numerici e, nei casi più articolati, di modelli avviene secondo un processo che è sintetizzato nei seguenti punti essenziali:

- l'analisi del problema e la formulazione di uno o più quesiti a cui dare risposta;
- la progettazione del sistema computazionale da impiegare per rispondere ai quesiti del problema;
- la ripartizione dei compiti e l'attribuzione delle attività computazionali personali;
- il coordinamento e la verifica delle attività personali;
- l'implementazione operativa o stabile del sistema computazionale;
- l'applicazione del sistema computazionale e la stesura della risposta al problema.

Questi passi vengono praticamente realizzati con delle riunioni in cui il colleghi si confrontano e forniscono il proprio contributo all'analisi del problema, alla formulazione dei quesiti, alla progettazione del sistema computazionale e alla discussione dei risultati ottenuti o alla stesura della risposta. Le attività di assegnazione dei compiti, di verifica del loro svolgimento nei tempi attesi, di sviluppo e di versionamento del codice, fino all'implementazione e all'applicazione del sistema computazionale, si avvalgono di opportuni software che sono stati integrati in un unico strumento che viene indicato per brevità come il **"Sistema di sviluppo collaborativo di modelli e software del CRMA"**.

3. Il sistema di sviluppo collaborativo di modelli e software

Lo sviluppo collaborativo di codici e modelli è basato su tre software fondamentali ed una procedura di documentazione. I tre software sono:

- **Git** (http://git-scm.com/), utilizzato per la tracciabilità ed il controllo di versione del codice sviluppato.
- **Jenkins** (http://jenkins-ci.org/), utilizzato per la verifica del funzionamento delle parti di codice realizzate attraverso la tecnica di *Continuous Integration* (CI)
- **TRAC** (http://trac.edgewall.org/), utilizzato per l'attribuzione dei compiti e il monitoraggio dello stato di avanzamento dello sviluppo dei codici, dei modelli e della documentazione.

I tre software sono integrati, ovvero quando un codice viene versionato stabilmente, tramite Git, vengono attivati dei test che ne eseguono il controllo, tramite Jenkins, e la documentazione del versionamento viene messa a disposizione e collegata ai compiti assegnati, tramite TRAC.

La documentazione dettagliata dei codici sviluppati, avviene sia tramite testi che immagini e schemi. Anche se non esiste una prescrizione su quali strumenti concettuali utilizzare per progettare il codice, il modello o parte di essi, viene caldamente suggerita l'applicazione dei concetti di base di UML (http://www.uml.org/), in particolare dei diagrammi del tipo statico: class e object diagrams; dei diagrammi di tipo dinamico: activity, sequence e use case diagrams. La documentazione dettagliata viene trasformata in opportune pagine Wiki, che sono rese disponibili tramite l'implementazione wikiCRMA (http://ms05lxarpa.arpa.fvg.it/wiki) ed in alcuni casi in articoli tecnico scientifici o rapporti interni al CRMA.

Nei casi in cui lo sviluppo collaborativo richieda la stesura di un progetto articolato, che comporta il conseguimento di parecchi obiettivi, viene utilizzato la tecnica del diagramma di Gantt per stendere e documentare gli obiettivi fondamentali, i quali saranno conseguiti attribuendo specifici compiti ai colleghi. Il diagramma Gantt si integra nel sistema di sviluppo tramite la corrispondenza biunivoca che viene attribuita tra gli obiettivi del diagramma e gli obiettivi del progetto TRAC associato al diagramma. Infatti ogni giorno, viene eseguita automaticamente una verifica su tutte le attività gestite tramite TRAC, la quale produce dei file in formato csv il cui contenuto può essere collegato all'analogo file csv derivante dal diagramma di Gantt, realizzato tramite software GanttProject (http://www.ganttproject.biz/).

L'implementazione operativa o stabile del sistema computazionale è basata sul concetto di workflow. Due sono i workflow manager adottati dal CRMA. Il principale è l'ecFlow (https://software.ecmwf.int/wiki/-display/ECFLOW-/Home), utilizzato nell'ambiente di calcolo HPC, mentre per flussi di calcolo coinvolgenti singoli PC, specialmente in ambiente Windows si fa uso del Kepler (https://kepler-project.org/)

Tutti questi strumenti sono disponibili sul cluster di calcolo FENICE.

4. L'utilizzo di Git per la creazione di reporitories e il collegamento con TRAC

Il fulcro informatico del sistema di sviluppo collaborativo di modelli software è Git.

I repository Git sono collegati a progetti TRAC e Jenkins in modo tale che le operazioni di salvataggio di file nei repository remoti di Git, ovvero quelle che tecnicamente sono chiamate le push, scatenano delle azioni nei progetti TRAC e Jenkins.

Quindi l'operazione preliminare del processo collaborativo è la creazione di un repository Git, seguendo i sottoelencati punti.

- Creazione di un repository Git (subsection 4.1)
- Modifica del file di configurazione di Gitolite (subsection 4.2)
- Associazione di un repository Git a TRAC (subsection 4.3)
- Associazione di un repository Git a Jenkins (subsection 4.4)
- Utilizzo di Git per gestire automaticamente i ticket di TRAC (subsection 4.5)

4.1 Creazione di un repository Git

L'archiviazione permanete di software sulla FENICE necessita della disponibilità di un repository, che se non è già esistente deve essere creato. La procedura per la creazione di un repository ex novo è la seguente.

La directory in cui sono depositati i file riguardanti l'amministrazione è di proprietà dell'utente operative ed è:

/u/arpa/operative/apps/git-admin.

Operazioni preliminari eseguite una volta sola
Per la realizzazione di repository, per gli utenti del gruppo ARPA, si fa uso dello strumento **gitolite** (http://gitolite.com/-gitolite/), seguendo una procedura che è stata definita nell'ambito dei servizi forniti da eXact Lab (http://www.exact-lab.it/).

Solo per il primo uso dello strumento gitolite è stato clonato il repository costruito appositamente per l'amministrazione degli altri repository, ovvero il gitolite-admin.git. Il comando è il seguente:

git clone
ssh://git@grid1.mercuriofvg.it/gitolite-admin.git

[operative@access git-admin]$ git clone ssh://git@grid1.mercuriofvg.it/gitolite-admin.git
Initialized empty Git repository in /u/arpa/operative/apps/git-admin/gitolite-admin/.git/
remote: Counting objects: 123, done.
remote: Compressing objects: 100% (93/93), done.
remote: Total 123 (delta 27), reused 0 (delta 0)
Receiving objects: 100% (123/123), 18.47 KiB, done.
Resolving deltas: 100% (27/27), done.

```
[operative@access git-admin]$ la
total 12
drwxr-xr-x3operative arpa 4096 Sep 15 12:45 .
drwxr-xr-x4operative arpa 4096 Sep 15 12:17 ..
drwxr-xr-x6operative arpa 4096 Sep 15 12:45 gitolite-admin
```

Eseguita l'inizializzazione del repository locale di Git per **git-admin**, come per qualsiasi altro repository, dalla directory /u/arpa/operative/apps/git-admin/gitolite-admin è stato eseguito un allineamento con il remote repository. Le operazioni eseguite sono state queste:

```
git status
git pull
```

Dopo la prima clonazione non è più necessario rieseguire le operazioni sopra descritte in quanto si usa il Git come per qualsiasi altro codice si intenda sviluppare; a tali operazioni è abilitato solo dall'utente operative.

Operazioni da eseguire ogni volta che si intende creare un nuovo repository

La prassi operativa del CRMA per la creazione di un repository Git, per qualsiasi scopo, prevede che il repository sia associato ad un progetto TRAC ed eventualmente ad uno o più progetti Jenkins per la pratica del continuous integration. Più repository possono essere associati allo stesso progetto TRAC, ma non viceversa. Quindi, nel caso più semplice il repository ed il progetto TRAC hanno il medesimo nome, mentre nei casi complessi il nome del progetto potrebbe coincidere con quello di uno solo dei repository, oppure essere diverso da quello di tutti i repository ad esso legati. Inoltre deve essere assegnato un responsabile alla manutenzione e all'aggiornamento delle pagine TRAC associate al progetto. La creazione avviene secondo i compiti qui di seguito esposti che vanno eseguiti in sequenza.

1. Un utente del cluster qualsiasi richiede la creazione di un nuovo repository al collega che ha in gestione l'utente operative. Il nome del repository viene definito da operative. Nel caso un repository esista già e un utente ne chieda l'autorizzazione all'uso, si procede come nel caso della creazione di uno nuovo, ma solo considerando i punti che riguardano gli utenti nel file di configurazione di gitolite e i progetti Jenkins da associare al repository.

2. L'utente operative modifica il file di configurazione gitolite e aggiunge solo il nome del nuovo repository e gli utenti abilitati, infine esegue un Git push per allineare le versioni del file di configurazione gitolite (**gitolite.conf**).

3. L'utente operative popola il nuovo repository depositando un file README nel quale introduce le seguenti informazioni: a cosa si riferisce il repository, l'indirizzo completo del repository remoto, chi ha richiesto la creazione del repository, la data di creazione, chi era la persona fisica che aveva in carico l'utente operative nel momento della creazione del repository. Alla fine esegue un Glit push. Con questa operazione il nuovo repository non sarà più vuoto.

4. L'utente operative crea un nuovo progetto di TRAC che ha lo stesso nome del repository e nella pagina principale riporta le informazioni che sono state inserite nel file README del punto precedente.

5. L'utente operative modifica nuovamente il file di configurazione gitolite e aggiunge le specifiche per il collegamento del repository con lo strumento collaborativo TRAC; inoltre aggiunge solo il token che attiva gli eventuali progetti Jenkins, ma non lo hook.post-receive per Jenkins e neppure fa riferimento ad alcun progetto, in quanto non esistono. Il nome del token Jenkins deve essere quello attribuito al repository con l'aggiunta della stringa "_verify" Infine esegue un push per allineare le versioni del file di configurazione gitolite.

6. L'utente operative modifica per la seconda volta il file README aggiungendo quali sono le variabili che individuano il nome del progetto TRAC, chi è l'utente TRAC responsabile del mantenimento del progetto TRAC associato al repository, e eventuali altre informazioni che si ritengono utili. Indica anche qual è il token Jenkins che gli utenti dovranno utilizzare per attivare i progetti Jenkins associati al repository. Alla fine esegue un Git push.

7. L'utente operative, aggiorna la pagina principale del progetto TRAC associato al repository aggiungendo le informazioni che ha inserito nel file README al punto precedente. Subito dopo comunica, via e-mail e/o tramite ticket del progetto TRAC, a tutti gli utenti abilitati che è stato creato il repository e il nome dell'utente responsabile del mantenimento delle pagine TRAC del progetto.

8. Gli utenti che lo desiderano, quando lo riterranno opportuno, nel proprio account Jenkins, creano il progetto Jenkins caratterizzandolo con le specifiche del repository, del token Jenkins ad esso associato e che troveranno sulla pagina del corrispondente progetto TRAC o sul file README. Il nome del progetto Jenkins non deve contenere spazi e deve essere breve ed auspicabilmente richiamare il nome del repository. Infine comunicano all'utente operative che hanno definito un progetto Jenkins che va abilitato e il nome del progetto.

9. L'utente operative modifica il file di configurazione gitolite e aggiunge le specifiche per il collegamento del repository con lo strumento collaborativo Jenkins aggiungendo lo hook.post-receive per Jenkins e il token relativo al nome del progetto Jenkins comunicatogli dal richiedente. Più di un progetto possono essere definiti purchè separati da uno spazio bianco. Alla fine esegue un Git push.

Un esempio del README file è disponibile nel repository ambiente_FENICE, sotto il nome di README.tpl.

4.2 Modifica del file di configurazione di Gitolite

La definizione del nuovo repository avviene aggiungendo il nome del repository che si intende creare al file di configurazione **gitolite.conf** che si trova nella sotto directory **conf** di **/u/arpa/operative/apps/git-admin**.

Oltre al nome del repository, che è case sensitive, si deve anche associare almeno un utente al repository. Il formato da seguire per queste operazioni è descritto nei dettagli dalla procedura messa a disposizione da eXact Lab e brevemente illustrato da questo esempio nel quale l'utente operative può leggere (R), scrivere (W) e rimuovere (+) il repository:

repo **ambiente_FENICE**
option hook.post-receive = jenkins trac
config jenkins.project = "ambiente_FENICE"
config jenkins.token = "ambiente_verify"
config trac.envdir = "/storage/trac/projects/ambiente_FENICE"
RW+ = operative
RW = giaiottid
R = montanarif
R = jenkins

Al termine della modifica del file di configurazione eseguire l'allineamento del local repository al remote:

```
git add gitolite-admin/conf/gitolite.conf
git commit
git push
```

4.3 Associazione di un repository Git a TRAC

Prima di eseguire il collegamento del nuovo repository creato con Gitolite ad un progetto TRAC qualsiasi, si deve eseguire almeno una operazione di push sul repository appena creato. Secondo la procedura, questa operazione la esegue l'utente operative che aggiunge al repository il file README. Ciò è necessario in quanto Gitolite crea dei repository vuoti e nel caso il progetto TRAC sia associato ad un repository vuoto, si incappa in un messaggio d'errore nella sezione **sorgenti** del progetto. Se questo accade, conviene clonare il repository vuoto appena creato ed eseguire almeno un commit, allora il messaggio d'errore dovrebbe sparire dal progetto TRAC. Per esempio:

```
git clone (ssh://git@grid1.mercuriofvg.it/REPOSITORY_CREATO.git)
cd REPOSITORY_CREATO
vim README
git add README
git commit -m "Initial commit"
git push origin master
```

Solo un repository Git può essere associato ad un progetto TRAC e viceversa, ovvero la corrispondenza è biunivoca. La procedura è la seguente:

1. Collegarsi all'indirizzo web: https://grid1.mercuriofvg.it/admin/
2. Selezionare l'opzione "Associate a git repository to a Trac project".
3. Si aprirà una maschera in cui è possibile scegliere il progetto TRAC, il repository git e quindi il nome da dare a quel repository all'interno di TRAC. Tutti i campi sono obbligatori.
4. Una volta operate le scelte basta un click su Submit per avviare la procedura di associazione.

4.4 Associazione di un repository Git a Jenkins

Quando un repository esiste, ogni utente che possiede un account sul sistema di Continuous Integration Jenkins può costruire un suo personale progetto e associarlo al repository. Questa associazione richiede che l'utente operative abiliti l'utente denominato Jenkins a leggere il repository e la definizione delle variabili necessarie a **Jenkins** per costruire le built. Qui di seguito viene riportato un esempio in cui tre progetti Jenkins sono stati definiti con il nome: ambiente_FENICE, verifica_FENICE e Altro_progetto.

- repo ambiente_FENICE
- option hook.post-receive = jenkins trac
- config jenkins.project = "ambiente_FENICE verifica_FENICE Altro_progetto"
- config jenkins.token = "ambiente_verify"
- config trac.envdir = "/storage/trac/projects/ambiente_FENICE"
- RW+ = operative
- RW = giaiottid
- R = montanarif
- R = jenkins

Successivamente l'utente che possiede un account sul sistema Jenkins può crearsi un progetto di Jenkins e vi associa il repository al progetto seguendo le indicazioni dell'interfaccia web di creazione. Si noti che la variabile associata a jenkins.token, nell'esempio ambiente_verify, deve essere fatta coincidere con il campo **Authentication Token** che si trova nell'interfaccia di Jenkins che definisce il progetto.

4.5 Utilizzo di Git per gestire automaticamente i ticket di TRAC

I compiti attribuiti debbono essere gestiti dall'utente interessato tramite l'interfaccia TRAC dopo aver eseguito il login. L'utente deve ricordare che per ricevere le notifiche relative ad un progetto TRAC deve inserire il proprio indirizzo e-mail tra le preferenze della pagina del progetto.

E' anche possibile eseguire operazioni di chiusura dei compiti assegnati anche direttamente tramite Git, con un'operazione di commit nella quale sono utilizzate, durante la

descrizione del commit, delle parole chiave con un'opportuna sintassi. Viene scoraggiata la chiusura di compiti con questa procedura in quanto l'interfaccia di TRAC permette di essere più dettagliati ed estesi nella descrizione di come il compito è stato svolto, mentre viene incoraggiato l'uso dei riferimenti ai compiti assegnati (ticket) durante i commit. Qui sotto è descritta brevemente la sintassi per la gestione dei ticket tramite commit di Git.

Quando un utente esegue un commit, tramite il software di versionamento Git, gli viene richiesto di commentare il suo commit. Nella scrittura del commento (commit log message) è possibile inserire delle parole chiave, che sono interpretate da software TRAC e che scatenano delle azioni sui ticket e sulla documentazione riportata da TRAC. La sintassi da utilizzare è sintetizzata nel seguito, mentre per ulteriori dettagli si faccia riferimento all'apposita pagina web di TRAC (http://trac.edgewall.org/wiki/CommitTicketUpdater).

La sintassi è la seguente: **[command] [ticketreference]**.

Il carattere ":" (due punti) può essere infrapposto tra il command e il ticketreference, inoltre il ticketreference permette l'utilizzo di più di un numero di riferimento a ticket, i quali debbono essere separati da spazi o virgola ed anche dalla parola "and". Alcuni esempi sono:

close #10 Chiude il ticket numero 10
fixes #10 Chiude e risolve il ticket numero 10
fixes #10 and #11 Chiude e resolve I ticket 10 e 11
see issue 5 Vedi il ticket 5
references #5, #6 Fa riferimento ai ticket 5 e 6
fixes #10 and #11 Chiude e resolve I ticket 10 e 11

Sitografia e Bibliografia

[1] Wikipedia. List of XML and HTML character entity references. http://en.wikipedia.org/wiki/List_of_XML_and_HTML_-character_entity_references (Elenco dei caratteri speciali e loro formattazione in XML e HTML).

[2] TYPO3 Association. Document Library - All About Learning TYPO3. https://typo3.org/documentation/document-library/ (Manuali per l'editing delle pagine in Typo3).

[3] IETF Internet Engineering Task Force. File transfer protocol (ftp). http://tools.ietf.org/html/rfc959 (Il protocollo di trasferimento dei file - FTP).

ARPA FVG-Aria-Elaborati tecnico-scientifici
(http://cmsarpa.regione.fvg.it/cms/tema/aria/utilita
/Documenti_e_presentazioni/tecnico_scientifici.html)