**WORK PRODUCT(ARTIFACT): DEVELOPER TEST**

The Developer Test validates a specific aspect of an implementation element.

**PURPOSE**

This artifact is used to evaluate whether an implementation element performs as specified.

**RELATIONSHIP**

|  |  |  |  |
| --- | --- | --- | --- |
| **Roles** | Responsible:   * [Developer](http://epf.eclipse.org/wikis/openup/process.openup.base/capabilitypatterns/developer_82EAB9C9.html?proc=_SuWj4dOPEdyqlogshP8l4g&path=_SuWj4dOPEdyqlogshP8l4g,_ieYHIVQwEd2i9JHp7xurvw,_51ewYdOPEdyqlogshP8l4g,_aSrtcdOOEdyqlogshP8l4g,_TV_k8dOKEdyqlogshP8l4g,_0j1oYKqeEeGLQ_Sv1UBWjg) | Modified By:   * [Developer](http://epf.eclipse.org/wikis/openup/process.openup.base/capabilitypatterns/developer_82EAB9C9.html?proc=_SuWj4dOPEdyqlogshP8l4g&path=_SuWj4dOPEdyqlogshP8l4g,_ieYHIVQwEd2i9JHp7xurvw,_51ewYdOPEdyqlogshP8l4g,_aSrtcdOOEdyqlogshP8l4g,_TV_k8dOKEdyqlogshP8l4g,_0j1oYKqeEeGLQ_Sv1UBWjg) | |
| **Input To** | Mandatory:   * [Run Developer Tests](http://epf.eclipse.org/wikis/openup/process.openup.base/capabilitypatterns/run_developer_tests_91CCE06A.html?proc=_SuWj4dOPEdyqlogshP8l4g&path=_SuWj4dOPEdyqlogshP8l4g,_ieYHIVQwEd2i9JHp7xurvw,_51ewYdOPEdyqlogshP8l4g,_aSrtcdOOEdyqlogshP8l4g,_TV_k8dOKEdyqlogshP8l4g,_iRZNQNOKEdyqlogshP8l4g) | Optional:   * [Implement Solution](http://epf.eclipse.org/wikis/openup/process.openup.base/capabilitypatterns/implement_solution_B7C117A5.html?proc=_SuWj4dOPEdyqlogshP8l4g&path=_SuWj4dOPEdyqlogshP8l4g,_ieYHIVQwEd2i9JHp7xurvw,_51ewYdOPEdyqlogshP8l4g,_aSrtcdOOEdyqlogshP8l4g,_TV_k8dOKEdyqlogshP8l4g,_hVe-oNOKEdyqlogshP8l4g) | External:   * None |
| **Output From** | * [Implement Developer Tests](http://epf.eclipse.org/wikis/openup/process.openup.base/capabilitypatterns/implement_developer_tests_6442995D.html?proc=_SuWj4dOPEdyqlogshP8l4g&path=_SuWj4dOPEdyqlogshP8l4g,_ieYHIVQwEd2i9JHp7xurvw,_51ewYdOPEdyqlogshP8l4g,_aSrtcdOOEdyqlogshP8l4g,_TV_k8dOKEdyqlogshP8l4g,_gXo2UNOKEdyqlogshP8l4g) | | |

**DESCRIPTION**

|  |  |
| --- | --- |
| **Main Description** | This artifact covers all of the steps to validate a specific aspect of an implementation element. For example, a test could ensure that the parameters of a method properly accept the uppermost and lowermost required values. A developer test specifies test entries, execution conditions, and expected results. These details are identified to evaluate a particular aspect of a scenario.  When you collect developer tests for a specific implementation element, you can validate that the element performs as specified.  The tests be self-documenting so that it is clear upon completion of the test whether the implementation element has run correctly. |
| **Brief Outline** | Although there is no predefined template for this work product, and testing tools affect how the work product is handled, you should address the following issues:   * Setup * Inputs * Script * Expected Results * Evaluation Criteria * Clean-Up |

**TAILORING**

|  |  |
| --- | --- |
| **Impact of not having** | If you do not run developer tests, you cannot ensure that elements that you modify over time are working. This can inhibit iterative development and maintenance. |
| **Reasons for not needing** | If you can embed the tests into the production code, you might not need a separate work product. Nonetheless, some level of support for developer testing is always necessary when you develop application software. |
| **Representation Options** | Suggestions and options for representing this work product:  **Suggestion: Automated code unit**  The most appropriate technique for running these tests is to use code that tests the implementation element scenarios and that you can run regularly as you update the system during development.  When code is the sole form of the tests, ensure that the code is self-documenting. The code should document the specifications of the conditions you are testing and the setup or clean-up that is required for the test to run properly.  **Option: Manual instructions**  In some cases, you can use manual instructions. For example, when testing a user interface, a developer might follow a script, explaining the implementation element. In this case, it is still valuable to create a test harness that goes straight to the user interface. That way, the developer can follow the script without having to follow a complicated set of instructions to find a particular screen or page.  **Option: Embedded code**  You can use certain technologies (such as Java(TM)5 Test Annotation) to embed tests in the implementation. In these cases, there will be a logical work product, but it will be assimilated into the code that you are testing. When you use this option, ensure that the code is self-documenting. |