Versioning Systems

Motivation and Principles
SVN, git, github

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

UNIVERSITÉ DE RENNES 1

istic Informatique Électronique

Material

https://github.com/acherm/teaching/tree/master/PDL/

Previously

Constitution des groupes

- Gros couac (17 octobre)…
- Au final
  - 3 groupes PCM
  - 2 groupes WebFML
  - 1 groupe 3D (à discuter)
- Est-ce que tout le monde est affecté à un groupe?
- Début vendredi

Quatre objectifs et rendus

- (CD) Comprendre et documenter un projet existant
- (SP1) Sprint 1
- (SP2) Sprint 2
- (PR) Présentation

Quatre objectifs, quatre rendus

- (CD) Comprendre et documenter un projet existant

- Ecrire une documentation développeur sur la page github de votre projet
  - Modifier et compléter
  - Etendre

CD

- Résultat: soumission de la documentation via une « pull request » sur github
  - Version PDF également par mail
- En réalisant cet exercice de documentation, vous devriez être en mesure de mieux comprendre le projet, son organisation, ses technologies, ses fonctionnalités, etc.
  - Indispensable pour S1
- Contribution à un projet open source

CD (1)

- Documentation orientée développeur
- Le document doit permettre de trouver les réponses à des questions comme…
  - (context) A quoi sert le projet?
  - (setup) Comment récupérer le code source? Comment installer et exécuter le projet? Quels sont les pré-requis? Les technologies utilisées? Comment vérifier que mon installation est correcte?
  - (tutorial) Hello world?
  - FAQ?

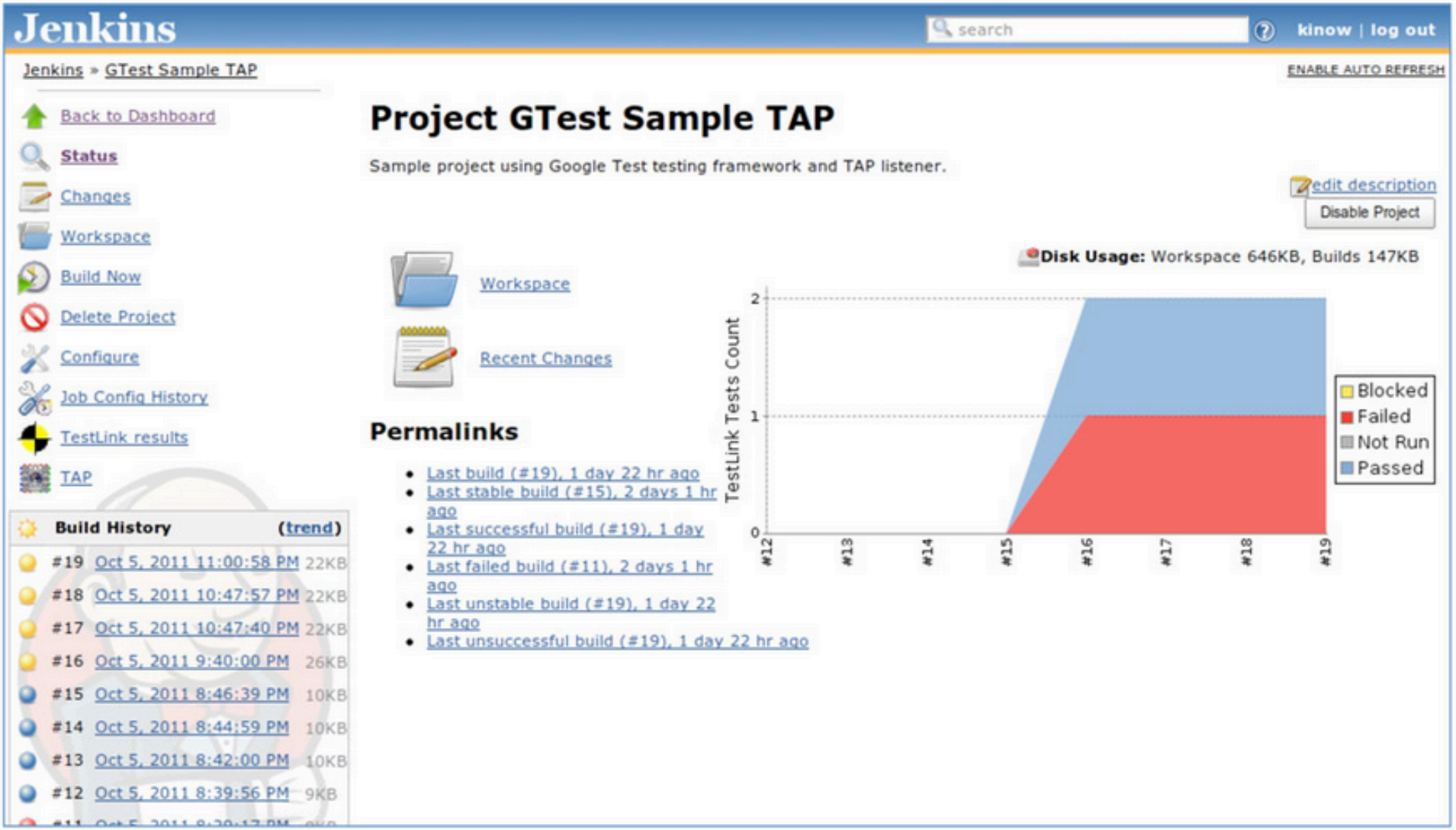
CD (2)

- Documentation orientée développeur
- Le document doit permettre de trouver les réponses à des questions comme…
  - (organisation) Quelle est l'architecture du projet? Point d'entrée? Packages/modules principaux? Classes? Méthodes?
  - (extensibilité) Comment maintenir et étendre la fonctionnalité Y?
  - (installation bis) Comment packager l'application? Comment lancer des cas de tests?

CD (3)

- Documentation orientée développeur
- Le document doit permettre de trouver les réponses à des questions comme…
  - (organisation) Quelle est la licence du projet? Convention de nommage?
  - (limitations) Quels sont les bugs connus?
  - (organisation) Intégration continue? Mise en production?
  - …

Continuous Integration: Jenkins



54 commits | 4 branches | 0 releases | 2 contributors

branch: master | PCM / +

Paerser Fix / update website

ahervieu authored 15 days ago — latest commit 658a3a9ee1

org.diverse.PCM	Paerser Fix / update website	15 days ago
.gitignore	clean repo	27 days ago
README.md	Update README.md	25 days ago

README.md

PCM

This project contains development artifacts used to perform research around product comparison matrices (PCM). This work is currently involving member of the diverse (DIVERsity-centric Software Engineering) research team (http://diverse.irisa.fr/).

Development tools :

- Play
- Maven
- Continous integration with Jenkins (https://ci.inria.fr/)
- We use intellij with the KMF plugin

Framework(s) :

HTTPS clone URL

https://github.com/

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

Current status

We have a basic version of FAMILIAR environment with

- a textual editor (very basic) for specifying scripts
- a console to interact (very basic again)
- way to execute a script
- way to reset
- (partially) the logics for handling a "ksynthesis" session

It works with the Play! framework 2.2.0 (http://www.playframework.com/documentation/2.2.0), the Scala version. We also rely on some Javascripts (ACE editor and jqconsole). More details in the dedicated page.

Setting up in Eclipse

- Download play : http://www.playframework.com/
- Install play as follow : http://www.playframework.com/documentation/2.2.1/Installing
- In webfml directory, start play (in the console enter the command play), in play enter the command eclipse : http://www.playframework.com/documentation/2.2.1/IDE
- In the eclipse webfml project, create a folder lib <!-- * Export a runnable jar from this project : https://github.com/FAMILIAR-project/familiar-language in this destination set: FMLApp/lib/FML-1.2.jar and select "Extract required libraries into generated JAR" -->
- Copy the jar of SWT from your version of eclipse in the *lib* folder (SWT is platform dependent)
- You are ready to work

Compile and Run

- In your source directory open the terminal
- start play (command : play)
- to compile type the command compile
- to run type the command run
- go to the url : localhost:9000

WebFML

README.md

OpenScadScriptAnalyzer

OpenScadScripAnalyzer is an app runtime based on node.js. You can download things(scad files) from www.thingiverse.com and parse them with OpenScadScriptAnalyzer. This project is published on the Docker also.

Downloads from Docker

- **v0.1:** (Sep 5, 2014, based of Node v0.10.31)
  - openScadScriptAnalyzer : **docker pull jiyoungparkkim/openscadscript-analyzer:0.1**
  - mongodb-base: **docker pull jiyoungparkkim/openscadscript-analyzer:0.1**

Run in command

```
$ sudo docker run --name db  -d jiyoungparkkim/mongodb_base:0.1
$ sudo docker run --name web -d -p 9000:9000 --link db:db jiyoungparkkim/openscadscript-analyzer
```

Downloads from git

3D

- **Prerequisites**
  - mongoDB
  - node.js
  - npm
  - bower

CD (4)

- Avant d'écrire la documentation… il faut comprendre le projet
- Etape 0: lire la documentation, récupérer le code source, et installer
  - L'expérience sera très certainement douloureuse (noter les manques, car votre documentation est censée améliorer l'actuelle)
  - Il faut que chaque membre du groupe soit capable de faire fonctionner le projet!
  - Ne pas hésiter également à reporter des bugs via github

CD (5)

- Avant d'écrire la documentation… il faut comprendre le projet
- Etape 0: lire la documentation, récupérer le code source, et installer
- Etape 1: effectuer une modification sur le projet (typiquement une correction d'un bug, un ajout de commentaire, l'écriture d'un cas de test) et soumettre une « pull request »

CD (6)

- Avant d'écrire la documentation… il faut comprendre le projet
- Etape 0: lire la documentation, récupérer le code source, et installer
- Etape 1: effectuer une modification sur le projet
- En parallèle: comprendre les technologies utilisées, les fonctionnalités du système, etc. pour pouvoir adresser toutes les questions…

Current status

We have a basic version of FAMILIAR environment with

- a textual editor (very basic) for specifying scripts
- a console to interact (very basic again)
- way to execute a script
- way to reset
- (partially) the logics for handling a "ksynthesis" session

It works with the Play! framework 2.2.0 (http://www.playframework.com/documentation/2.2.0), the Scala version. We also rely on some Javascripts (ACE editor and jqconsole). More details in the dedicated page.

Setting up in Eclipse

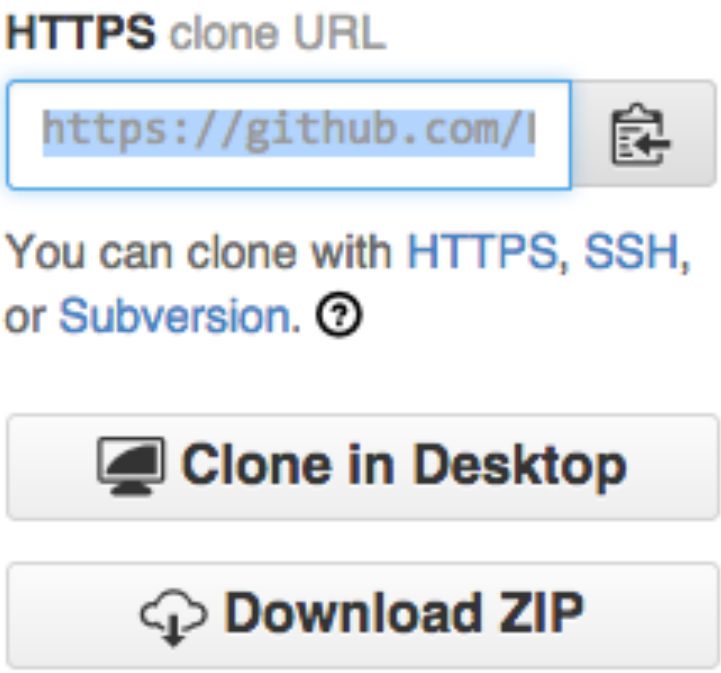
- Download play : http://www.playframework.com/
- Install play as follow : http://www.playframework.com/documentation/2.2.1/Installing
- In webfml directory, start play (in the console enter the command play), in play enter the command eclipse : http://www.playframework.com/documentation/2.2.1/IDE
- In the eclipse webfml project, create a folder lib <!-- * Export a runnable jar from this project : https://github.com/FAMILIAR-project/familiar-language in this destination set: FMLApp/lib/FML-1.2.jar and select "Extract required libraries into generated JAR" -->
- Copy the jar of SWT from your version of eclipse in the *lib* folder (SWT is platform dependent)
- You are ready to work

Compile and Run

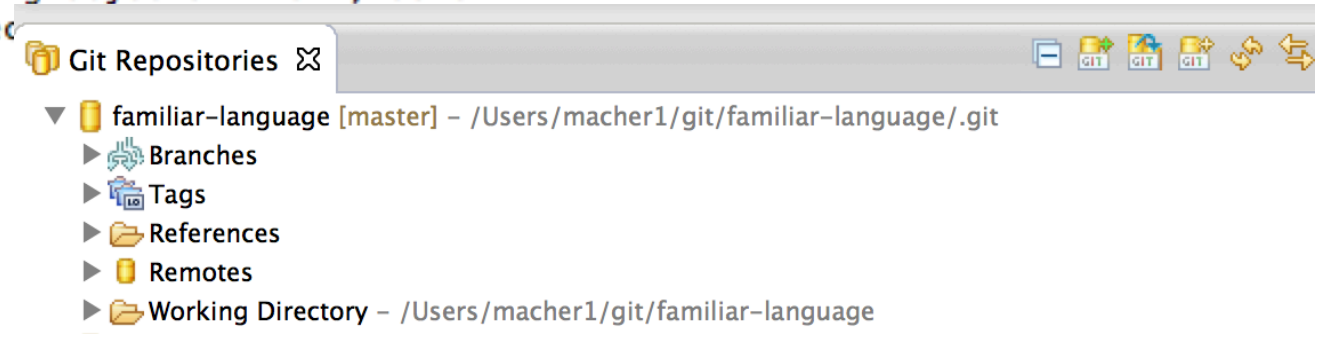
- In your source directory open the terminal
- start play (command : play)
- to compile type the command compile
- to run type the command run
- go to the url : localhost:9000

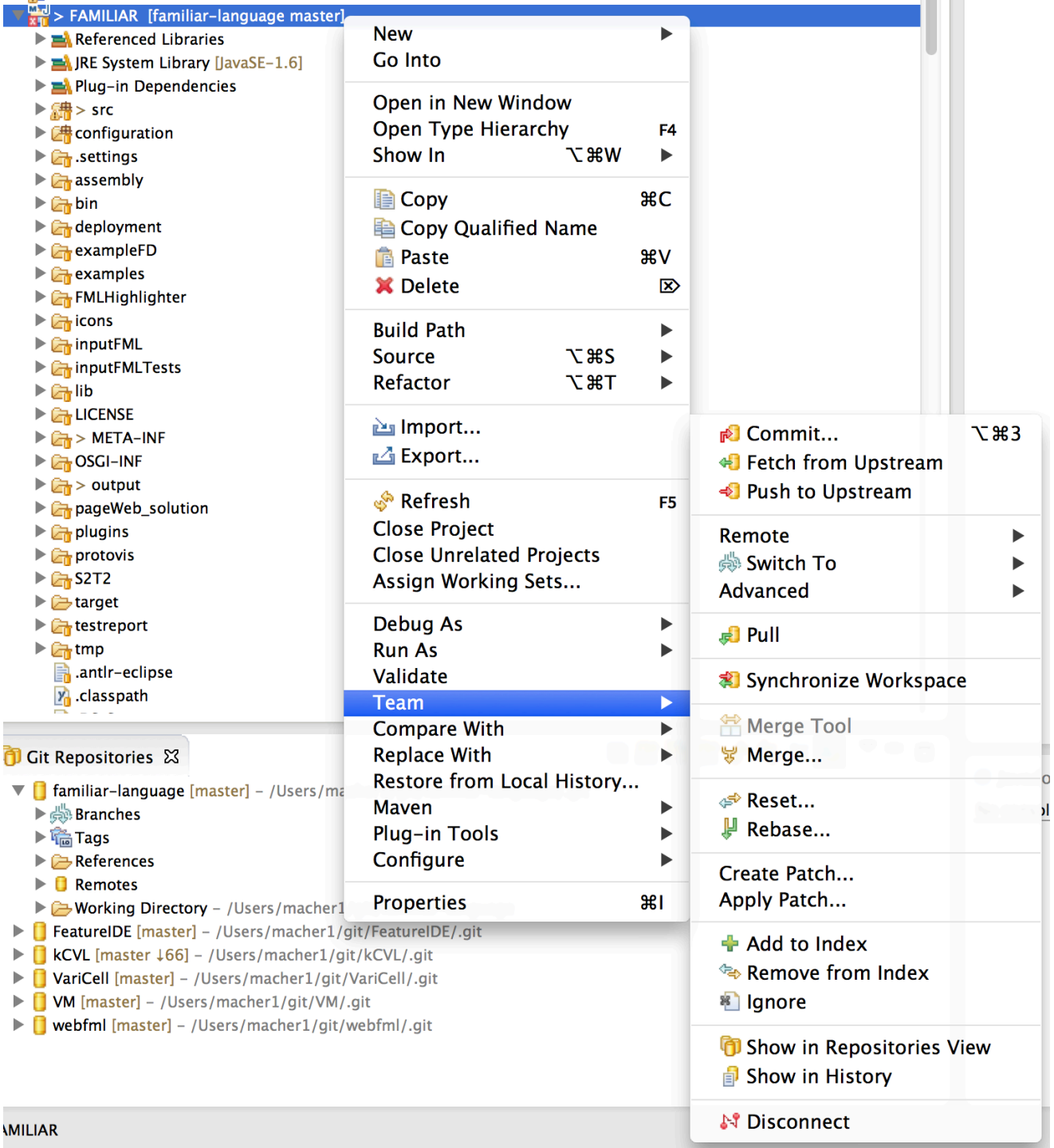
WebFML

https://github.com/FAMILIAR-project/familiar-language.git




```
macher:FML macher1$ git clone https://github.com/FAMILIAR-project/familiar-
language.git
Cloning into 'familiar-language'...
remote: Counting objects: 22811, done.
Receiving objec
```

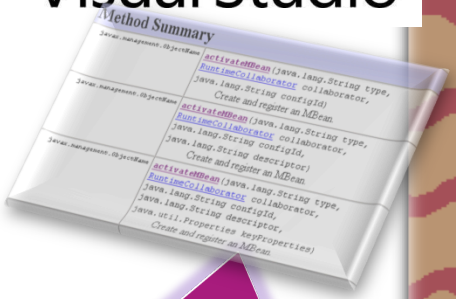


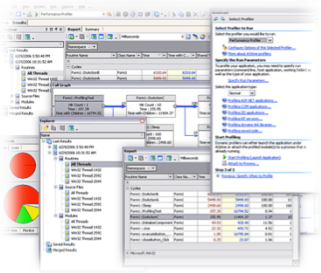

> FAMILIAR [familiar-language master]
Referenced Libraries
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
> src
configuration
.settings
assembly
bin
deployment
exampleFD
examples
FMLHighlighter
icons
inputFML
inputFMLTests
lib
LICENSE
> META-INF
OSGI-INF
> output
pageWeb_solution
plugins
protovis
S2T2
target
testreport
tmp
.antlr-eclipse
.classpath
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Debug As
Run As
Validate
Team
Compare With
Replace With
Restore from Local History...
Maven
Plug-in Tools
Configure
Properties ⌘I
Commit... ⌥⌘3
Fetch from Upstream
Push to Upstream
Remote
Switch To
Advanced
Pull
Synchronize Workspace
Merge Tool
Merge...
Reset...
Rebase...
Create Patch...
Apply Patch...
Add to Index
Remove from Index
Ignore
Show in Repositories View
Show in History
Disconnect
Git Repositories
familiar-language [master] - /Users/ma
Branches
Tags
References
Remotes
Working Directory - /Users/macher1
FeatureIDE [master] - /Users/macher1/git/FeatureIDE/.git
kCVL [master ↓66] - /Users/macher1/git/kCVL/.git
VariCell [master] - /Users/macher1/git/VariCell/.git
VM [master] - /Users/macher1/git/VM/.git
webfml [master] - /Users/macher1/git/webfml/.git
AMILIAR

Versioning in a nutshell


4°

Software Engineering



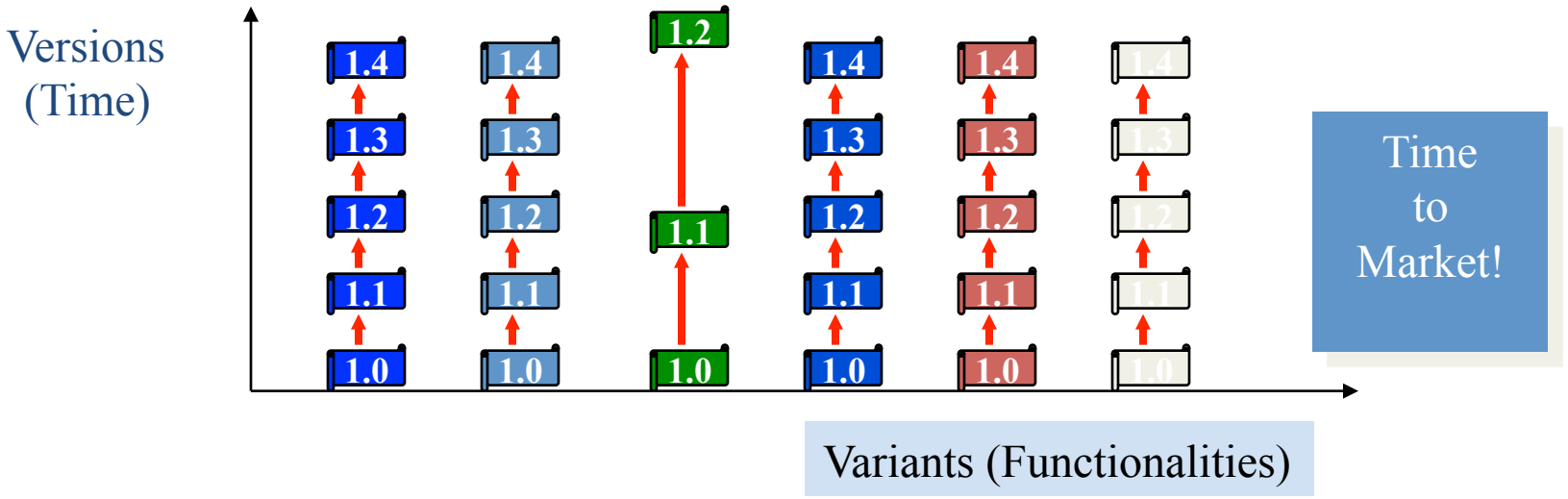


Développer du logiciel ~=

« Multi-Person Construction Of Multi-Versions Programs »

David Parnas, 2014

Software Engineering (back)



The 3 Dimensions of Software Configuration Management

[Estublier et al. 95]

- The Revision dimension
  - Evolution over time
- The Concurrent Activities dimension
  - Many developers are authorized to modify the same configuration item
- The Variant dimension
  - Handle environmental differences

- Even with the help of sophisticated tools, the complexity might be daunting

Versioning of source code

- **Collaborative** software engineering

- To master
  - software development by very large developer teams
  - parallel implementations (experiments, vendors)

- Goals
  - Increase productivity of developers and software robustness
  - Low-down development costs

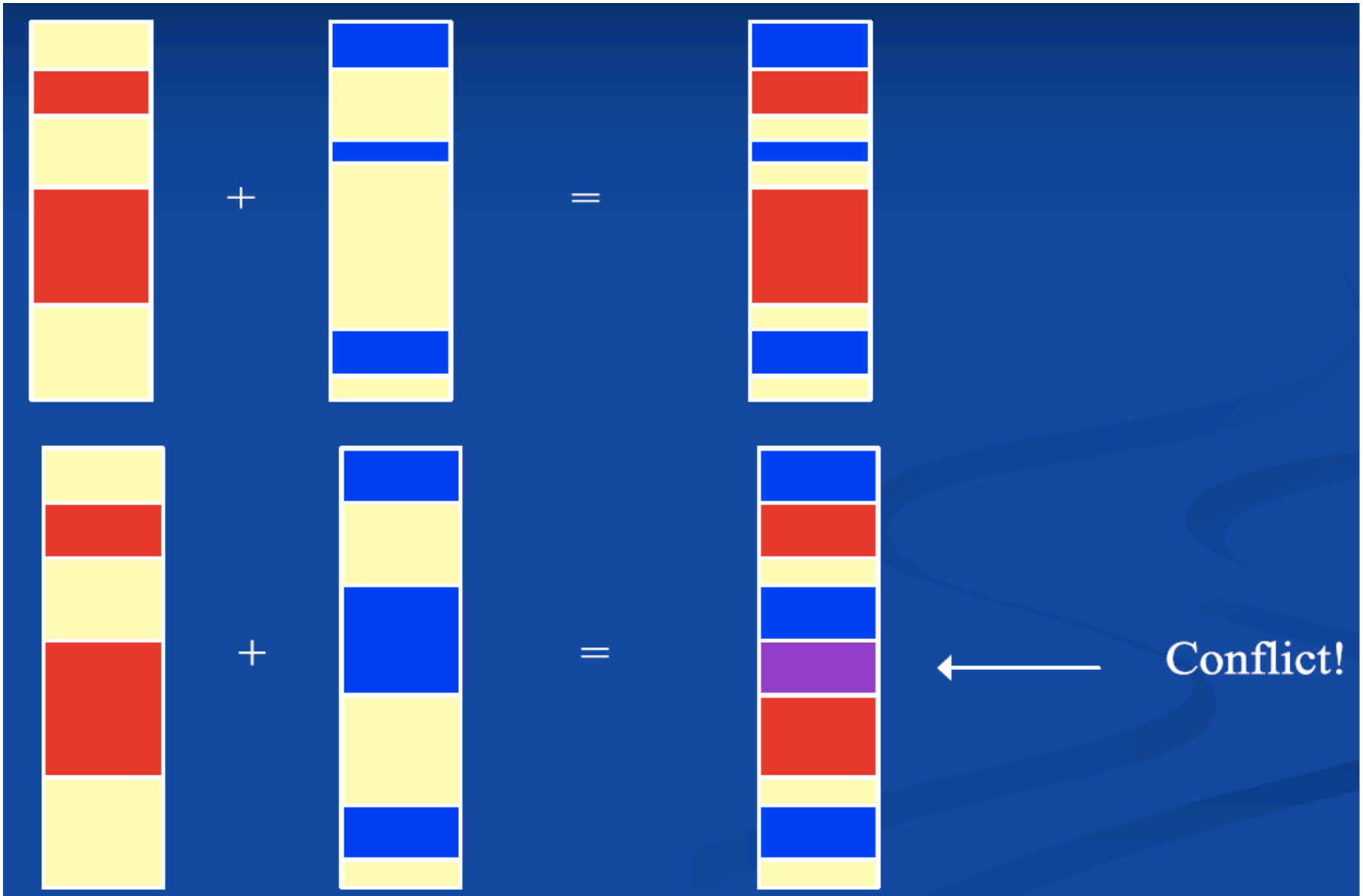
- Manage software system configuration
  - to control software systems evolution
  - evolution tracking (time-machine)
  - issue and bug tracking

Versioning : What for?

- **History** of versions
  - back to an older version in case of errors
- **Alternative** versions (branching)
  - different design/implementations (maybe experimentals) for the same module
- **Collaborative** access by many developers
  - audit modification history
    - how many commits by X ?
    - when most of the commits are done?
    - ...

Concurrency management

with a simple picture (or why it is not that simple)



Concurrency control: approaches

- Doing nothing!
- Lock-Modify-Unlock (Pessimistic)
  - SCCS, RCS
  - Decrease productivity
- Copy-Modify-Merge (Optimistic)
  - Conflicts resolution when concurrent modifications (which are actually rare)
    - Merge, Selection, …
  - CVS, SVN : Client level resolution
- Policy-based
  - Merging and validation process for each code contribution

Concept of Version

- Trunk
  - main development
- Branches
  - Alternatives to trunk
    - Different design/implementation (experimental), vendor-specific
- Revisions
  - Sequence of versions
- Tags
  - Symbolic references to revisions (Tiger, LongHorn, …)
    - Represent a public release (R), a milestone (M)
- Branch merging

Git


GIT

≠


SUBVERSION

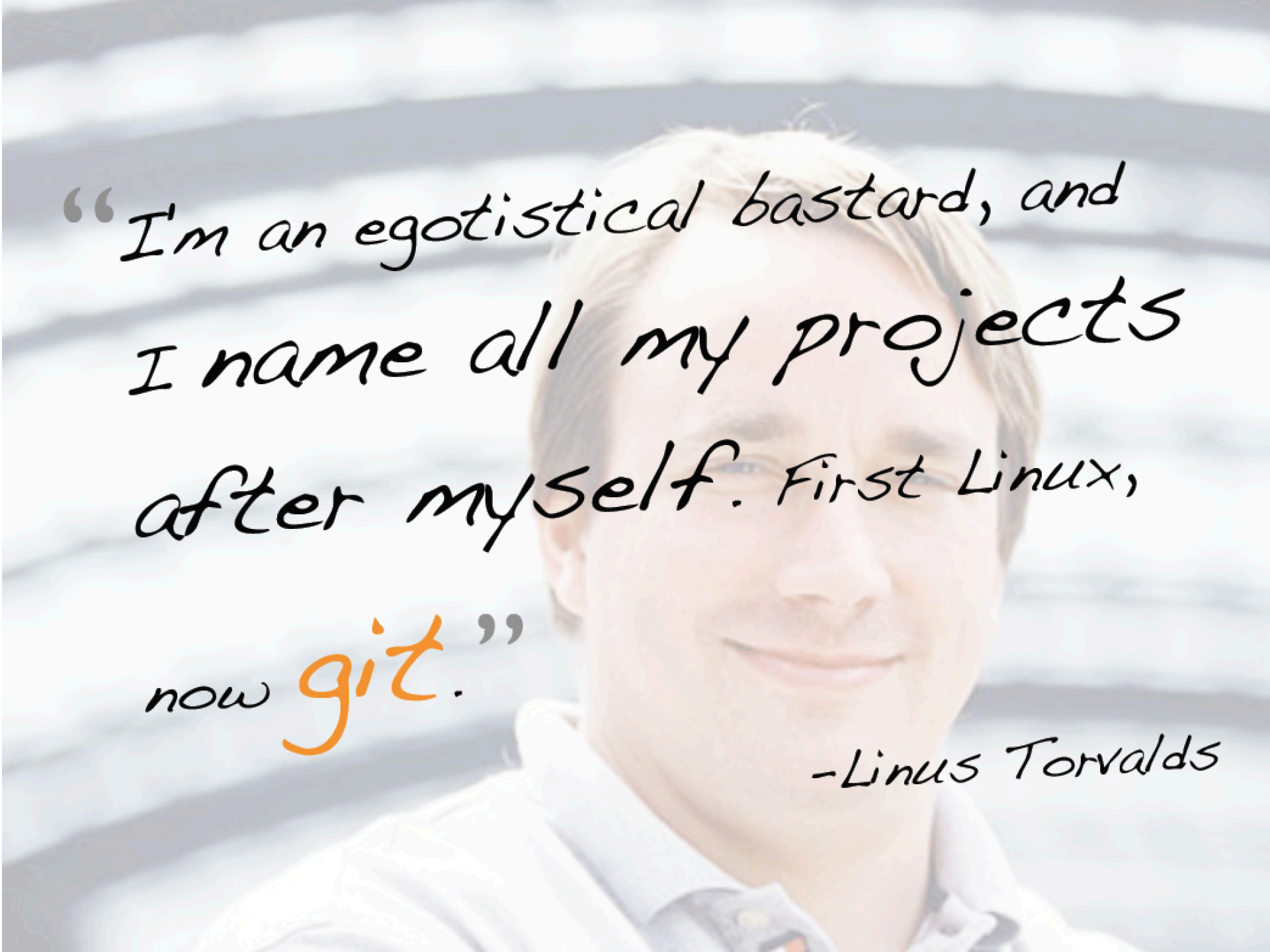

github
SOCIAL CODING
Signup and Pricing
Explore GitHub
Features
Blog
Login
1,067,856 people hosting over 3,012,331 git repositories
jQuery, reddit, Sparkle, curl, Ruby on Rails, node.js, ClickToFlash, Erlang/OTP, CakePHP, Redis, and many more
Find any repository
twitter
facebook
rackspace HOSTING
digg
YAHOO!
shopify
EMI
six apart
git /'gɪt/
Git is an extremely fast, efficient, distributed version control system ideal for the collaborative development of software.
git·hub /'gɪt,hʌb/
GitHub is the best way to collaborate with others. Fork, send pull requests and manage all your public and private git repositories.

« Git »

-noun

an unpleasant
or contemptible **person**

-Oxford English Dictionary


"I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."
-Linus Torvalds

Git Advantages

- Resilience
  - No one repository has more data than any other
- Speed
  - Very fast operations
- Space
  - Compression can be done across repository not just per file
  - Minimizes local size as well as push/pull data transfers
- Simplicity
  - Object model is very simple
- Large userbase with robust tools

Important Commands

- Getting a Repository
  - git init
  - git clone
- Commits
  - git add
  - git commit
- Get changes with
  - git fetch (fetches and merges)
  - git pull
- Propagate changes with
  - git push

Mainly based on the content of

http://people.irisa.fr/Anthony.Baire/git/

(one of the best introduction of Git/SVN)

Github

Current status

We have a basic version of FAMILIAR environment with

- a textual editor (very basic) for specifying scripts
- a console to interact (very basic again)
- way to execute a script
- way to reset
- (partially) the logics for handling a "ksynthesis" session

It works with the Play! framework 2.2.0 (http://www.playframework.com/documentation/2.2.0), the Scala version. We also rely on some Javascripts (ACE editor and jqconsole). More details in the dedicated page.

Setting up in Eclipse

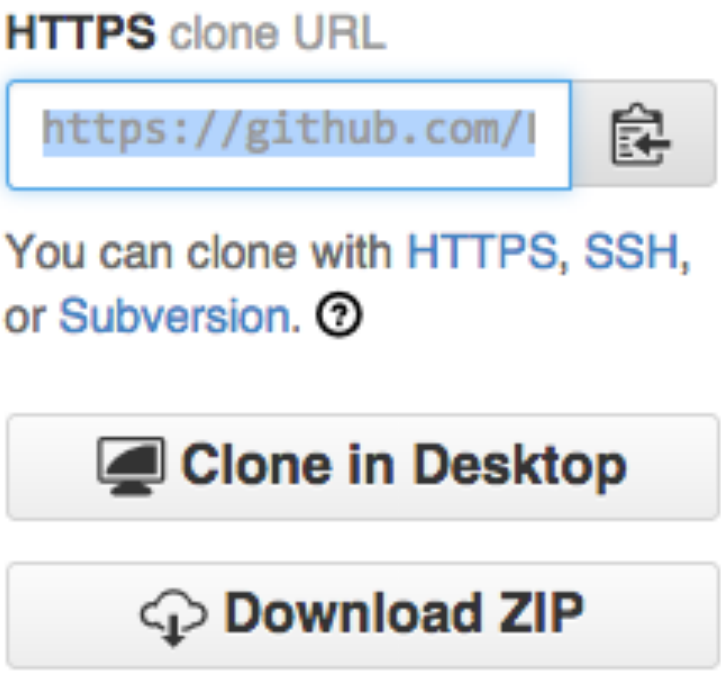
- Download play : http://www.playframework.com/
- Install play as follow : http://www.playframework.com/documentation/2.2.1/Installing
- In webfml directory, start play (in the console enter the command play), in play enter the command eclipse : http://www.playframework.com/documentation/2.2.1/IDE
- In the eclipse webfml project, create a folder lib <!-- * Export a runnable jar from this project : https://github.com/FAMILIAR-project/familiar-language in this destination set: FMLApp/lib/FML-1.2.jar and select "Extract required libraries into generated JAR" -->
- Copy the jar of SWT from your version of eclipse in the *lib* folder (SWT is platform dependent)
- You are ready to work

Compile and Run

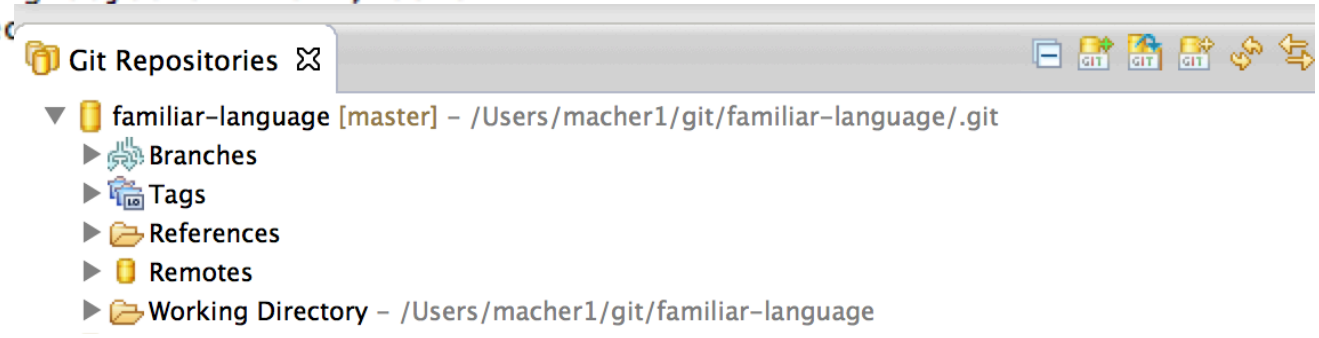
- In your source directory open the terminal
- start play (command : play)
- to compile type the command compile
- to run type the command run
- go to the url : localhost:9000

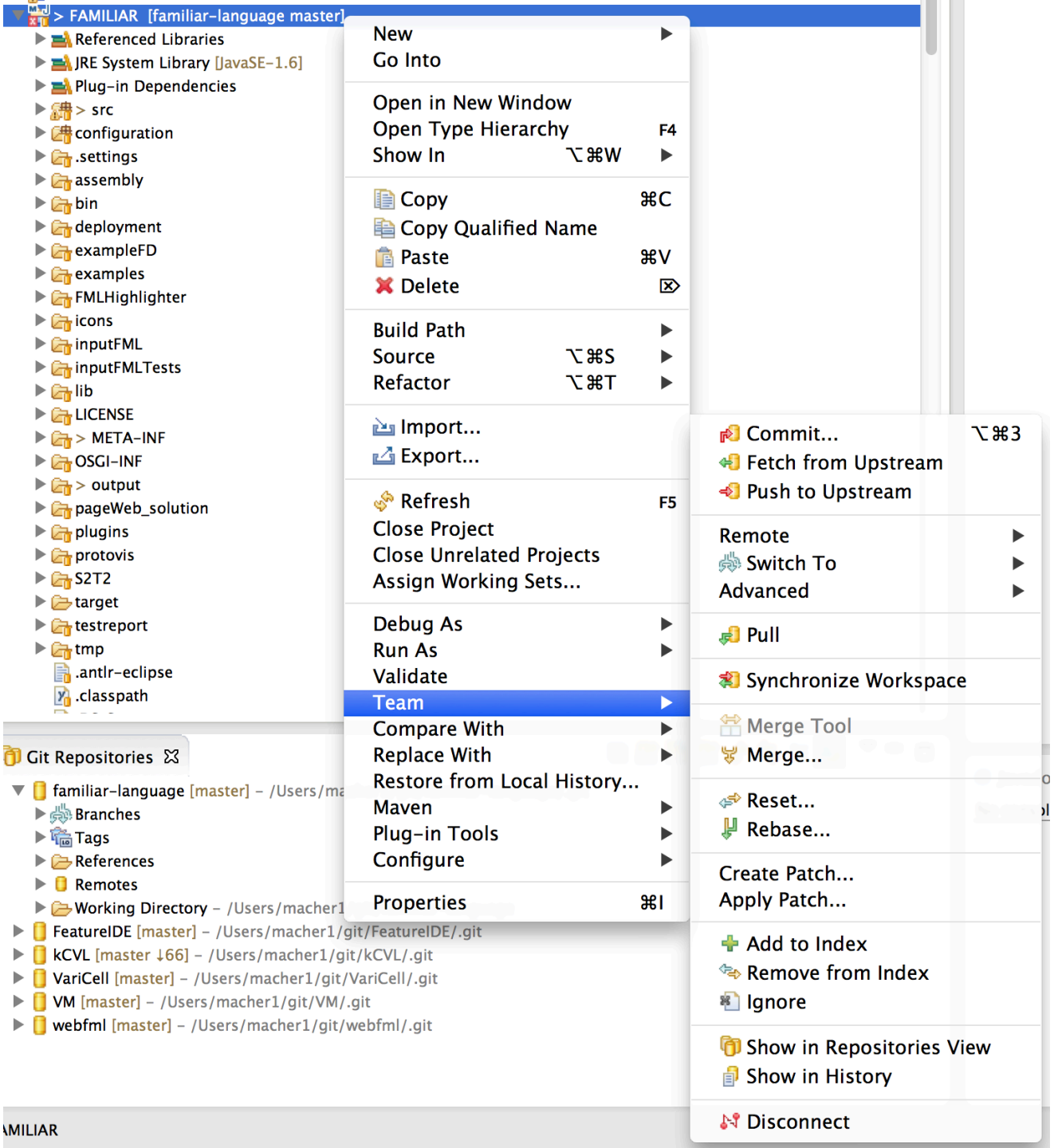
WebFML

https://github.com/FAMILIAR-project/familiar-language.git




```
macher:FML macher1$ git clone https://github.com/FAMILIAR-project/familiar-
language.git
Cloning into 'familiar-language'...
remote: Counting objects: 22811, done.
Receiving objec
```




> FAMILIAR [familiar-language master]
Referenced Libraries
JRE System Library [JavaSE-1.6]
Plug-in Dependencies
> src
configuration
.settings
assembly
bin
deployment
exampleFD
examples
FMLHighlighter
icons
inputFML
inputFMLTests
lib
LICENSE
> META-INF
OSGI-INF
> output
pageWeb_solution
plugins
protovis
S2T2
target
testreport
tmp
.antlr-eclipse
.classpath
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Debug As
Run As
Validate
Team
Compare With
Replace With
Restore from Local History...
Maven
Plug-in Tools
Configure
Properties ⌘I
Commit... ⌥⌘3
Fetch from Upstream
Push to Upstream
Remote
Switch To
Advanced
Pull
Synchronize Workspace
Merge Tool
Merge...
Reset...
Rebase...
Create Patch...
Apply Patch...
Add to Index
Remove from Index
Ignore
Show in Repositories View
Show in History
Disconnect
Git Repositories
familiar-language [master] - /Users/ma
Branches
Tags
References
Remotes
Working Directory - /Users/macher1
FeatureIDE [master] - /Users/macher1/git/FeatureIDE/.git
kCVL [master ↓66] - /Users/macher1/git/kCVL/.git
VariCell [master] - /Users/macher1/git/VariCell/.git
VM [master] - /Users/macher1/git/VM/.git
webfml [master] - /Users/macher1/git/webfml/.git
AMILIAR

Pull Request

Offrir un ensemble cohérent de changements à un responsable de projet

« Je prends la responsabilité de ce changement; êtes-vous d'accord pour l'intégrer? »

https://help.github.com/articles/using-pull-requests/

Branch

Develop features on a branch and create a pull request to get changes reviewed.

Discuss

Discuss and approve code changes related to the pull request.

Merge

Merge the branch with the click of a button.

Documentation (CD) + Code (S1)

Conclusion

Résumé: Git + Projet

- Versioning systems
  - Mandatory for your professional career
  - Multi-person construction of multi-versions programs
- SVN, Git
- Github
  - Git support with facilities
  - Pull request

- You can start your project!