

Team Collaboration with TestingWhiz

TestingWhiz 6.0 11/20/2017



Contents

What is Collaborative Software Development?2
How Does Collaboration Work in Automated Software Testing?2
Need for Version Control in Collaborative Approach2
Benefits of Using Version Control System in Collaborative Approach2
Following Standard Practice – To be a Good Team Player2
Restoring Different Versions without Messing Things2
Unlimited Tracking of Who did What3
TestingWhiz Features enabling Collaboration3
Methods3
Data Tables3
Import Test Wizard3
Final Step12



What is Collaborative Software Development?

Collaborative Software Development is an approach allowing multiple team members from analysis, development and quality assurance to share work, ideas, and tasks for achieving common goals. The method specifically aims to increase the success of teams as they engage in collaborative problem solving, thereby goals of the project.

How Does Collaboration Work in Automated Software Testing?

In the context of automated software testing, collaboration can occur between developers and testers in terms of creating test plans, test scenarios, test scripts, and test data jointly along with analyzing test results.

Need for Version Control in Collaborative Approach

Collaborative approach is not limited to sharing a common workspace, environment or a shared files/folder. Rather, it is enabling teams to work on a common file yet impart absolute freedom to make changes at any time with Version Control System.

In traditional set-up, team members verbally inform about working on a file so that other team members do not work on it to avoid duplication. This process is extremely prone to errors, as someone will overwrite someone else's changes, sooner or later.

With a version control system in place, everybody in the team can work absolutely freely - on any file at any time. The version control system will later allow you to merge all the changes into a common version. There's no question where the latest version of a file or the whole project is. It's in a common, central place, i.e. your version control system.

Benefits of Using Version Control System in Collaborative Approach

Following Standard Practice – To be a Good Team Player

Version control system allows independent working in a team or on your own. It acknowledges that there is only one project. Therefore, there's only one version on your disk that you're currently working on. Everything else -- all the past versions and variants are neatly packed up inside the VCS. When you need it, you can request any version at any time and you'll have a snapshot of the complete project right at hand.

e.g. Test Case 1 for Jira ID 1 committed to a repository as Version 1. Later there was a negative scenario added for the same requirement and you label it as Version 2.

Restoring Different Versions without Messing Things

Being able to restore older versions of a file (or even the whole project) effectively means one thing: you can't mess up! If the changes you've made lately prove to be redundant, you can simply undo them in a few clicks. Knowing this should make you a lot more relaxed when working on important bits of a project.

e.g. Test case 2 for Jira ID 2 committed to a repository as Version 1. Someone modified the case and made it unstable and committed it as Version 2. You know the issue, revert to Version 1.



Unlimited Tracking of Who did What

Every time you save a new version of your project, your VCS requires you to provide a short description of what was changed. Additionally, (if it's a code / text file), you can see what exactly was changed in the file's content. This helps you understand how your project evolved between versions.

e.g. Test case 3 for Jira id 3 committed by developer1. Test case 3 got modified by developer 2 for covering exception handling reasons. Test case 3 got modified by developer 3 for optimization reasons.

TestingWhiz Features enabling Collaboration

Methods

Any method that performs a single conceptual task should be able to stand on its own as a first-class candidate for reuse and hence labeled as a Method. TestingWhiz allows creating Methods out of Test Steps. These Methods can take parameters, can call other methods, can call itself, can make use of data tables, can reference parameters, can return values, can perform conditional execution and lot more. Once Methods are created, they can be called in a test case using operation named 'Call Method'.

Besides calling Methods from the same Test Project, TestingWhiz also allows calling Methods from other TestingWhiz project file, allowing dynamic linking features. This feature offers unique reusability and maintenance benefits to enterprise teams.

Data Tables

Data tables are interim data storage locations in TestingWhiz simulating an Excel-like interface. These data tables can be populated by querying a database, importing excel files, pasting data from the clipboard or generating data using built-in test data generator. Once data is inside data table, it can be shared across Test Project.

Import Test Wizard

TestingWhiz offers a unique wizard to import Test Cases, Methods and Data Tables. Using this wizard, scripts can be merged and updated easily. Below are the steps demonstrating import wizard capabilities from the tool for merging TestingWhiz projects.



Step 1: Use File > Import Test Project

🗳 Te	estingV	Vhiz Enterp	orise - Ci	\Users\nsan	nghavi\Desktop\SVNCopy.twiz
File	Edit	Settings	Tools	Scheduler	Support View Help
	New			Ctrl+N	
	Open			Ctrl+0	
	Open	Sample		+	
	Save			Ctrl+S	
	Save a	s			
	Impor	t Test Proje	ect		
	Impor	t From Exc	el	Ctrl+I	
	Export	to Excel		Ctrl+E	xchange_Rate
	Comp	are File			cast Id
	Share	twiz Projec	t		
	Licens	e Informat	ion		
	Exit			Alt+F4	
					-

Step 2: Select the TestingWhiz project from where you want to import data

CartestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\Merge.twiz		
File Edit Settings Tools Scheduler Support View Help		
📔 📕 💐 💐 🔾 - 🕨 - 🕕	🖂 🔲 💽 📓 📰 🛛 🚖 🐝 🕯	}
😢 Test Project 🗐 Data Table 🧧 *Objects 🗖 🗆	😢 Import Test Project	×)
$\blacksquare \blacksquare $	Browse File	
4 😱 Regression Suite	Select a TestingWhiz Project file.	Screenshot
File1_Suite1		vascriptki
TC_Perform Set Variable and Global Variable		
TC_Global Variable_Validation	Select File D:\Kevin\Project\loylogic_Prac\File1.twiz	
TC_Perform_Accept Alert		
TC Perform Reject Alert		
TC Perform Mouse Over		
TC Perform Scroll Up Scroll Down		
TC Perform Right Click On Image		
TC Perform Close Pop Up Window		
TC Perform Doubleclick		
TC Perform DragnDron		
Search		
Ath Bush	< Back Next > Finish Cancel	
Service and the service and th		
Call Method =	19 Fellolini Jeuglobalvallable	
P 👷 Capture	20 Write Message To Set Va	riable As text is w 🗌
Check		

Step 3: Select test cases you want to import



🖂 💿 🌌 🛤 🚖 🔂 🐳 🕸
😢 Import Test Project
Import Test Object
Elements in green are new while the ones in red are conflicted.
A 🙀 Regression Suite
Image: A
TC_Perform Set Variable and Global Variable
TC_Global Variable_Validation
TC Perform Reject Alert
TC Perform Mouse Over
TC_Perform_Scroll Up_Scroll Down
TC_Perform_Right Click On Image
all TC Deform Close Don Un Window
< Back Next > Finish Cancel

Step 4: Select data tables you want to import





Step 5: Select methods you want to import

the second s	
	🗰 🕸 👘
😢 Import Test Project	
Import Method	.
Methods in green are new while the ones in red are conflicted.	
Methods	
Methodi	
< Back Next > Finish	Cancel
19 Fellolli Seligiobalvallable	

Below are some sample scenarios demonstrating state, pre-merge condition and post merge outcome.



Person 1	Person 2	Action	Expected outcome
File1.Suite 1.TC 1-10	File1.Suite 2 TC 1-10	Merge	Generate a new merged file with Suite 1 – TC 1-10
			from file 1 and Suite 2 – TC 1-10 from file 2



Person1 File_Suit1 having 10 Test Case







Person2 – File_Suit2 having 10 Test Case

StestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\Merge.twiz
File Edit Settings Tools Scheduler Support View Help
🥃 📕 🛒 🛒 🔾 • 🕨 🛛
😢 *Test Project 🗐 Data Table 🧧 *Objects 🗆 🗆
$\checkmark \And + - \land \checkmark \swarrow$
4 🙀 Regression Suite
File1_Suite1
File1_Suite2
🌞 Test Command 🗾 Methods 🛛 💌 Variable 🕂 — 🗖 🗖

After Merge [Import Test Project] from Person 1 File Suit1 & Person 2 File Suit2



Person 1	Person 2	Action	Expected outcome
File1.Suite 3. TC 1	File2.Suite 3. TC 2	Merge	Generate a new merged file with Suite 1 – TC 1-
			10, Suite 2 – TC 1-10, Suite 3 – TC 1-2

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\File1.twiz	
File Edit Settings Tools Scheduler Support View Help	
📒 📕 🕺 🛒 🔾 - 🕨 - 🕕	
😢 Test Project 🗐 Data Table 🧧 Objects 🗌 🗌	5
$\blacksquare \blacksquare $	
4 🙀 Regression Suite	
File1_Suite1	
File1_Suite2	
File1_Suite3	
HER TC2	
Test Command Methods 🛛 🕼 Variable 🕂 🗕 🗆	

Person1 File_Suit3 TC1



After Import Person1 Suit3_TC1

PrestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\File2.twiz
File Edit Settings Tools Scheduler Support View Help
_ 📒 📕 🛒 🛒 ◯ - ▶- 🕕
😢 Test Project 🗐 Data Table 🧧 *Objects 🗌 🗌
4 🙀 Regression Suite
File1_Suite1
File1_Suite2
4 🛅 File1_Suite3
泰 TC1
💩 TC2
🌞 Test Command 🧧 Methods 🛛 🕼 Variable 🕂 🗕 🗖

Person2 – File Suit3 TC2



After Import Person2 Suit3_TC2



Person 1	Person 2	Action	Expected outcome
File method 1	File method 1	Merge	Keep Person 1's changes

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\File1.twiz
File Edit Settings Tools Scheduler Support View Help
📒 📕 🛒 🛒 🔾 - 🕨 - 🕕
😢 Test Project 🗐 Data Table 🥃 *Objects 🗖 🗆
$\blacksquare \blacksquare $
 Regression Suite File1_Suite1 File1_Suite2 File1_Suite3
🏘 Test Command 🗾 Methods 🛛 💷 Variable 🕂 — 🗆 🗆
Search
Methods Method1

Person1 File Method1

Stream Contemporation - D:\Kevin\Project\loylogic_Prac\Merge.twiz
File Edit Settings Tools Scheduler Support View Help
🧧 📕 🛒 🛒 🔾 - 🕨 - 🛛 [
😂 Test Project 🗐 Data Table 🧧 *Objects 🖓 🗆
🔺 📷 Regression Suite 🔄
File1_Suite1
TC_Perform Set Variable and Global Variable
TC_Global Variable_Validation
🌞 Test Command 🗾 *Methods 🛛 🕼 Variable 🕂 — 🗆 🗖
Search
Methods Method1

After Import method1 from Person1 File



Person2 – File Method1

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\Merge.twiz
File Edit Settings Tools Scheduler Support View Help
🔰 📕 🐺 🕅 🔾 - 🕨 - 🛛
隆 Test Project 🕤 Data Table 🧧 *Objects 🗆 🗆
$\circledast \circledast + - \blacktriangle \checkmark \swarrow$
Image: A generation Suite Image: A generation Suite Image: A generation Suite Image: A generation Suite
TC_Perform Set Variable and Global Variable
TC_Global Variable_Validation
🔹 Test Command 🎫 *Methods 🛛 🗔 Variable 🕂 — 🖓 🗖
Search
Methods
Method1
Method1_File2.twiz

TestingWhiz preserves both the files if they are having same name. User can decide to discard one based on code inside.



Person 1	Person 2	Action	Expected outcome
File method 1	File method 1	Merge	Keep Person 2's changes

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\File1.twiz
File Edit Settings Tools Scheduler Support View Help
🧧 📕 🕺 🕺 🔾 - 🕨 - 🕕
😢 Test Project 🗐 Data Table 🧧 *Objects 🗆 🗆
 Regression Suite File1_Suite1 File1_Suite2 File1_Suite3
🌞 Test Command 🗾 Methods 🛛 🔝 🕼 Variable 🕂 — 🗖 🗖
Search
Methods





After Import method1 from Person1 File

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\File2.twiz
File Edit Settings Tools Scheduler Support View Help
📁 📕 🕺 🛒 🔾 - 🕨 🏾 [
🎱 Test Project 🗊 Data Table 🥃 *Objects 🖓 🗆
$\mathbf{v} \otimes \mathbf{v} - \mathbf{v} \mathbf{v}$
🔺 📷 Regression Suite 🔷
File1_Suite1
TC_Perform Set Variable and Global Variable
TC_Global Variable_Validation
🌞 Test Command 🗾 Methods 🛛 💌 Variable 🕂 — 🗖 🗖
Search
Methods

Person2 – File Method1

TestingWhiz Enterprise - D:\Kevin\Project\loylogic_Prac\Merge.twiz
File Edit Settings Tools Scheduler Support View Help
🍯 📕 🕺 🛒 🔾 - 🕨 - 🛛 [
😢 Test Project 🗐 Data Table 🧧 *Objects 🗆 🗆
$\mathbf{x} \mathbf{x} + \mathbf{-} \mathbf{x} \mathbf{z}$
4 🙀 Regression Suite 🔶
File1_Suite1
TC_Perform Set Variable and Global Variable
TC_Global Variable_Validation
[346] TC D. 4 A
🔯 Test Command 📑 *Methods 🛛 🕼 Variable 🕂 — 🗖 🗖
Search
Methods
Method1
Method1_File2.twiz

TestingWhiz preserves both the files if they are having same name. User can decide to discard one based on code inside.



Person 1	Person 2	Action	Expected outcome
File data table 1	File data table 1	Merge	Create a new file with both data tables 1 and 2





Person1 DataTable



Before Import Data Table

Person2 Data Table



TestingWhiz preserves both the files if they are having same name. User can decide to discard one based on code inside.



Final Step

TestingWhiz allows integration with versioning tools like SVN. The merged scripts can be checked in to your SVN repository using TestingWhiz IDE itself. Below are the steps:

Setup SVN repository:

Seconfiguration		lest	inavi	
General Execution	SVN			
 Add-ons Cloud Execution Mobile Testing Report Recording Data Validation Rules Screen Mail Database FTP TestingWhiz Grid Report Repository Bug Tracking Tool Test Management Tool Risk Based Testing SVN Install/Update 	Alias test	URL https://192.19 https://192.19	User Name nsanghavi	Add Edit Remove
			Test Connec	tion Apply
	·			DK Cancel

Select the project and perform Check-in, Check-out, Show history or Revert operations.



Optionally, once the code is merged, use your existing Version Control System installed on your machine to check-in changes in SVN, GIT or some such and yes, do not forget to put comment as a reason for check-in.

Below are samples for SVN as a repository.

Sample Commit:

Recent messages	A new version of TortoiseSVN is available. Please go to http://tortoisesvn.net to get it.
Regression suite for testing Oracle, <u>RostgreSQL</u> , DB2 & Mon	database operations. Tests MySQL, MSSQL, ggQDB
Changes made (double-click on file for diff): Check: All None Non-versioned N	: /ersioned Added Deleted Modified Files Directories
Path	Extension Status
🔽 📄 [Regression]db.twiz	.twiz added
	G
٠ [ال	G



Sample Show Log:

y Messages	s, Paths, Auth	nors, Revisions, Bug-IDs, Dat	e, Date From: 7/24	/2017 -	To	p: 7/	24/2017	
Revision	Actions	Author	Date			Message		
9454	÷	nsanghavi@CYGNET	Monday, July 24, 201	7 3:08:36	PM	Regressio	n suite f	for t
<pre> Aegressi CostgreS </pre>	on suite f QL, DB2 &	III For testing database MongoDB	operations. Tests	MySQL, M	ISSQI	L, Oracle		
Regressid PostgreS(Path	on suite f QL, DB2 &	III For testing database MongoDB	operations. Tests	MySQL, M	(SSQ) Col	L, Oracle	, Revis	ion
 Regression PostgreS(Path /trunk/E 	on suite f QL, DB2 & Documents/F.	III For testing database MongoDB Testing/D. Test Scripts/Regri	operations. Tests ession/[Regression]db.twiz	MySQL, M Action Added	(SSQ) Coj	L, Oracle py from path	, Revis	ion
<pre> Regressi PostgreS(Path</pre>	on suite f QL, DB2 & Documents/F.	III For testing database MongoDB Testing/D. Test Scripts/Regri	operations. Tests ession/[Regression]db.twiz	MySQL, M Action Added	(SSQ) Coj	L, Oracle	Revis	ion
< Regressi PostgreS Path /trunk/[on suite f QL, DB2 & Documents/F.	III For testing database MongoDB Testing/D. Test Scripts/Regri	operations. Tests ession/[Regression]db.twiz	MySQL, M Action Added	Cop	L, Oracle	Revis	sion
<pre> Regressi Regressi PostgreS Path /trunk/E </pre>	on suite f QL, DB2 & Documents/F.	III For testing database MongoDB Testing/D. Test Scripts/Regri	operations. Tests ession/[Regression]db.twiz	MySQL, M Action Added	Co	L, Oracle	Revis	iion
<pre> Regressi PostgreS(Path //trunk/E /trunk/E /t</pre>	on suite f QL, DB2 & Documents/F.	III For testing database MongoDB Testing/D. Test Scripts/Regri n revision 9454 to revision 94	operations. Tests ession/[Regression]db.twiz III #54 - 1 revision(s) selected,	MySQL, M Action Added	Cop	L, Oracle py from path	Revis	ion
<pre> Acgressi CostgreS Path /trunk/C /trunk/C Acceleration Show on Stop on </pre>	on suite f QL, DB2 & Documents/F. vision(s), from y affected pa	III For testing database MongoDB Testing/D. Test Scripts/Regree m revision 9454 to revision 94 ths	operations. Tests ession/[Regression]db.twiz III 454 - 1 revision(s) selected,	MySQL, M Action Added	Col	L, Oracle py from path ged paths	, Revis	ion
	Don suite f QL, DB2 & Documents/F. Vision(s), from y affected pa copy/rename nerged revisio	III Eor testing database MongoDB Testing/D. Test Scripts/Regre m revision 9454 to revision 94 ths	operations. Tests ession/[Regression]db.twiz III #54 - 1 revision(s) selected,	MySQL, M Action Added showing 1 (Cop	L, Oracle	, Revis	ion

End