

**Easy! Software LLC**

---

---

**Easy!Flow**  
**Process Definition**  
**White Paper**

---

<b>Author(s)</b>	<b>Easy! Software LLC</b>
<b>Version:</b>	<b>1.5</b>
<b>Status:</b>	<b>Released</b>

Easy! Software LLC  
**General Business Information**

This document contains information, which is under protection of copyright. Any distribution, revision and publication of this document are prohibited. The usage of this document must be under the authorization from Easy! Software LLC.

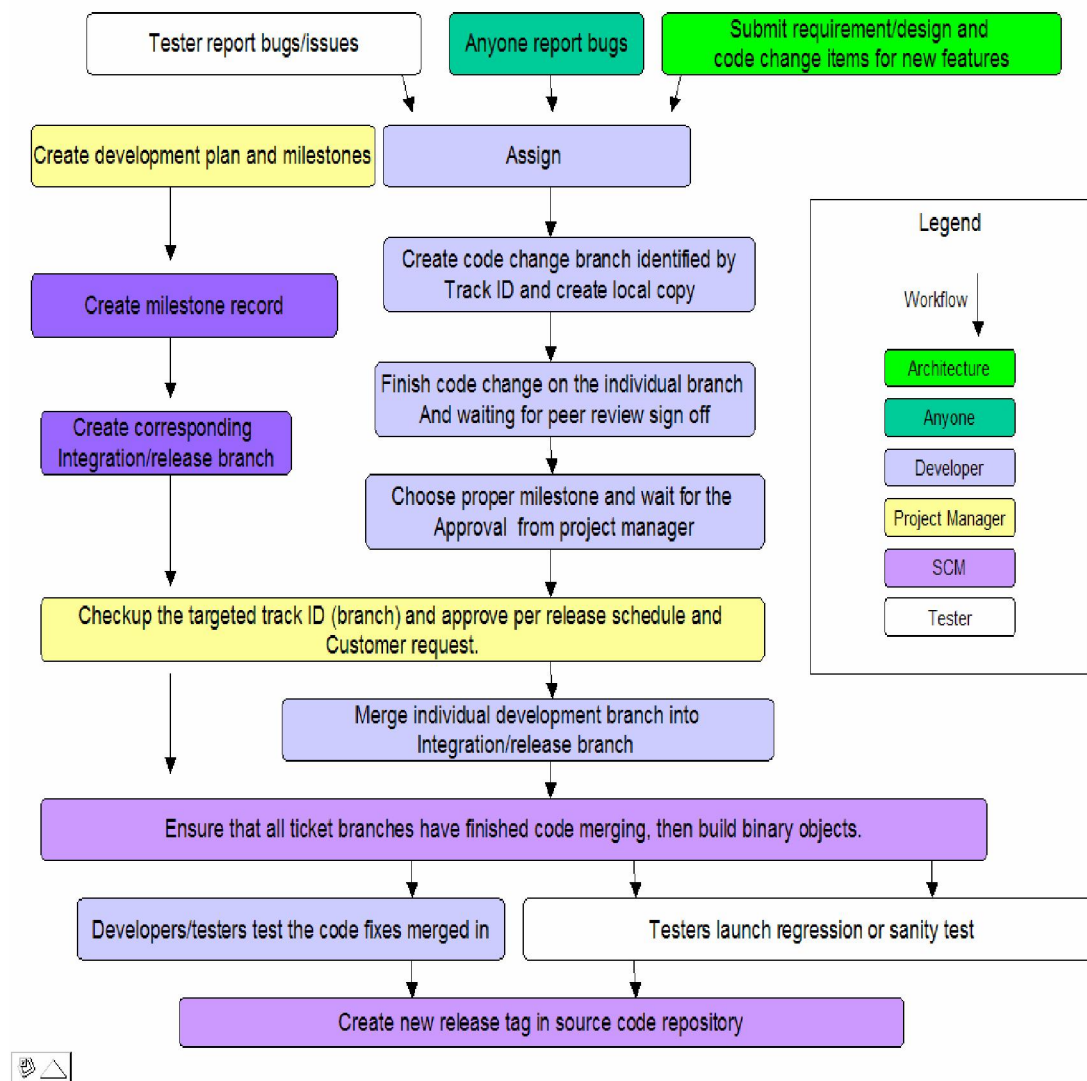
© **Easy! Software LLC. 2008**  
www.easysoftwarellc.com (US)  
www.easysoftwarellc.com.cn (China)

## Index

1. Process Overview .....	3
2. Streamlined workflow regarding different roles .....	4
3. Specification for General Metrics .....	8
Appendix A – Reference Links .....	11
Appendix B - CMMI PAs .....	12

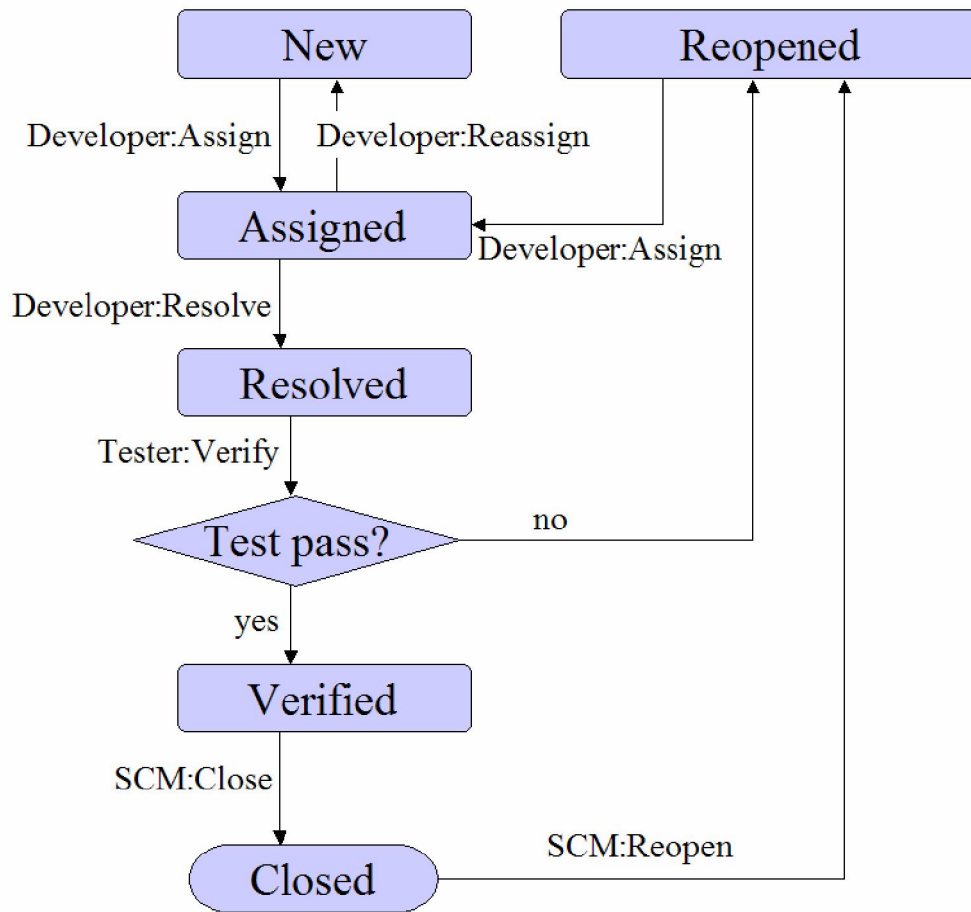
# 1. Process Overview

Collaborative process workflow is defined as below chart. Different area colors represent different roles. One person may have one or more roles during the execution of the process.



(Workflow Overview)

From the perspective of an individual ticket, below chart depicts the lifecycle statuses and transition activities.



(Ticket lifecycle)

## 2. Streamlined workflow regarding different roles

The whole collaborative process streamlines the work of each role as following details,

### 1 Workflow of architect role:

(1) Breakdown requirement items to component design and implementation items by documentation or using parent-child hierarchical tickets with corresponding types/subtypes.

(2) If you are using documentation method rather than parent-child tickets to manage your requirement and design, make sure that you have created all implementation tickets in Easy!Flow and marked them with tracking id of corresponding design or requirement items to maintain the traceability.

### **I Workflow for developer role:**

(1) If you have “canassign” permission and have the “assign task” responsibility, query for newly reported tickets, assign engineer for implementation, and mark original estimation of the working hours for the ticket (If you can’t estimate the working hours at this moment, leave it for the assignee to specify), specify whether review is required, finally change the ticket status as “assigned”. For those developers who have no “canassign” permission, skip this step.

(Note: once a ticket is changed to “assigned” status and original estimation is set to a non-zero number, original estimation will be readonly for non-admin users.)

(2) Query for the “assigned” tickets waiting on you, create related ticket development branch, get local copy, do your code change and commit from your own local copy to the ticket branch.

(Note: if you’d like to use one branch to solve more than one ticket, just link other tickets as the child of your main ticket. Please mark the subtype of those “other tickets” as “NA” so that the status of those tickets can be

changed to resolved & fixed without code merge.)

(3) If review is required for this ticket, create peer review for the code change and invite proper reviewer(s) and moderator to finish peer review for your code change. Do rework per moderator's request if any issue found during the execution of peer review.

(4) Target you ticket to proper milestone. If "punchlist approval" is required, mark "Punchlist\_Approved" flag as "?" and select a proper Project Manager as "Requestee" for this flag, so that Project Manager will know that you are applying to merge code into this milestone.

(5) Merge your code change into the integration branch of the target milestone.

(Note: a ticket branch can be merged multiple times into a target integration branch.)

(6) Once you consider the ticket work is done, mark ticket status as "Resolved" with solution "Fixed" (for the merged fixes).

(7) If review is enabled for current product, query for the open reviews waiting on you, submit comments and finish the reviews assigned to you.

## **I Workflow for tester role**

(1) If you have "canassign" permission, query for tickets targeted to current milestone in the punchlist query page, assign a tester as "QA contact" for the ticket.

(2) Query the tickets assigned to you as "QA Contact" and test the

bug/functionality referred by the ticket.

(3) If test passed, mark the ticket status as “VERIFIED”. Otherwise, fills in the reason of test failure and reopen the ticket so that the ticket owner will know that.

### **I Workflow for SCM and release coordinator role**

(1) With admin permission. Create milestones (if milestones are not created by project manager) and related integration branch, and then call for integration merge.

(2) Query punchlist to monitor integration process. After all the code changes have been merged, launch integration build to generate the binary deliverables for testers to launch test.

(3) If any issue found during the integration test, coordinate the ticket owner to fix and merge again.

(4) After all the tickets on the release punchlist passed test, make a tag for the integration branch so that later development can base on it.

(5) Close all tickets targeted to the milestone and close the milestone.

### **I Workflow for review moderator role**

(1) If review is enabled for current product, query for the open reviews waiting on you as the moderator, go through the review comments and make sure that the code change owner has fixed all the issues referred in the reviewers’ comments.

(2) If no major issues found or the code change owner has fixed the issues

found during the review, close the review.

## **I Workflow for Project Manager**

(1) Base on the project plan, create milestones and target project/feature tickets to proper milestones (the project/feature tickets are created by architecture engineers).

(2) If “punchlist approval” is enabled for current product, query for the punchlist status of current milestone, and approve ticket to merge into this milestone by marking “Punchlist\_Approved” flag as “+” or reject by marking it as “-“.

(3) Use time tracking to monitor the progress of development works on different milestones.

(4) Use metrics to monitor process execution and get management info per organization or department business goals.

## **3. Specification for General Metrics**

**I Business Goal 1 (for cost):** Analyze the effort of individuals to make the management have a better understanding of resource allocation and the performance of team members.

### **Related Metrics:**

(1) The bugs with different severity levels found by individual testers within recent month (quarter or year).

- (2) The bugs/features with different severity levels resolved by individual developers through actual code change within recent month (quarter or year).
- (3) The bugs/features with different severity levels tested (verified) by individual testers within recent month (quarter or year).
- (4) Hour based working progress for different milestones regarding individual developers.

**I Business Goal 2 (for quality):** Analyze the defect and quality effort distribution to let management know where is the quality bottleneck and how much quality effort is allocated to it.

**Related Metrics:**

- (1) The distribution of reported bugs per component/product basis.
- (2) The distribution of reported bugs with high severity in different components.
- (3) How many bugs at high severity levels have performed or not performed code peer review.

**I Business Goal 3 (for schedule):** Analyze the development progress status to let the management identify schedule risk or divergence between plan and reality.

**Related Metrics:**

- (1) The on-plan and delayed tasks on component and product basis.
- (2) The on-plan and delayed tasks on milestone basis.

**I Business Goal 4 (For the showstoppers):** Highlight the showstoppers to let management know currently critical issues.

**Related Metrics:**

- (1) All open and critical bugs reported by tester on different component of different products.
- (2) All critical requirement or feature tasks on going.

## Appendix A – Reference Links

Document library:

Easy!Software LLC US: [www.easysoftwarellc.com/documentation.html](http://www.easysoftwarellc.com/documentation.html)

Easy!Software LLC China: <http://www.easysoftwarellc.com.cn/documentation.html>

## Appendix B - CMMI PAs

<i>Name</i>	<i>Abbr</i>	<i>ML</i>	<i>CL1</i>	<i>CL2</i>	<i>CL3</i>	<i>CL4</i>	<i>CL5</i>
Requirements Management	REQM	2	<b>Target Profile 2</b>				
Project Planning	PP	2					
Project Monitoring and Control	PMC	2					
Supplier Agreement Management	SAM	2					
Measurement and Analysis	MA	2					
Process and Product Quality Assurance	PPQA	2					
Configuration Management	CM	2					
Requirements Development	RD	3	<b>Target Profile 3</b>				
Technical Solution	TS	3					
Product Integration	PI	3					
Verification	VER	3					
Validation	VAL	3					
Organizational Process Focus	OPF	3					
Organizational Process Definition +IPPD	OPD +IPPD	3					
Organizational Training	OT	3					
Integrated Project Management +IPPD	IPM +IPPD	3					
Risk Management	RSKM	3					
Decision Analysis and Resolution	DAR	3					
Organizational Process Performance	OPP	4		<b>Target Profile 4</b>			
Quantitative Project Management	QPM	4					
Organizational Innovation and Deployment	OID	5	<b>Target</b>				

<i>Name</i>	<i>Abbr</i>	<i>ML</i>	<i>CL1</i>	<i>CL2</i>	<i>CL3</i>	<i>CL4</i>	<i>CL5</i>
Causal Analysis and Resolution	CAR	5	<b>Profile 5</b>				