

Testfarm, una eina per millorar el desenvolupament del programari lliure

Pau Arumí
Grup Tecnologia Musical
Univ. Pompeu Fabra
parumi@iua.upf.edu

Mohamed Sordo
Univ. Pompeu Fabra
neomoha@gmail.com

David García
Grup Tecnologia Musical
Univ. Pompeu Fabra
dgarcia@iua.upf.edu

Xavier Amatriain
CREATE
Univ. of California Santa Barbara
Santa Barbara, CA, USA
xavier@create.ucsb.edu

Sumari

En aquest article presentem una nova eina anomenada testfarm que permet automatitzar les tasques de construcció i testeig de programari i monitoritzar el seu estat (per exemple: apunt per release, falla un test, no compila un programa...). Expliquem les seves funcionalitats contrastant-les amb eines similars i posem de relleu la seva utilitat en diferents escenaris d'ús. A continuació exposem les decisions de disseny més rellevants i la metodologia àgil que s'ha seguit per portar a terme el seu desenvolupament. Per finalitzar, posem testfarm dins el context del desenvolupament de programari lliure, tot intentant respondre com aquesta eina (i altres similars) pot ajudar a millorar-ne el procés.

1 Introducció

Testfarm [21, 19] és una aplicació client-servidor escrita en python que permet monitoritzar l'estabilitat i eficiència d'un *projecte* de programari sota desenvolupament, a base de fer constants *tasques* de compilació i tests automàtics en diferents ordinadors *clients* en diferents plataformes (sistema operatiu, distribució, hardware ...) i enviar-ne els informes a un *servidor*.

A la figura 1 tenim un navegador mostrant la pàgina de monitorització d'un *projecte* (CLAM) que ens ofereix el *servidor* de testfarm. La pàgina s'organitza en quatre columnes que corresponen als quatre ordinadors *clients* que té aquest projecte i, a cada columna, es disposen unes caixetes de colors que representen les *tasques* que s'han executat i el seu estatus. Cada caixeta té un link a la seva pàgina de detalls, vegeu la figura 2, que mostra

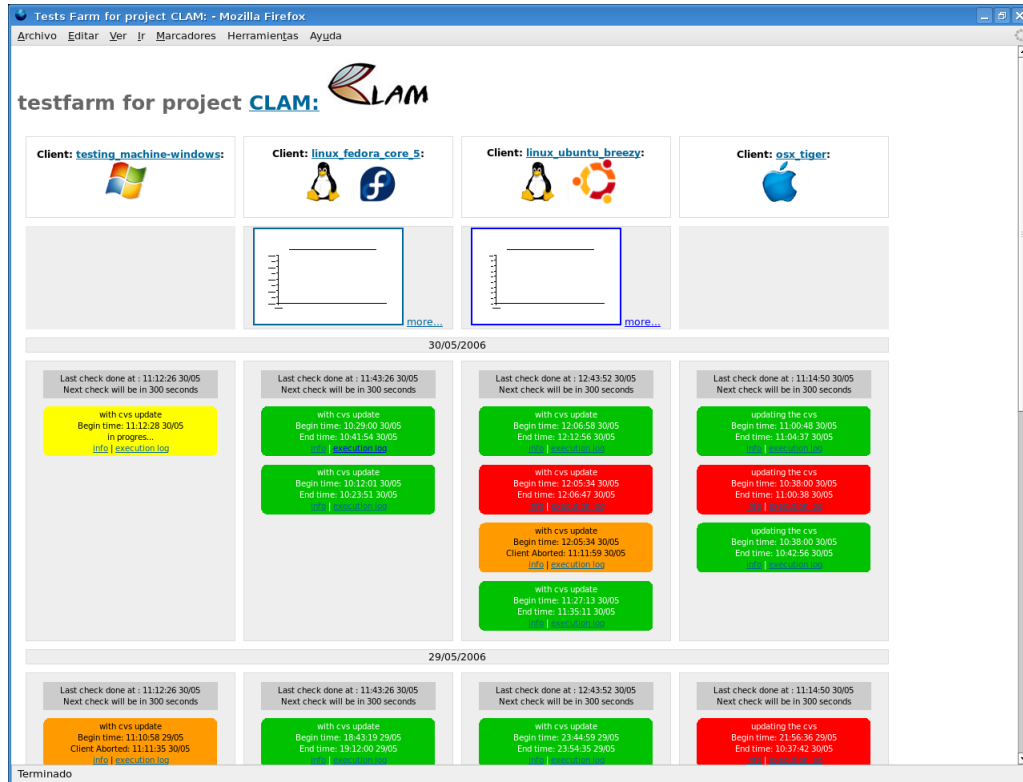


Figura 1: Pàgina de monitorització d'un projecte que usa testfarm. S'aprecia com s'estan executant tasques en quatre plataformes diferents.

les seves sub-tasques amb el log de comandes executades i el seu resultat.

L'article s'estructura de la següent manera. La secció 2 compara testfarm amb eines similars i justifica la necessitat del seu desenvolupament. La secció 3 mostra els beneficis d'usar testfarm en diferents situacions o escenaris de desenvolupament. La secció 4 dibuixa breument el disseny de testfarm i explica el seu procés de desenvolupament. A la secció 5 s'argumenta com testfarm i eines similars poden canviar —a millor, és clar— les metodologies de desenvolupament de programari lliure. Finalment, la secció 6 estableix conclusions i apunta les línies futures de desenvolupament.

2 Com es compara a eines similars ?

Testfarm ha rebut dues fonts d'inspiració principals: En primer lloc, del Mozilla Tinderbox, del qual agafa idees de visualització dels resultats en html. I en segon lloc, del runTests del projecte CLAM, que es pot considerar el predecessor de testfarm ja que comparteix autoria. Del runTests de CLAM, agafa idees de *nivells d'alarmes* i de detecció dels *commits* culpables.

Però testfarm té una aproximació molt més simple que Tinderbox i molt més genèrica que

runTests. I, el més important, té potents mecanismes d'extensibilitat a base de codi Python endollable.

En general, testfarm presenta les següents diferències respecte a les alternatives existents:

- Testfarm no depèn de cap eina apart de *python* (versió 2.4), *apache*, *mod_python* i l'eina de dibuixat de gràfiques *ploticus*. A més, en depenen de forma opcional. Només amb *python* ja pots usar testfarm. A diferència de Mozilla Tinderbox [22] , de Flamebox [15] i de Buildbot [13], testfarm no depèn ni de bonsai [12] ni d'altres eines de sistemes de control de versions (SCV a partir d'ara). Aquest fet és rellevant perquè facilita molt la posada en marxa de testfarm. Configurar bonsai és una tasca carregosa i que requereix un alt nivell tècnic. I tampoc depèn de cap base de dades, tot i que això potser canviarà en un futur.
- Un client de testfarm és útil encara que no es connecti a un servidor, ja que el client pot generar les pàgines de monitorització html localment. I un servidor testfarm no coneix els seus clients. Aquesta independència permet que la tasca d'afegir noves pàgines html de projecte i noves columnes de client sigui totalment automàtica: Només cal que l'script del client faci referència a un nou projecte i nou client.
- Els scripts escrits pels usuaris i que són executats pels clients, són molt simples, basats en comandes de shell i permeten estructurar les tasques, facilitant així la localització dels problemes mirant l'html del servidor. A la figura 3 es pot veure un exemple de script de client. Aquests scripts són extensibles i permeten endollar funcions python per publicar dades d'informació, canviar l'estatus (èxit, falla, etc.) d'una comanda, afegir estadístiques a ser dibuixades, etc. Tot això permet a testfarm que s'integri de forma natural amb llenguatges de programació heterogenis i amb eines que van des d'entorns de desenvolupament (*frameworks*) de testeig automàtic (per exemple, els de la família xUnit, com *MiniCppUnit* ¹ fins a SCVs. Els SCVs més populars són *cvs* i *svn*, però n'hi ha altres que ofereixen funcionalitats addicionals molt interessants com *bazaar* i *bitkeeper*. Veure [10] per un estudi complet.
- I és multi-plataforma, naturalment! Funciona com a mínim a Linux, Mac OSX, FreeBSD i Windows.

¹*MiniCppUnit* [17] ha estat desenvolupat per dos autors d'aquest article (Pau Arumí i David García) i té una aproximació molt més simple i alhora un estil més C++ que les alternatives més usades com ara *cppUnit*.

```
BEGIN_TASK "with cvs update" 2006-06-01-19-15-48

BEGIN_SUBTASK "Deployment"
cd f:\clam-sandboxes [OK]
cd testing-clam [OK]
cvs -q up -dP [OK]

INFO: P scons/libs/processing/SConscript
P test/UnitTests/DescriptorsTests/SpectralPeakDetectTest.cxx

cd f:\clam-sandboxes [OK]
cd testing-clam\scons\libs [OK]
echo seting QTDIR to qt3 path [OK]

INFO: QTDIR set to f:\clam-external-libs\qt\

scons configure prefix=f:\clam-sandboxes local sandbox_path=f:\clam-external-libs
qt_includes=f:\clam-external-libs\qt\include qt_libs=f:\clam-external-libs\qt\lib with_portmidi=1
release=1 double=1 [OK]
scons [FAILURE]

OUTPUT: scons: Reading SConscript files ...
WARNING: at least one module configuration file is missing. Running automatically as a 'scons configure
[display more...]

END_SUBTASK "Deployment"

END_TASK "with cvs update" 2006-06-01-19-20-44 False

Testfarm is free software. Learn about Testfarm.

Terminado
```

Figura 2: Pàgina de detalls d'execució d'una tasca. Es pot observar com la última sub-tasca de "Deployment" ha fallat

Testfarm està sent activament desenvolupat principalment per Pau Arumí i Mohamed Sordo, amb col.laboracions de Bram de Jong, David García i d'altres desenvolupadors de CLAM (C++ Library for Audio i Music) [2, 1].

El projecte va començar per necessitats del projecte CLAM i posteriorment s'ha desenvolupat en el marc d'un projecte final de carrera a la UPF, i continuarà sent desenvolupat com a projecte de programari lliure (PL a partir d'ara) col.laboratiu, amb un interès especial per l'estudi de metodologies de desenvolupament del PL.

3 Escenaris d'ús de testfarm

Desenvolupant en solitari: En aquest cas no necessitem un servidor de testfarm. El propi ordinador client és qui s'encarrega de generar els informes en html. Treballar localment amb testfarm és especialment útil quan tenim uns tests automàtics que triguen molt a executar i volem evitar les esperes. Simplement deixem que el testfarm corri en segon plà, executi les tasques especificades així que detecti canvis al codi (usant eina de control de versions o no) i ens informi quan trobi un problema. És molt útil deixar un navegador obert amb la pàgina html (local) que monitoritza el nostre codi en el nostre entorn de treball.

Quan introduïm un defecte, el navegador ens mostrarà una caixa vermella. Per identificar la causa de l’error, seguim l’enllaç cap als detalls on veurem la sortida complerta² de la comanda que ha fallat. Així de simple.

Desenvolupant en multi-plataforma: Compilar molt freqüentment el codi que estem modificant és una pràctica reconegudament bona. Si tenim tests automàtics, també és important executar-los amb molta freqüència. El punt clau és caçar els errors el més ràpid possible. Però quan treballem amb projectes multi-plataforma el problema es complica. Un codi legal en una plataforma pot no ser-ho en una altra. Això pot ser per molts motius, començant per diferències entre compiladors i llibreries. Quan treballem amb testfarm, cada cop que registrem³ codi al repositori del SCV, tots els clients de testfarm es “desperden” i començaran a executar la bateria de compilacions i tests. Així, només cal anar registrant canvis de codi al SCV de forma molt sovintejada, i anar comprovant que mantenim totes les plataformes en verd. Quan apareix una caixa vermella, seguim el link que ens porta al log de la tasca que ha fallat accedint a l’error concret. Amb aquesta informació intentem corregir el problema al codi, el registrem al SCV i mirem el nou resultat. Donat que testfarm facilita molt afegir nous clients, l’escenari de desenvolupament multi-plataforma és realment potent. Per exemple, en el projecte CLAM [14] —el primer projecte a usar testfarm— s’usen clients de testfarm localitzats a Catalunya i a Califòrnia.

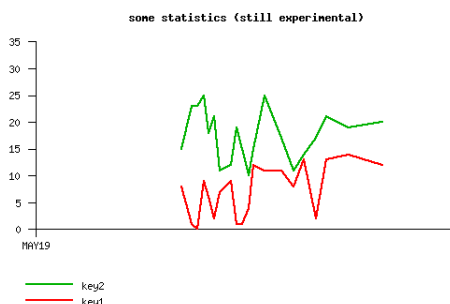


Figura 4: Exemple d’estadística, generada “endollant” una funció python a l’script del client.

Fent optimitzacions : Un script de client testfarm permet afegir estadístiques d’una forma molt simple, a base d’“endollar” funcions python que llegeixen la sortida d’una comanda i retornen un diccionari. Testfarm automàticament recull les dades retornades per aquestes funcions “endollades” i genera diagrames com el de la figura 4. Per exemple, al projecte CLAM interessa que certes funcions de processat de l’audio s’executin usant el menor temps possible. Utilitzant eines command line de *profiling*, com *valgrind* [11] generem unes gràfiques d’eficiència i d’aquesta manera monitoritzem l’evolució del paràmetre

²tant *cerr* com *cout*

³En anglès *checkin* o *commit*

```

#! /usr/bin/python
from testfarm import *
from testfarm.cvstools import *

clam = Task(
    project = Project('CLAM'),
    client = Client('testing_machine-linux_breezy'),
    task_name='doing cvs update'
)

clam.add_deployment( [
    'cd $HOME/clam-sandboxes/testing-clam',
    {CMD: 'cvs -q up -dP', INFO: filter_cvs_update},
    'cd $HOME/clam-sandboxes/testing-clam',
    'scons configure prefix=$HOME/local',
    'scons install',
] )

clam.add_subtask('Unit Tests (with scons)', [
    'cd $HOME/clam-sandboxes/testing-clam',
    'cd scons/tests',
    'scons test.data_path=$HOME/CLAM-TestData clam_prefix=$HOME/local',
    {CMD: 'unit_tests/UnitTests', INFO: lambda x: x, STATS: number_of_unittests},
] )

clam.add_subtask('Music Annotator installation', [
    'cd $HOME/clam-sandboxes/testing-annotator',
    'cd cvs up -dP',
    'scons clam_prefix=$HOME/local',
    'scons install',
] )

Runner( clam,
    continuous = True,
    remote_server_url = 'http://clam.iua.upf.edu/testfarm.server'
)

```

Figura 3: Exemple de script d'usuari. Es pot observar com es poden “endollar” funcions python per generar informació (INFO) i estadístiques (STATS). La funció *number_of_unittests* retorna un diccionari de clau-valors, a partir del qual es crearan diagrames automàticament.

que volem optimitzar. Visualitzar gràficament com ha canviat aquest paràmetre després de cada registre al SCV és de gran utilitat. Òbviament, aquesta tècnica encara és més útil si l'apliquem a tots els clients testfarm. D'aquesta manera detectem subtiletes entre diferents compiladors que poden tenir molt impacte en les optimitzacions. Les estadístiques també poden ser útils per monitoritzar altres tipus de coses. Per exemple la mida de l'executable final, el número d'avisos (warnings) del compilador, el pes del paquet o instal·lador que els usuaris es descarregaran, etc. En futures versions de testfarm tenim previst d'afegir una funcionalitat d'alarma que detecti quan una estadística baixa (o puja) considerablement i ho notifiqui per mail o IRC.

Cada dia una release: Els scripts de clients poden executar qualsevol comanda. No estan limitats a compilació i executar tests. En particular és útil per fer una release cada vegada que estem en verd. En el cas de CLAM quan totes les plataformes estan en verd, cada client construeix els seus paquets (segons la plataforma: .deb, .rpm, setup.exe, .dmg) amb la versió adequada usant la data actual i els pugem a la web de CLAM, junt amb un tarball del codi, apunt per ser descarregats per qualsevol usuari.

Però i si ningú fa tests? En projectes on *no* hi ha tests automàtics, testfarm també pot ser molt útil. El simple fet de detectar ràpidament qualsevol problema de compilació, instal·lació o execució en diferents plataformes ja és, per si sol, molt valuós. A més, cal tenir en compte que els programes que admeten paràmetres per línia de comanda permeten fer tests esquena-contra-esquena (back-to-back) [3] de forma senzilla. Només cal executar una comanda concreta i comparar el resultat amb un fitxer que conté el resultat esperat.

4 El desenvolupament de testfarm

El procés de desenvolupament de testfarm ha seguit el model de les metodologies àgils, en concret hem seguit les pràctiques principals de l'Extreme Programming (XP) [3].

Desenvolupament dirigit per tests: (o test driven development) [4] escrivint tests ⁴ abans de codificar la nova funcionalitat, i fent *refactorings* [6] després de passar cada nou test per mantenir el *disseny simple*. El desenvolupament dirigit per tests ens garanteix que tota l'aplicació està ben coberta per tests i executant sovint els tests ens permet descobrir bugs així que els introduïm. Actualment som pocs desenvolupadors tocant el codi de testfarm, però si la situació canvia, segur que usarem testfarm per monitoritzar el propi testfarm i assegurar-nos que no es trenca cap test.

⁴usant *pyUnit*

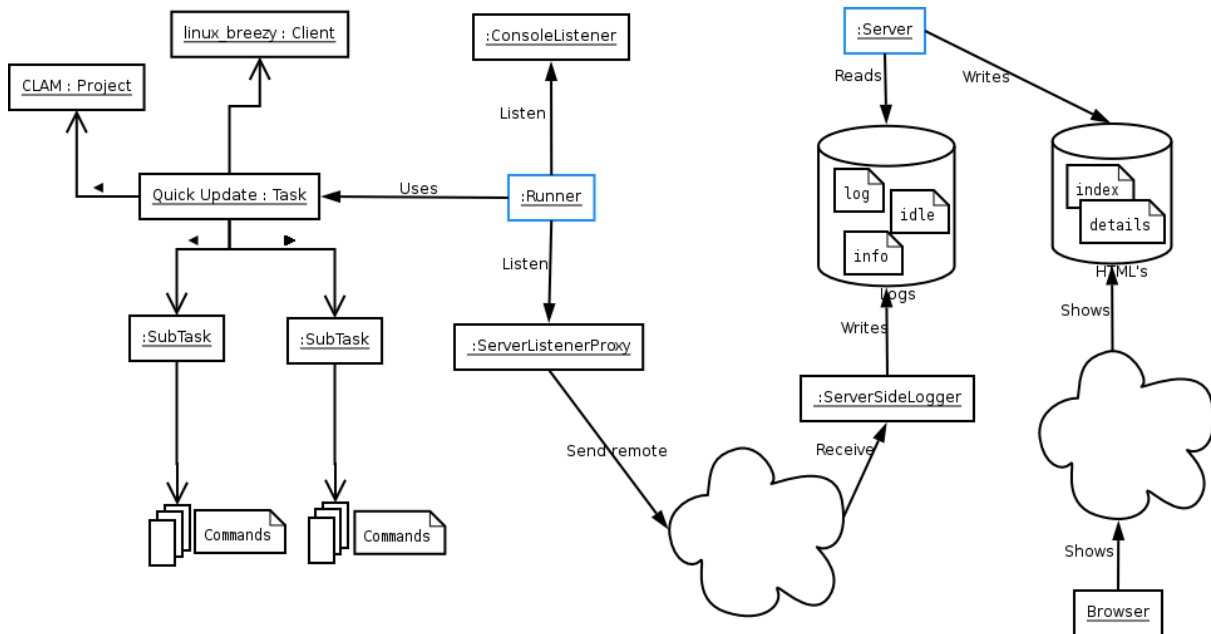


Figura 5: Diagrama objectes de testfarm amb l'exemple de CLAM. En color diferent, els dos objectes “actius” (propietaris d'un thread)

Com és testfarm per dins? La figura 5 ens mostra el diagrama d'objectes del sistema en un escenari d'exemple (no confondre amb un diagrama de classes). Els dos objectes principals són el *Runner* i el *Server* ja que són *actius*, és a dir, executen un procés. El *Runner* està a la màquina client mentre que el *Server* està al servidor. El *Runner* es “desperta” quan detecta que hi ha un nou canvi al SCV i executa la *Task* i les *SubTasks* corresponents a un *Project*. En l'exemple dibuixat el projecte és “CLAM” i la tasca es diu “Quick Update” (aquí “update” es refereix a que es fa un *cvs update* en comptes d'un *checkout net*) El *Runner* envia missatges als dos *Listeners* que té subscrits: el *ConsoleListener* que monitoritza el procés des del terminal del client i el *ServerListenerProxy* que delega els missatges a una entitat remota anomenada *ServerSideLogger*. Aquest és l'encarregat d'afegir entrades als fitxers de logs. Per altra banda, l'objecte actiu *Server* està constantment mirant si hi ha canvis als logs i quan troba canvis actualitza els fitxers html (estàtics) que faci falta, deixant-los apunt per ser visualitzats pels usuaris que apunten el navegador a la pàgina de monitorització del projecte CLAM.

5 Com pot testfarm ajudar al procés de desenvolupament del programari lliure ?

Com hem vist en els escenaris d'ús, testfarm dóna visibilitat de l'estat amb que es troba un projecte. Aquesta visibilitat és interessant sobretot pels desenvolupadors però també

per tota la comunitat al voltant d'un projecte. Per exemple, els usuaris poden tenir una idea de com d'estable és la versió en desenvolupament, quines novetats hi ha (testfarm pot monitoritzar els canvis al SCV, instal·lar-se paquets generats automàticament amb la última versió estable (en el sentit que obté un verd al testfarm). En aquest sentit, ajuda al procés de release fent que el cicle de release sigui tant curt com es vulgui (fins al límit d'una release per cada commit!), portant la celebrada màxima “release often” [9] fins les màximes conseqüències.

Tenir testfarm o eines similars integrades a un projecte són un gran element motivador perquè els desenvolupadors afegixin tests automàtics. Podriem afirmar que converteix els tests en un aspecte central del desenvolupament: testfarm dóna gran visibilitat als tests que fallen i just en el moment de fer un commit. Identificant el “culpable” fins i tot!

Els projectes en entorns lliures sovint es divideixen en mòduls i cada mòdul té un “propietari” (o bé és una propietat compartida per un grup reduït) que és qui revisa els patches rebuts per altres desenvolupadors i assegura l'estabilitat del mòdul. [7] Per tant, és natural que el propietari del mòdul li interessi assegurar una bona cobertura amb tests automàtics i fins i tot exigeixi que els patches amb nova funcionalitat vagin acompanyats de tests.

Permet que els usuaris no programadors participin de noves maneres: oferint temps de processador de les seves màquines. És el propi usuari qui, seguint unes instruccions establertes, pot afegir el seu client al servidor testfarm del projecte. D'aquesta manera un projecte pot estar-se testejant en multitud de plataformes i en entorns ben variats.

Segurament el principal punt diferenciador de testfarm de la resta d'aplicacions similars és que està pensat perquè sigui molt dinàmic per tal que s'adapi bé al desenvolupament de programari en entorns comunitats. Concretament, testfarm és dinàmic en un doble sentit: un servidor testfarm pot ser configurat com a “public”, significat que qualsevol pot crear un projecte en aquell servidor. De forma similar, es poden afegir nous clients a un projecte. El servidor simplement afegix una nova columna quan detecta un nou client. En futures versions de testfarm caldrà afegir gestió de projectes i clients per part del servidor. Però el punt clau és que el fet d'afegir clients i projectes ha de ser essencialment simple.

Considerem també que els servidors de testfarm o similars quedarien ben integrats en eines de desenvolupament col·laboratiu com GForge [16] i per tant, en entorns col·laboratius com sourceforge [18].

S'ha discutit molt sobre la relació entre les metodologies àgils com XP [3] i les metodologies (emergents) del desenvolupament del PL ⁵. Els dos estils de programació comparteixen moltes característiques que alhora els diferencien de les metodologies tradicionals de l'enginyeria del software [8]. Per exemple: requeriments canviants, constant revisions de codi (peer-reviewing), releases sovintejades, documentació i comunicació àgil.

⁵Amb desenvolupament de PL ens referim també a desenvolupament en entorns lliures, és a dir amb una comunitat auto-regulada

Per altra banda, hi ha certes pràctiques de les metodologies àgils que no han trobat tant bona cabuda dins les metodologies PL, destacant de forma significativa: la propietat compartida del codi i els refactorings constants. La tesi dels autors és que eines com testfarm són elements que habiliten l'ús de metodologies àgils en el desenvolupament de PL. Contant amb una base de codi ben coberta per tests automàtics que s'executen cada vegada que hi ha un registre al SCV i amb clients de diferents plataformes, es pot relaxar molt les restriccions d'escriptura al repositori (commits). Cal tenint en compte que la política de registres a la branca principal del SCV no hauria de permetre que es registrin canvis que trenquin cap test a cap client.⁶

Això permet un major nombre de desenvolupadors puguin canviar el codi de forma àgil, sense haver d'esperar un procés de validació manual. A més, la xarxa de salvament que suposa tenir els tests funcionant permet —i de fet, anima— que els desenvolupadors facin refactorings per mantenir el codi net i el disseny simple, apunt per incorporar noves funcionalitats. De la mateixa manera habilita l'ús de la pràctica del desenvolupament dirigit pels tests (*test driven development*) [4]

No és que els refactorings [6] siguin una tècnica nova en el desenvolupament de PL. Ben al contrari: És ben conegut que els projectes més actius de PL estan en evolució constant, pateixen reescriptures totals, es divideixen en nous mòduls per una millor gestió, etc. [5] Però les metodologies àgils ens mostren que en el desenvolupament de PL és possible —i bo— portar els refactorings a un nivell encara més alt, i eines com testfarm permeten que aquests refactorings es facin més sistemàticament, per més gent i amb més qualitat.

6 Conclusions

Amb data de 1 juny de 2006 testfarm es troba a la versió 0.8.2. La versió actual ja és molt usable i estable. Prova d'això és que el projecte CLAM [2] i altres projecte del Grup de Tecnologia Musical a la UPF l'estan usant intensament i amb bons resultats. Per exemple, gràcies a testfarm el desenvolupament de CLAM és, per primera vegada, totalment multi-plataforma: Tot i que els desenvolupadors treballen fonamentalment en l'entorn GNU/Linux, els problemes a altres plataformes (Windows, Mac OSX) es detecten i corregeixen a l'instant. Els scripts testfarm del projecte CLAM, apart de compilar les llibreries i aplicacions i passar els tests automàtics també s'encarreguen de preparar els paquets “instal·ladors” i pujar-los a la web. Així, els usuaris de CLAM poden seguir de ben aprop el desenvolupament amb la comoditat de no haver de compilar. En altres paraules, s'aconsegueix augmentar considerablement la base de testejadors de versions de desenvolupament.

⁶La idea és que els canvis al codi es podrien primer registrar a una (nova) branca de curta vida i només “confirmar” a la branca principal un cop tots els clients de testfarm hagin donat la seva conformitat. Lo interessant de tot plegat és tot pot fer-se de forma ràpida i automàtica.

6.1 Què queda per fer?

Les funcionalitats previstes per les futures versions estan especificades al roadmap de testfarm [20] Aquí podríem destacar: empaquetar testfarm per una fàcil instal·lació, resoldre problemes de concurrència al servidor, afegir un planificador de tasques del client (sheduler), un modul de “alarmes” endollables i millorar la noció temporal a la pàgina de projecte, afegint-hi informació dels commits que s’han fet.

Testfarm ja ha tingut la validació de resultar útil en un projecte de PL com CLAM i en un parell de projectes de codi no obert de la UPF. Tot i així queda pendent una validació a més gran escala. Falta, sobretot, fer experiments amb projectes de mida mitjana i desenvolupament actiu —en són candidats especialment aquells projectes que ja tenen cobertura de tests— i incorporar al projecte testfarm tot el feedback i aportacions possibles.

6.2 Agraïments

Els autors volen agrair les aportacions que ha fet l’equip de desenvolupament de CLAM, a Bram de Jong pel seu feedback constant i al Grup de Tecnologia Musical de la Universitat Pompeu Fabra en general.. El projecte ha estat parcialment subvencionat per l’ajut de la Generalitat de Catalunya, exp. 200/05 ST al projecte CLAM.

Referències

- [1] AMATRIAIN, X., AND ARUMÍ, P. Developing cross-platform audio and music applications with the clam framework. In *Proceedings of the 2005 International Computer Music Conference (ICMC'05)* (2005). in press.
- [2] ARUMÍ, P., AND AMATRIAIN, X. Clam, an object-oriented framework for audio and music. In *Proceedings of 3rd International Linux Audio Conference* (Karlsruhe, Germany, 2005).
- [3] BECK, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 1999.
- [4] BECK, K. *Test-Driven Development: By Example*. Addison-Wesley Professional, 2002.
- [5] ERENKRANTZ, J. Release Management Within Open Source Projects. at <http://www.ics.uci.edu/~wscacchi/Papers/Open-Source-Research/OSSE3-Erenkrantz.pdf>, accessed 20 (2003).
- [6] FOWLER, M. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.
- [7] GONZÁLEZ-BARAHONA, J., AND ROBLES, G. *Free Software Engineering: A Field to Explore*, 2003.
- [8] MASSEY, B. Why OSS Folks Think SE Folks Are Clue-Impaired. *3rd Workshop on Open Source Software Engineering*, 91–97.
- [9] RAYMOND, E. The Cathedral and the Bazaar [On-Line]. Retrieved, August (2002).
- [10] SHAIKH, M., AND CORNFORD, T. Version Control Tools: A Collaborative Vehicle for Learning in F/OS. *Proceedings of the 4th Workshop on Open Source Software Engineering* (2004), 87–91.
- [11] Valgrind website: <http://valgrind.org/>.
- [12] Web del projecte Mozilla Bonsai: <http://www.mozilla.org/projects/bonsai>.
- [13] Web del projecte buildbot: <http://buildbot.sourceforge.net>.
- [14] CLAM website: <http://clam.iua.upf.edu/>.
- [15] Web del projecte Flamebox: <http://flamebox.sourceforge.net>.
- [16] Web del projecte GForge: <http://gforge.org>.
- [17] Web del projecte MiniCppUnit: <http://www.iua.upf.edu/~parumi/MiniCppUnit>.
- [18] Web de l'entorn de desenvolupament col.laboratiu Sourceforge: <http://sourceforge.net>.

- [19] Web del projecte Testfarm : <http://testfarm.sourceforge.net/>.
- [20] Roadmap del projecte Testfarm: http://iua-share.upf.es/wikis/testfarm/index.php/Changelog_and_Roadmap.
- [21] Wiki del projecte Testfarm: <http://iua-share.upf.es/wikis/testfarm/>.
- [22] Web del projecte Mozilla Tinderbox: <http://www.mozilla.org/tinderbox.html>.