The goal of this process is to produce a more robust system that better supports the business activities of our customers. The process includes:

- **Collect** change requests (a generic term that encompasses defects, bugs, requests, suggestions). This is a continuous process.
- **Identify** the scope of the next release and determine which change requests will be included in the next build. The change requests should be grouped according to the functional and/or physical area of the system the request will change.
- **Document** the requirements, functional specification and implementation plans for each grouping of change requests.
- **Implement** the change. This may involve changes to source code, database structure, or user documentation.
- **Test** or verify the change:
  - **Unit testing** is done by the person who made the change, usually the programmer. It is very specific to just the area changed.
  - **Function testing** tests a functional area of the system to see that everything works as expected.
  - **Regression testing** is system-wide to insure that all areas of the system still function as expected. This validates that the change caused no unexpected side effects and that the system still has the overall functionality it had before the change. Regression testing could consist of testing all functional areas of the system.
  - **User acceptance testing** is done by a select group of end users, usually run in parallel to the existing system. This is the final test before the system “goes live.”
- **Release** the system to production. This is the physical process of upgrading the production system to the new release. This may involve executable files, database changes, data migration, and the installation of new support software.

### Roles in the Change Request Process

The roles currently identified are:

- Change Request Creator
- Change Request Coordinator
- Change Request Reviewer
- Analyst
- Developer
- Tester

Since each person in a company may have more than one role, it is important to know where that role acts in the process, what actions must be performed in each role, and how to use the software to accomplish that role. (See Table 1)

Each role has certain choices. Depending on the action chosen, different values need to be set for the process to continue to flow smoothly. (See Diagram 1)

### Change Request Creator

The creator of a change request is the person who enters the change request into the system. They are responsible for the entry of the following fields.

**Summary**: A short description of the change request is required.

**Type**: This field is required and indicates the category, from “Crash – Data Loss” to “Feature Request.”

**Product**: The executable that the change request will affect.

**Entered by**: Automatically set to the logged-in user.

**Priority**: Assigned by the Change Request Coordinator, but can be entered here as a guideline to the importance of the change request from the user’s perspective.

**Component**: The sub-system of the product.

**Severity**: Indicates the seriousness of the issue.

**On the Detail Tab**

**Found by**: This field is usually the person who found the issue.
Since this person will be testing this item when it is finished, the ‘found by’ person should be very knowledgeable.

**Description:** This field should contain a detailed account of the issue.

**Steps to reproduce:** The reproduced field set to an appropriate value and any steps needed to reproduce the issue should be documented.

**Workaround:** A place to document a workaround, if one has been found.

**Attachments:** A place to attach any supporting documents. Generally this is a Word or rich text document into which text, output, screenshots, etc. would be placed to illustrate both the behavior that exists and the desired change. Documents of any type may be attached.

**Assign Defect:** All change requests are automatically assigned to the Change Request Review Committee when they are created. Each member is notified by e-mail every time a new change request is entered. The exception is if the priority is IMMEDIATE. Development receives notification of any IMMEDIATE priority change requests so that these issues can be resolved quickly.

**Change Request Review Committee**

When a change request is created in TestTrack Pro, an e-mail is automatically sent to each member. The e-mail will contain the summary and detail description but not the attachments. In addition, the notification will start with this text.

"X" the action you recommend and reply
- [ ] Fix immediately in production
- [ ] Fix in upcoming release
- [ ] I need more information (Please specify what you need)
- [ ] Reject this change request

It is the responsibility of each person to review and respond to each change request. Often it will be necessary to view the change request and examine the steps to reproduce, the workaround, any attached documents, and discuss the consequences before responding. Once you have reached your decision, simply "X" your choice and reply. If a consensus is not reached, the Change Request Review Committee will determine the action to be taken when it next meets.

---

**Table 1: Roles**

<table>
<thead>
<tr>
<th>Role</th>
<th>CR Liaison</th>
<th>User</th>
<th>Developer</th>
<th>CIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Creator</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CR Reviewer</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR Coordinator</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyst</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Tester</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram 1: Change Process**

**Change Request Coordinator**

The Change Request Coordinator applies the action determined by the committee in the following ways:

Set the **Priority** as determined by the committee.

1. If this needs to be fixed immediately in production:
   a. Assign to a developer.
   b. Set **version** to “Current Production.”
   c. Set **disposition** to “OK to implement.”
2. If this is to be fixed in upcoming release:
   a. If the version assigned is currently in work:
      i. Assign to a developer.
      ii. Set **disposition** to “OK to implement.”
b. If a future version:
   i. Assign to “Pending.”
   ii. Set disposition to “Fix in future release.”

3. If more information is needed:
   a. Assign to a resource to do the investigation
      (usually the change request creator).
   b. Set disposition to “Need Customer Input.”

4. If this change request has been rejected:
   a. Close.
   b. Set disposition to “Rejected.”

When a new version is being planned, all the pending change requests should be reviewed to see what will be included in this new release. The change requests that are selected should have their disposition changed to “OK to implement” and Assign to a developer.

**Analyst**

The analyst is assigned to review/research issues to determine the impact of the requested change. They will recommend if the change should be made and if so, ways to implement that change.

When the information has been gathered, the analyst should do the following:

1. Attach their finding as a document or at least a note in the description.
2. Assign the change request to the user named REVIEW.
3. Change the disposition to “Open—Not Reviewed.”

**Developer**

The developer should only work on change requests that are assigned to them and have a disposition of “OK to implement.”

The version should either be “Current Production” or the current version in work. If all of these conditions are not met, check with your supervisor to resolve the issue BEFORE starting the work.

The developer will plan and implement the changes and then do initial testing.

They will:

- Add comments and release notes as necessary.
- Mark it fixed. This involves selecting either “Fixed, needs verification” (the more common) or “Fixed and Close.” The version, resolution, notes and other fields should be filled out.

The change request will automatically be assigned to the change request creator for testing.

**Tester**

The tester will verify that the change performed as expected with no side effects.

If it passes, it should be verified as “Pass and Close.” Fill in version and notes fields.

If it fails the tests, verify it as “Failed.” The tester will need to manually assign it to whoever fixed this item. This can be determined by looking in the workflow tab.

**Defect Tracking**

The defect tracking software TestTrack Pro is used to:

1. Enter change requests (defects or enhancements) in one central location.
2. Allow supporting documentation to be collected.
3. Track change requests from detection through resolution.

**TestTrack Pro**

The defect tracking software used is TestTrack Pro and SoloSubmit from Seapine Software. TestTrack is the client software that will be used by development personnel (programmers, QA, analysts, and managers), to modify change requests and move them from detection to resolution. SoloSubmit is a browser-based GUI that allows users to enter new change requests.

**Project Setup**

TestTrack allows much customization. The following are settings that are made by a user from the Administrator group and affect all users.

**User Groups**

In addition to the groups provided, the following users groups were added:

- **Group Name**: Reviewers
- **Description**: Members review and prioritize change requests
- **Users in Pop-up**: Yes

**Users**

Users were defined for special purposes. Only the “Admin” user was retained from the original database. (See Table 2)

**Database Options**

From the main menu select Edit, then Options, then Database Options.

**General Tab**

Where the project name and description are specified.
**Defects Tab**
Deferred numbering of defects may be enabled. This allows defects to be entered and then assigned a number during the initial verification phase.

The next defect number and the increment between numbers can be specified.

Version numbering of either free text or restricted to a list can be specified. Set to restrict to the list.

Logging of history should be enabled.

**Send Mail Tab**
Enable sending of e-mail. Use defect-tracking administrator for return address.

**Workflow Tab**
Leave all unchecked. Enabling these options adds extra states to the process. These may be needed later.

**Assign Tab**
The following are changes to the defaults:

“Defect created” set to “Assign to following user.”
That user is “Review.”

The Review user will have an e-mail address that is actually a group. All reviewers will be a member of that group and will get a notification every time a defect is created.

“Defect fixed” set to “Assign to user who entered it.”

“Defect is re-opened” set to “Assign to following user.”
That user is “Review.”

**Import Mail Tab**
Everything disabled. We are not using the e-mail import feature at this time.

**SoloSubmit Tab**
Check “Enable entering defects via SoloSubmit web page.”

Set “SoloSubmit HTML page” to “SoloSubmit.html.”

Check “E-Mail an acknowledgement to the submitter.”

Check “Enter the next sequence number in the defect's reference field.”

**Activities**
From the main menu select Activities, then Configure, then the list you wish to modify.

The entries for each of these lists will need to be customized for the specific company needs.

**Defect Types**
- Crash – Data Loss
- Incorrect Functionality
- Cosmetic
- Investigate
- Crash – No Data Loss
- Erroneous Data
- Feature Request

**Defect Resolutions**
- Code Change
- Documentation Change
- Not Our Bug
- On Hold
- Database Change
- Clarification
- Not a Bug

**Defect Priorities**
- Immediate (use ONLY for problems that must be fixed immediately)
- High
- Low
- Medium
- Future Release

**Defect Seversities**
- Causes Crash
- Workaround
- No Workaround
- Cosmetic

**Component Names**
- Directory
- Reports
- Registration
- Setup

**Reproducible Levels**
- Always
- Rarely
- No Attempt Made
- Sometimes
- Could Not

**Disposition Names**
- Open – Not Reviewed
- Need Customer Input
- Fix In Future Release
- Rejected
- Open – Reviewed
- OK to Implement
- Hold

---

**Table 2: Defined Users**

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>User Name</th>
<th>Password</th>
<th>User Group</th>
<th>E-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
<td>Pending</td>
<td>HCCBCB</td>
<td></td>
<td>Reviewers</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Engineers/Developers</td>
<td>N/A</td>
</tr>
</tbody>
</table>

---
**Version Names**
- Current Production  •  Ver 1.0
- Ver 1.1  •  Ver 2.0

**E-Mail Message Templates**
This allows the various e-mail messages to be customized.

The word defect had been replaced by change request, a more generic term encompassing both defects and changes in functionality.

The defect added message has been modified by adding the following text. The added text is to be used by each Change Request Reviewer to respond to each change request in a more automatic manner.

"X" the action you recommend and reply

- [ ] Fix immediately in production
- [ ] Fix in upcoming release
- [ ] I need more information (Please specify what you need)
- [ ] Reject this change request

**Configure Custom Fields**
This allows user-defined fields to be created and modified.

No custom fields at this time.

**Required Fields and Default Fields**
This area has two functions: 1) selection of required fields and 2) the setting of default values. The fields are divided into several areas: general fields, found by fields, assigned fields, estimated fields, fixed fields, release to test fields, verified fields, release to customer test fields, customer verified fields, closed fields, reopen fields, release note field, comment fields.

These settings need to be recorded.

If the **Required** column is empty, the value is NO. If **Default** column is empty, the value is NOT SET. (See Below)

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Required</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Fields</td>
<td>Summary Type</td>
<td>Yes</td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Priority</td>
<td>Yes</td>
<td>Open-Not Reviewed</td>
</tr>
<tr>
<td></td>
<td>Disposition</td>
<td></td>
<td>Processing Assistant</td>
</tr>
<tr>
<td></td>
<td>Product</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Component Reference</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td>Found By Fields</td>
<td>Found By Version Found</td>
<td>Yes</td>
<td>Current user</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Reproduced</td>
<td></td>
<td>No Attempt Made</td>
</tr>
<tr>
<td></td>
<td>Steps To Reproduce</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Computer Config</td>
<td></td>
<td>User's Test Config</td>
</tr>
<tr>
<td></td>
<td>Other Hardware/Software</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td>Assigned Fields</td>
<td>Assigned To</td>
<td>Yes</td>
<td>Current user</td>
</tr>
<tr>
<td></td>
<td>Assigned By</td>
<td>Yes</td>
<td>Current date</td>
</tr>
<tr>
<td></td>
<td>Date Assigned</td>
<td>Yes</td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Assigned Notes</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td>Estimated Fields</td>
<td>Estimated By</td>
<td>Yes</td>
<td>Current user</td>
</tr>
<tr>
<td></td>
<td>Estimated Date</td>
<td>Yes</td>
<td>Current date</td>
</tr>
<tr>
<td></td>
<td>Estimated Effort</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Completion Date</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Estimation Version</td>
<td></td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>Estimation Notes</td>
<td></td>
<td>Blank</td>
</tr>
</tbody>
</table>

**About the Author**
Garth Wilcox is a software manager. He can be contacted at garthw@mindspring.com.