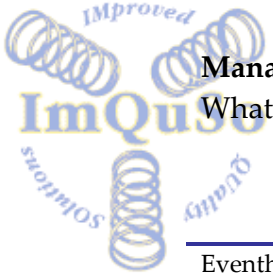
Eventhandling



Management Summary:
What sort of eventhandler do you need.

TABLE OF CONTENTS

1 Introduction

Software becomes more complex, larger in size, built by bigger teams sometimes on different geographical locations. This all has the unwanted side effect that more bugs and change requests are produced.

In the old days small notes were sufficient to handle all problems and changes. Today software can have thousands of bugs and probably the same amount of requested changes that all need proper management.

Developers, Testers, Analysers, etc. must have access to the bugs, project leaders, controllers need to make overviews to determine the state of the bugs and track changes.
But most of all, these bugs and changes need to be controlled so that only the things will be fixed that where planned to be fixed.

In their striving for excellence most organisations will adapt to one of the major quality models, most of these models will enforce the organisation to do defect/change tracking.
This and the practical need (as described earlier) to control these disturbances, will lead to the statement that all organisations eventually want to track the progress of their changes and problems.

2 Purpose

This document is not a detailed specification for a defect tracking system. Its main purpose is to provide the reader with all kinds of subjects that need to be reviewed when defining a specification for such a defect tracking system.

The reader can use this document, as a short of checklist and it will give some solution for common problems.

Field, report, graph, measurement names used are just for demonstration purposes

3 General

3.1 Organisational constraints

Tracking should be a visualisation of the control you have on the progress of a Problem Solution, Change implementation, etc. So the level control you want to have over the progress will have a major impact in the way you track and manage this progress. The questions you need to pose are:

- You want to adapt to a model. E.g. ISO9000, CMM, CMMI-SW, ITIL, etc.
  Most models have defined authority for change control.
- You have a small or large organisation/project.
  Large organisations need more communication so they need more management and control.
- You expect few of a lot of problems, changes, etc.
  A project that expect 10 problems to be manages typically don't want to implement a 30 pages long change control procedure.

This document describes the situation where you have formal control.
It is easy to extend this model with even stricter control or with fewer control points it is up to you and your organisation how you want to implement this.

3.2 Problem and Change definition

Most organisation recognise that they want to track their problems and their changes (or whatever name they want to give to them, bug-fix, Specification-change, Field-error, etc.). There are few organisations that explicitly make NO distinction between these two.

To make the distinction between these two I would like to use the next descriptions.

- Change Request
  Results in modification of a/any specification/requirements document
- Problem Report
  A none compliance to a specification/requirement.
  This implies that none of the specification/requirements documents need to be modified.

This document will show that if we ignore the grammatical differences and dependencies, "changes are implemented" and "problems are resolved" etc. we can treat all of them in the same manner.

We can even go further and think of other things that we want to track within the same concept.

- Actions
  These are small unplanned investigations, feasibility studies, activities as a result of e.g. inspections.
  If you read the remainder of the document you will discover similarities, which make the suggestion of tracking these actions with the same concept, look feasible.

- Open Issues
  Things that need to be resolved in the organisation / project
  (Can be seen as feasibility studies, etc)

To make the rest of the document easier to read the Problem Reports, Change Requests and Actions, etc. are defined as a single group called: Events.

Note on: intuitive behaviour
If someone writes a Problem Report that a product is not reacting or responding in an intuitive way then this PR should be converted into a Change Request because there is probably no requirement that defines that behaviour.

PR's on intuitive behaviour are generally created because there is a missing gap in the requirements. And as we all know: Requirements and specifications should be clear and unambiguous. So why not converted this PR into a CR, and defined the behaviour as required or as unwanted.
Do not forget to update your documents or another person will raise this issue with a PR again!

3.3 Terms and Abbreviations

To Be Defined

4 The Event Life Cycle

In the event handler system rules apply that define the behaviour of the system. For me the lifecycle is the best entrance for a description start of such an event handler system.
Using the life cycle we define:

- The stages an event passed before it reaches an end situation (State Names).
- The authority a person must have to move the event to a next state (Authority).
- The possible state transitions that are available (Transitions).
  When some has to be informed that something has to be done. (Notification).
- The things that have to be done
  like e.g. Action to do and data fields to be filled.

4.1 The roles

In the event handler somebody has to do something at a certain point. This can be an action of make a decision.
What are possible roles in the system?

- Someone who creates new events
  Let's call this role: Submitter
- Someone who can do an impact investigation and analysis regarding the location of the problem, possible solution and effort estimation.
  Let's call this role: Analyser (Investigator could also.)
- A person who make the actual changes to solve the problem.
  Let's call this role: Solver
- Someone who verifies the implemented solution.
  Lets call this role: Tester
- A person with the responsibility and authority to manage the changes.
  Most of the times this is a group of 'wise' man with a chairman called: Change Control Board. See § 4.2 The Change Control Board
  There can also be a 'Secretary' who on behalf of the CCB does the changes in the event handler.
- Someone who maintains the event handler on IT infrastructure level.
  Most of the time this person is also responsible for tailoring the default installation to the organizational needs.
  This is a role mostly enforced by the event handler tool called: administrator

4.2 The Change Control Board

At a certain point someone has to make a decision on what to do with all events, e.g. if we are going to solve them? For what release should this be done?, Who has to do it?, etc.

In most organisations the Change Control Board manages the events. In some organisation the CCB is also authorises the release of products.
A CCB typically consists of:

- The project leader (chairman)
- The Configuration Manager (secretary)
- The architect (vice Chairman, vice secretary)
- The quality assurance officer.
- Representative of the Quality Department.

Note:

- The Quality Assurance Officer
  Most organisations have a software quality/development manual that describes the processes and procedures that software development projects are obligated to use in the organisation. In these organisation there is most of the times a 'Quality Assurance Officer' that is assigned by the management of the organisation to a project and acts there as an

independent observer and verifier to guard the implementation of the quality manual procedures, templates, methods, etc.

- The Quality Department
  Some organisations also have an independent Quality Department that tests the quality of the products that this organisation delivers. A QD member is part of the CCB so that QD knows when the products are ready for testing and what is fixed/modified in the new release.

4.3 The state names

An event will pass a number of stages in its life cycle before it is dealt with.

A minimum life cycle could be: from Known to Resolved or from New to Resolving to Closed

As you see from the examples the state names should be clear and preferably short.
State names should not be nouns but verbs. Nouns reflect the activities between the states so do not use one state "Resolving" but two states "Open " and "Closed"

The state names in our lifecycle are:

State Name	Description
New	Newly entered in the system.
Analysis	Impact analysis has to be done.
Analysed	An analysis has been done.
Open	Someone has responsibility to solve the event.
Solved	A solution is implemented.
Test	The implemented solution will be tested.
Verified	The implemented solution has passed the defined tests.
Closed	The Event is implemented in some form.
Rejected	Event will not be implemented for a specific reason.
Duplicate	A 'similar' event already exists in the registration system.
Decision	For some reason 'someone' needs to make a decision on what to do.
Hold	No decision taken yet.

4.4 State Transition Authorities

If management wants to control the flow in the lifecycle then not everybody should be allowed to make state transitions, so based on the roles we specify who have authority and state transition permissions.

Role	From State	To State	Comment
Submitter	-	New	An event is entered in the system.
Analyser	Analysis	Analysed	An impact analysis has been done with a proposed solution and an estimate for the fix & test effort.
Solver	Open	Solved	Problem fixed as described in the analysis.
		Decision	For some reason the problem could not be fixed as described in the analysis.
Tester	Test	Verified	Test results of the fix are positive.
		Decision	The fix failed at least one test.
CCB	Decision	Analysis Open Test	Allocate a role/person to analyse the event. Allocate a person to fix the event. Allocate a person to test the solution.
		Duplicate	The event is an copy form an already existing event or has the same cause.
		Rejected	Decision that this event will not be implemented for a specific reason.
		Hold	The CCB doesn't know (yet) what to do with this event.
	Analysed	Decision	The analysis has been done and the CBB has to decide what to do with this event.
	Solved	Test	The problem is solved and can be tested.
	Verified	Closed	The fix passed the tests and can be closed.
	Hold	Decision	The CCB knows how to handle this event.

With this authorisation scheme any non CCB role can only transition to two states:

1. A positive outcome or
2. A negative outcome.

These states are under control of the CBB. This implies that the CCB controls the flow during the entire lifecycle.

Efficiency fixes for the lifecycle:

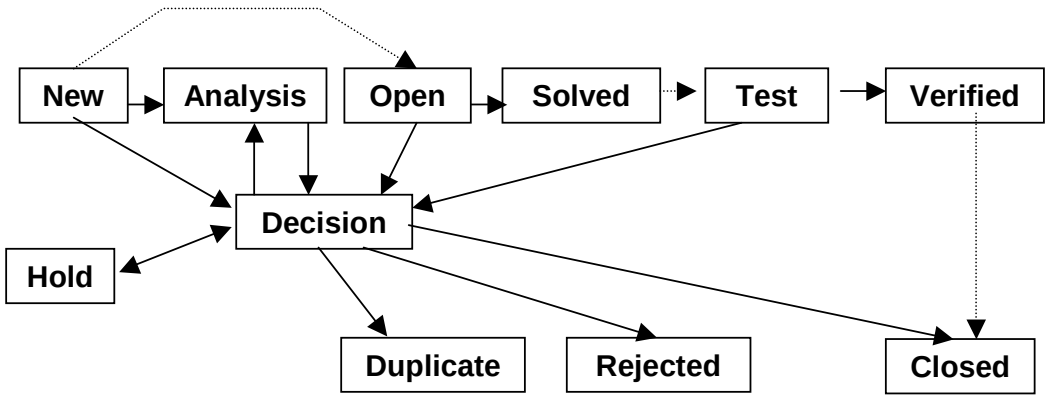
- The transitions:
  - From: Analysis to Decision and
  - From Analysis to Analysed to Decision

  can be combined into a singe transition:
  - From Analysis to Decision

  without authorization violation or information loss.
- The CCB has authority on all events as such it also has the authority to skip some parts of the lifecycle if needed.
  e.g. From New to Open if in the CCB can do the analysis.

4.5 Life Cycle flow chart

From → To



4.6 Notifications

In a normal world you will be told that you have to do something, in event handling this is the same. If you want someone to take action you have tell him. In an IT organization, email is the ideal medium to send notifications.
The common notifications are:

- The analyser in the Analysis state
- The solver in the Open state
- The tester in the Test state

The CCB usually uses reports to know which events are in what state and preferable will not be notified via email. Otherwise it will be a lot of Emails send to the CCB who loses oversight by the huge pile of emails

Other notifications that are possible in some tools:

- Notify now [Who has to be notified for this specific state change]
  People who only need to receive a notification of this transition are selectable from a dropdown list by the person who does the state transition
- Notify always [Who has to be notified on all state changes]
  This notification list is default set for every new event. Mostly the 'administrator' can modify the list on request.

5 The Data fields

5.1 Typical/Minimal data fields

Every change management system should have at least the following information fields:

- Unique identification.
  Mostly a number that refers only to this event.
- Title/synopsis.
  Short one line description of the event.
- Description.
  Long description on the event and how to reproduce it or why it is needed.
- Date on which the event is entered in the system.
  Used to decide if it is still relevant and for throughput calculations.
- Date when event reaches an end state.
  To see that the event is closed and for throughput calculations.
- Submitter.
  Who entered the event in the system and can a person we can ask for additional information if needed.

5.2 Extra data fields

A selection of the next data fields can be used in your organization based on you needs and willingness to log additional information.

5.2.1 Discovered in product

In what product did an event occur.

- Product identification:
  - Product.
    In which product was the event discovered.
  - Version.
    Which version of the product was the event discovered in.
    (Preferably this list of version numbers typical for the product selected)
  - Sub-system.
    In which sub-system of the product was the event discovered.
    (Preferably this list of sub-systems names is typical for the product and/or product-version-number combination selected)
  - Module.
    In which the sub-system module of the product was the event discovered.
    (Typically …)

5.2.2 Solved in product

In what product was the event solved.

- Product identification (Can be the same list as; 'Discovered in' product)
  - Product
  - Version
  - Sub-system
  - Module

The list of solved-in-products will be longer as discovered-in-product as new products / releases are continuously developed. As it will be difficult enough to maintain such (nested) list. It is suggest to keep use the same list for Discovered-in-product and Solved-in product for ease of maintenance.

5.2.3 Activity logging

To be able to store a 'textual 'description of what has been done or why a decision is made log/description fields could be introduced. Mostly these fields are a free format text fields that can handle a lot of data.
This can be one field where data is appended or a number of dedicated fields for specific purposes like:

- Analysis comment
  Description on the cause of the event and possible 'concept' solutions.
- Resolution comment
  Description on what has been done to solve this event.
- Evaluation comment
  Description evaluation result, what has been done and optionally why the result was negative.

5.2.4 Allocation of people / roles

To be able to do actual work on an event most systems have assignment options. This can be role or person based (or a combination of both?).

- Role based assignments.
  When you have logged-in to the system your (pre-defined) role is known and you are permitted to handle any event that your role is permitted to.
- Person based assignment.
  A specific task for an event is allocated to you and you are the only one that is allowed to do that action.

Most system have rules that allow superior roles e.g. administrator to do these activities as well for maintenance or other reasons.

The following type of allocations is common in most systems:

- Analysis allocation.
  A role (e.g. architects) or person is responsible for the elementary analysis of an event and give a suggestion on e.g.
  - IS this a valid event.
  - How to proceed with this event

  - A possible concept solution
  - An estimate of the time it will take to resolve this, etc.
- Resolver allocation.
  Mostly a person to resolve the event.
- Tester.
  A role or person is responsible for doing a specific test.

5.2.5 History/trace logging for an event

Most systems have a trace of history logging built in in which they log any change to the event like e..g status change, assignments, adding of information, etc.
A history log could look like:

1999-11-14	Created	
1999-11-15	kerkhofp	New to Analysis
		Assigned kerhofp for analysis
1999-11-17	kerkhofp	Modified Analysis comment
1999-11-17	kerkhofp	Change state to Analysed

5.2.6 Submitter identification

For feedback and additional information reasons, the identification of the person who entered the event needs to be stored.

- Personid
  Who entered this event in the system.
- Etc.
  Any other submitter information

The above formation should be acquired by the system from existing tables like, password files, access table, etc.

Sometimes and event is entered in the system on behalf of another person who has (for some reason) no access to the database. Common “Original submitter” fields are:

- Name
  Who requested this.
- Email Address
  Email address for electronic responses.
- Company name
  Name of the company of the submitter.
- Etc.
  Any other information

5.2.7 To support event handling priority

The next data fields are used by organisations to set the priorities for resolving the events.

Severity of the event.

What is the impact of this event on the product/project

- S No conformity with safety standards or safety requirements.
- A Results in a not producible / usable products.
- B Results in a product, which can be produced with big problems or not be accepted by a critical customer.
- C Results in a product, which can be produced or sold with minor difficulties.
- D Results in a product, which can be produced or sold and only critical customer will complain.

Priority/urgency.

How quick must this be handled?

- Immediately Now: drop all work (Personal contact needed to ensure it.)
- High As Soon As Possible
- Medium As soon as work permits.
- Low At your convenience.
- None If there is no other work to be done.

5.2.8 Due dates and actual completion dates

A due date indicates when an activity should be done.
This can be one field to be reused for each state/activity but better is to specify a specific field for defined states/activities. E.g.

- Analysis due date
- Resolve due date
- Verification due date

As you have a due date it is also preferable to have an actual completion date to see if the target is met. In some systems this date can be filled by the system when a transition/activity is completed. But a manual correction of that date is preferred as the registration of the transition in the system can be delayed due to 'administrative' reasons.

When doing a complete reassignment of the problem e.g. got from evaluated to analysis and all previous due dates and actual dates will be overwritten as the event goes thru the same lifecycle again. But from the history/trace log of the event one could deduct what has happened and retrieve the initial values.

5.2.9 To Support development analysis

The next data fields are used in mature organisation where detailed analysis of the data is done to get a better development process quality and to support the estimates/planning of new projects.

Type of error that caused this event.

See: § 10.2 Product Maturity based on the type of errors discovered
E.g. Timing error, Design error, Coding error, etc.

How was the event discovered:

What kind of activity lead to the discovery of this event?
This list is highly dependable on the Verification and Validation activities that are in place in your organization.
The activities can be:

- By functional test
  User test on the entire product (based on the CRS)
- By system test
  User test on the entire product (based on the FRS)
- By Beta test
  ??
- By Alpha test
  Test by the software team before delivery.
- By integration tests
  Test executing during the integration of software / hardware modules.
- By regression tests
  Sequence of selected tests to be repeated several times (preferably automated).
- By random unplanned tests.
- By in house normal use.
- By review
  Action point that was raised during a review.
- By Code review
  Action point that was raised during a review.
- By static code analysis
- By customer demand
- By customer use

In which phase was the event discovered.

In which development phase was the problem discovered (Strongly depends on the development life cycle used)

- Initialisation (Period of project assignment)

- Conceptualisation (Period of architecture decisions)
- Design (Period of specifying the product structure)
- Implementation (including module testing, integration, integration testing)
- Testing (Period for non development tests)
- Introduction (Period when the product is introduced)
- Others (All others. E.g. After care, pre development, etc.

In which phase was the event introduced.

In which development phase was the problem introduced (Strongly depends on the development life cycle used)

In which phase was the event resolved.

In which development phase was the problem resolved (Strongly depends on the development life cycle used)

The three fields 'Phase Discovered', 'Phase Introduced', and 'Phase Resolved' should have the same pick list.

5.2.10 Miscellaneous fields

The next data fields are used for miscellaneous purposes and don't fit in the previous categories.

- Binaries attachments
  Method to couple all kind of files like images, executables etc. to this event?

6 Organisational Issues

There are many different organizations, some are small, big, some operate in multiple cities or countries, others develop software that is a simple component in an overall system or product, and there are many more.

This chapter gives solutions how to implement these organisational differences.

6.1 Multi Disciplines

In some projects software development is only a sub-project in an overall system project. Other disciplines or departments can be electrical, mechanical, optical, service department or others.

If these other disciplines do not have their own registration system they can use the software system. But then measures have to be taken to handle these different event groups.

- How make department dedicated reports?

- Discipline related event managers.
  Each discipline (department, development team, etc.) involved in a multiple discipline project has to have a discipline related event manager to administer all related events within this discipline.
- How to assign an event to an discipline
  Besides the discipline related event managers, there must be a general event manager, who has no discipline restrictions. In the 'ENTERED' state, the general event manager assigns the event to a certain discipline. A discipline event manager is allowed to claim an event for his own discipline, if it is clear that the event should be handled by his discipline.

Multiple disciplines working on the same event.
See multiple sites, multi CCBs, and parent child relations

6.2 Multiple Sites

6.2.1 Problem description

It can be that different groups are working on the same product on different locations. This means that the availability of the tracking registration is a must.
Solutions are:

- A tools that is world wide assessable,
  e.g. a Word Wide Web interface or e-mail interface
- A tool that maintains the consistency between local on site installations and works with one 'virtual' database.
- The use of symbolic user for the remote parties
  (A lot of manual interaction and registration is still needed.)

Things to be decided are:

- Security
- Who should be able to submit, update or change the status of an event

One could say that a tool that supports multiple sites is a must to be able to do so. But formal agreements still have to be made so that everybody knows the rules and obligations. I would say that approximate 20% of the multisite work procedures is covered by the tool, the rest should be covered by the formal agreements, etc.

6.2.2 Possible Solution

All sites involved have to work virtually on the same event handler database. If the implementation should foresee in multiple copies of databases, synchronisation between these databases should take place on a real time base (packages used to update the local copies of the events should consist of delta's).

Besides the multi site issues mentioned already in chapter 'multiple discipline / product', it should also be possible to place the responsibility of a state transition in another site or keep it local but let some task be performed by another site.
It must also be possible to put the entire responsibility of a slave event (multiple product / project) in another site. The event handler must recognize different sites. E.g. user John at site A may not be mixed up with user John at site B.

6.3 Multiple CCBs or hierarchy in CCBs

In some projects software development is only a sub-project in an overall project. Other departments can be electrical, mechanical, optical, service department and others.

In such a structure a overall CCB exist who decides on all major issues and co-ordinates the different departments.
The responsibility of the lower CCBs is to manage their local events, report major problems to the overall CCB and to resolve events received from the overall CCB.

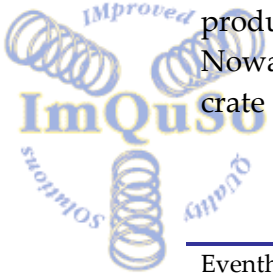
The communication and lifecycle interaction between the level of CCBs should be clear and formalised. (This is not an easy solution)

6.4 Multiple products of product variants

(Sometime called: platform product relation).
Some organisations create several products from the same source. E.g. they have a main/common source that that together with some specific files will generate their product variants. Sometimes these product variant are shipped to different part of the world (to be produced and maintained there) were they become a master themselves for their new (yearly) products.
Such a distribution could be:

- Taiwan To support all Asiatic countries.
- Brazil To support all South American countries
- Belgium To support the European market

Another grouping could be if we produce high-end and middle-end product from the same source. E.g. Software for an imbedded product is often developed for a single hardware architecture/platform that support a wide range of products.
Nowadays with the good, high speed electronic communication lines we can also crate a virtual archive accessible form all over the world.



I have never seen an organisation or a tool that could handle this. The only solution I have ever seen work is when a single department maintained all the sources for all products worldwide. But company politics made an end to this.

This is closely related to the reuse 'way of working'. For a more elaborate explanation on this subject and related PR CR handling see § 10.3 Change Control and Reuse of software.

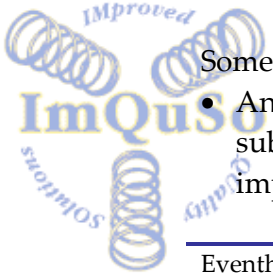
6.5 Multiple products / projects

Within a main project several different products can be developed or sub-projects can be initiated to develop a derived product. If a bug is found in the main product, it is possible the bug is valid for the derived product(s) or the related sub-project(s) too. In this way it should be possible to split-up (decompose) the main event into multiple 'slave events' that are introduced into the multiple project / product event handler. These slave events should be introduced into the ENTERED state. Initially the contents of the slave events are equal to the content of the master event. The handling of the events can be sequential, parallel or a combination of both. The main event, waiting in the MASTER-WAIT state, cannot be transferred to the CONCLUDED state until all slave events have reached an end state in their own lifecycle. The responsibility for this derived product / related sub-project can be in another site.

6.6 Multiple disciplines and product scenarios

With multi disciplines we mean that the event handler system will be used by different disciplines working together on the same system. E.g. software, electrical, mechanical, optical, etc.
Each discipline uses the same lifecycle but needs a filter to look at their on discipline related events. To solve this a simple attribute: Disicipline can be introduced that has a number of selectable values.

With product scenarios we main that we have one master product and we have several product derived from the master. Or we have a platform code and several products based on the same platform. Both product sets can be called a product family. Each event in a derived product needs to be investigated to see if the same event is also applicable for other product in the family.
If the tool supports master–slave and cloning solutions it can help here.



Some possible scenarios:

- An event enters the system. It is assigned to discipline A (ENTERED) and subsequently analysed (IN_REVIEW). There it is found to be an event that impacts several products, so it is advised to be decomposed into projects p1,

p2 and p3. Management (ANALYZED) agrees, so the event is sent to DECOMPOSE. There, the event manager creates three slave events, one for p1, p2 and p3. The original (now called master) event goes into MASTER-WAIT until all slave events have reached an end state in their life-cycle.

- An event enters the system. It is assigned to product P (ENTERED) and subsequently analysed (IN_REVIEW). There it is found that the event impacts different disciplines, so it is advised to be decomposed into disciplines d1, .., dn (all within the context of the product). Management (ANALYZED) agrees, so the event is sent to DECOMPOSE. There, the event manager creates n slave events, one for d1, .., dn. The original (now called master) event goes into MASTER-WAIT until all slave events have reached an end state in their lifecycle.

Note that the analysis done in the IN_REVIEW state of the original event is crucial for success for all master-slave eventhandling occurrences.

6.7 Multiple sites

If development or testing is done on different location all teams need access to the same event handler. Web based user interfaces (solutions) seems to be the best approach to tackle this problem.
Synchronising databases over multiple sites is possible but labour intensive and error prone.

6.8 Interaction with a Configuration Management tool

This is highly dependable on the type of coupling that is possible is between the two systems. See: 7.7 Coupling between a CM tool and change management

7 Scope Issues

7.1 Tools

There are many tools available for Change Tracking. You can find a couple of them via Internet links mentioned in: § 12 Change Tracking References

Selection criteria based on the functionality can be derived from this document. Infrastructure related requirements are defined next:

- User front-end (On which platforms can the tool be accessed)
  e.g. NT, Unix, Web-interface, Emails with keywords, etc.
- Databases (On which platform is the database repository located)
  e.g. NT, Unix, etc.
- Import facilities
  Needed when migrated from an old tool to a new tool.
  (They are seldom used!!)

- Export facilities (Maybe needed to make graphs etc.)
  Export to: excel, acces, asci, cvs, etc.

7.2 Capacity problem solutions

At some periods in time there can be more events then capacity to solve them. In that case it is nice if we can make a distinction between the events that have been selected to be resolved and the events that are people actually work on to resolve them.

This can be done with an attribute in the Open state (see …) but can also be implemented with an extra state in the lifecycle.
For the state solution we have the controlled and the uncontrolled solution.

Given the nest stages in or exemplarity lifecycle

In the **uncontrolled solution** we add a state Assigned that implies that an event is assigned to a developer/resolver to work on. As soon as he actually starts to work he moves the event to the Open state indication that the work has begun.

Advantages
- Single assigned of the CBB

Disadvantages
- Control on the priority of which the events will be resolved is at the developer/resolver
- Discipline needed to change the state of the event when work is started.

In the **controlled solution** we add a state Accepted that implies that the CCB has acknowledged that the event has to be solved but has no capacity to start the actual work. The CCB has to move the event to the Open state and allocate the event to a developer/resolver at the moment the capacity becomes available.

Advantages
- The CCB decides based in the available capacity of that moment who should work on what event
- CCB has full control over the flow.

Disadvantages
- An extra transition step is required from the CCB

7.3 Problem verification

The CCB members in most organisations are key persons with a huge workload. If the number of new entered events is more than the CBB can handle a filter should be applied so that the CCB can focus on the important decisions.

The filter can be that all new events are verified to so that only real problems reach the CCB. If this is combined with the "Analysis" state then the efficiency can even be improved a little bit more.

In this case the persons who do the event verification and analysis should operate independently of the CCB. They should pickup any new event immediately when it is entered in the system, do their thing and shift the event to the Verified state. This type of role allocation (opposite to person allocation) should be supported by the tool to be able to implement this.

7.4 Multiple tasks

It can be that multiple persons should perform work in the same state at the same event. E.g. a problem needs to be resolved in two different modules (bad design?) or two different persons should do a pre-analysis of a problem.

There are several solutions possible:

- The tracking tool is task oriented and supports multiple task for a single activity or state.
- Persons are allowed to re-assign each other while the event stays in the same state.
  Here the activities are done in sequence.
  Hazard: The event can be swapped around until the end of time.
- One person is assigned andd collects/administers all data from the activities and performs the state transition.
  Here some major offline tracking/co-ordination has to be done. But no extra tool functionality is needed.
- Use the parent – child option to resolve this see: § 7.5 Parent – child relations

The best solution is to avoid these situations if possible. (Otherwise the task based solution is preferred.)

7.5 Parent – child relations

If a tool supports parent child relations it is able to create a new event (optionally with the same contents) in another or the same database while the newly event still knows were it is created from (A child knows its parent).

Each child can then be handled as a normal event (and sometimes can have children itself).

Several decisions have to made here on the consequences of the relation:

- Do children follow the same lifecycle as the parent?
- Do testers test the children, the parent or both?
- Should the children and the parent do state changes in synchronisation?. E.g. nobody moves to test before all children are in the state Resolved.
- Maybe the parent should have a 'shorter' lifecycle'?
  From New to Parent to Closed.
- Reporting should be able to make distinction between parents and children.
  Five parents with each three children will result in 15 problems in the registration.
- Reports/views should be available to show parent – child relations.
  To be able to track the progress of their sisters or parent.

The parent - child relation seems like a nice option, however great care must be taken with implementing it.

7.6 Cloning

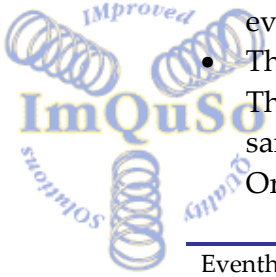
If a tool supports cloning it is able to create a new event (mostly with the same contents) in another or the same database. This can be handy is a software and electrical discipline need to work in an independent workflow on the same problem.

7.7 Coupling between a CM tool and change management

There can be several levels on interaction/coupling between the Change Management system and the Configuration Management system. Each implementation has consequences on the way the organisation manages their events and sources. Each coupling enforce the defined procedures but limits the flexibility of the 'project'.

The most obvious couplings are:

- None.
  Both systems are independent. The organisation/project has the responsibility to keep accounting information on what files have been changed for specific events.
- The event identification is stored in the CM system for every changed file.
  This can be done with development in branches where the branches have the same ID as an event.
  Or use a 'trigger' to store the event ID in the files CM meta information.



- A list of files (with their revision numbers) is inserted in a special field of the event in a database.
- The event identification of the event related to file changes is the first word in the check-in comment of the developer in the CM system.
- Change permission
  A developer is not allowed to modify a source/document unless he has an 'assigned' event to do so.
  Access restrictions can be on dedicated files or on a global level.
- Merge permission
  A Configuration Manager can't merge the changed source or branch unless the event has passed the test state.

The type of coupling used depends mostly on the maturity of the organisation. More mature organisations will generally use (are able to work with) a tighter coupling as less mature organisations

Note on Configuration Management
The usage of branches is done, to make sure that changes are done in an 'isolated' area and not done directly on the stable and verified sources. Afterwards such a branch needs to be merged into the master software by the Software Configuration Manager. Good software architecture makes it possible to have dozens of branches without having merge conflicts. A merge conflict occurs if changes in different copies of the same source need to be merged with the source master. A bad architecture is guaranteed to introduce merge conflicts and even prohibits working in isolation.

8 Data analysis

8.1 Tracking Metrics

Date of introduction (new)
Every state change
Date of every change

8.2 Analysis Metrics

The initial estimation of the time needed to fix this event.
Actual time needed for analysis
Comment to support / propose how the event is to be resolved (in the resolution state)
Comment to support / propose how the event is to be verified (in the verification state)

8.3 Resolution Metrics

Actual time needed for the fix
Comment to support / propose how the event is to be verified (in the verification state)

8.4 Verification Metrics

Actual time spend on the verification

8.5 Project Post Mortem Metrics

Note: some is this data is also collected for other purposes.
Total number of events (CRs and PRs)

- Rejected Events
- Equal Events
- Closed events
- (all other defined states)
- Total number of events passed to the next project (were on Hold)

Average effort per event.
Average lead-times

- Average decision time (From 'New' to "Open, Duplicate or Rejected").
- Average resolution time (From 'Open' to "Solved").
- Average verification time (From 'Test' to "Closed").
- Average lead time (From 'New' to an "end state").

How was the event discovered?

- In which phase was it discovered.
- In which phase was it introduced.
- In which phase was it resolved

9 Reporting

The tool should be able to generate report for different purposes.

9.1 Support for the CCB meeting

The Change Control Board needs to make decision on events that are in the states: New, Resolved, and Decision.
At regular intervals they also review all events that are in the hold state.

The CCB should also watch the progress of assigned events. E.g. get a list of events that haven't been changed in 5 days or changed in N days where N (number of days) is based on he priority of the event

Reports that the CCB needs to make the correct decisions:

- What's in state: New? (Detailed listing of every event)

- What's in state: Decision? (Detailed listing of every event)
- What's in state: Resolved? (Detailed listing of every event)
- What's in state: Verified? (Detailed listing of every event)
- What's in state: Hold? (3 line listing of very event, done once a month)
- Overview of events that didn't meet their progress deadline (1 line for each event)
  [ID, Type, State, Synopsis, Owner, Last Changed Date]
- Overview of event responsibility per person (1 line for each event)
  - Who is responsible in states: Analysis, Open, Test
  - With estimations of current open work. (Per event per allocated person)
  - Also an estimation overview for not allocated events

9.2 Progress reporting

In most organisations (with multiple projects), each project has to deliver a progress report with event info.
Typically used are:

- Historic growth graph
  Displays the historic growth of the events per state
  see: § 9.3.7 Historic growth graph of the events database per state
- Maturity Grid table
  See: § 10.1 Maturity Grid
- Estimations on the total expected events.

A special case can be the progress reminder mail that is send once a week?

- Everyone who has at least one event assigned to him gets a reminder mail.
  In the mail a list of all allocated events to that person.
  (for each product, states: Open, Assigned, Test

9.3 Post Mortem analysis support

Part of a Post Mortem project evaluation (after project ends) is a summary of performance indicators related to change control / management.
This is the typical information you would like to see:

9.3.1 An overview of the events and states.

A simple list of all states and the number of events that are in that state.

In most organization all events of a development have to be in an end state like: Closed, Rejected, Duplicate and Forwarded when to project is closed.
This enforces project management to decide what to do with 'open' events.
Most commonly they are transferred to a new development project.

9.3.2 An overview on how the events are found.

A table for CRs and/or PRs

How Found.	#	Relative %
Functional Tests		
System Tests		
….		
….		
Customer use		

9.3.3 An overview on when the events are found.

9.3.4 An overview on when the events were caused.

9.3.5 An overview on when the events were fixed.

A table for CRs and/or PRs

	When Found		When Caused		When Fixed	
Development phase	#	Relative %	#	Relative %	#	Relative %
Specification						
Design						
Integration testing						
Alpha testing						
Beta testing						

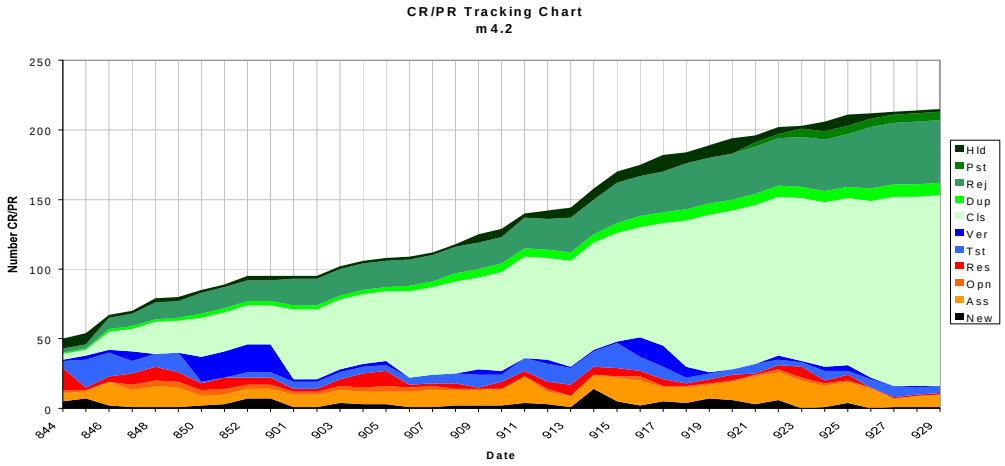
9.3.6 An overview of effort to fix a event per development phase

A table for CRs and/ or PRs

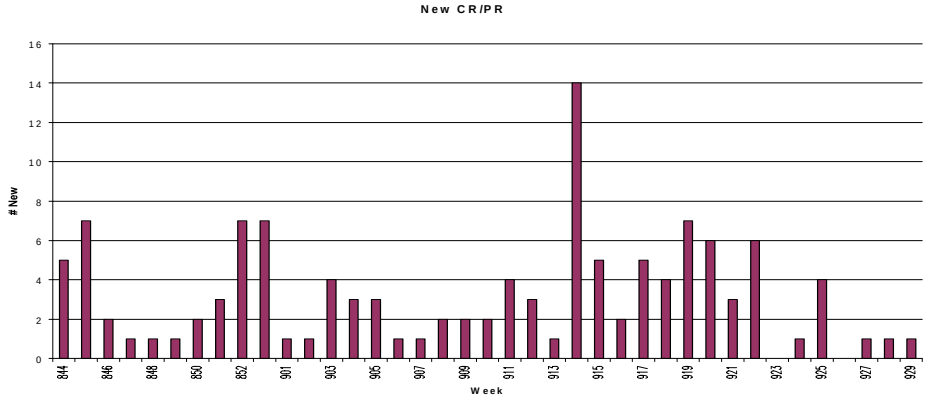
Activities where events are found.	Events related to specification activities	Events related to design activities	Events related to coding activities
Specification		----------	----------
Design			----------
Integration testing			
Alpha testing			
Beta testing			

9.3.7 Historic growth graph of the events database per state

(Based on weekly snapshots)



9.3.8 Overview of the number of events weekly found



Most of this information also serves it purpose at the weekly progress reporting and if f you have the correct tool support that creation of these tables/graphs should not be a problem

10 Related Subjects

10.1 Maturity Grid

10.1.1 Introduction

The design Maturity Grid has been developed to facilitate the decision “go” or “no-go” for passing project milestones.
Besides the milestone passing it offers the opportunity to calculate a Performance Indicator regarding the maturity of the design/product on a specific milestone.

For this reason every failure established during the evaluation period of the design models will be awarded with a gravity weight factor and an evolution factor.

10.1.2 Maturity Grid Parameters

Gravity of defects.

Every defect or shortcoming will be catalogued in ratio of gravity, viz.:

Gravity Code	Description
S	No conformity with safety standards or requirements.
A	Results in a not producible / usable products.
B	Results in a product, which can be produced with big problems or not be accepted by a critical customer.
C	Results in a product, which can be produced or sold with minor difficulties.

Evolution factor

Dependent on the gravity of a defect and the way to the implementation of a solution, the following evolution factors 0 up to and including 4 can be determined.

Evolution factor	Description
4	Cause unknown.
3	Cause known but solution unknown.
2	Solution implemented but not evaluated
1	Evaluation positive but not yet introduced in production.
0	Industrial introduced

10.1.3 Milestone decision

Maturity decision table.

		MS A	MS B	MS C	
Evolution	S	A	B	C	
4	(S4)	(A4)	(B4)	(C4)	
3	(S3)	(A3)	(B3)	(C3)	
2	(S2)	(A2)	(B2)	(C2)	MS D
1	(S1)	(A1)	(B1)	(C1)	MS E

10.1.4 Milestone Requirements

Milestone	Requirement
MS A	Sum of S4 and A4 classified defects must be zero.
MS B	Sum of S4, S3, A4, A3, B4 and B3 classified defects must be zero.
MS C	Sum of S4, S3, S2, A4, A3, A2, B4, B3, C4 and C3 classified defects must be zero.
MS D	Only C1 defects are allowed.
MS E	No defects are allowed.

NOTE: In case of combined milestones the requirements of the latest milestone are valid (e.g. A+B Implies use: B)

The milestone names used here are symbolic and should be replaced with the names used in the used in your software development life cycle.

10.2 Product Maturity based on the type of errors discovered

This is a summary from the article:
Orthogonal defect classification. A concept for in-process measurements.
By: R. Chillarege, I. Bhandri, J. Chaar, M. Halliday, D. Moebus, B. Bay and M.Y. Wong
Published in: IEEE Transactions on Software Engineering, Vol 18, Nov 1992

The authors describe a method to define the maturity of software, based on the nature/distribution of the found defects. To do so they divide the defects into several categories. Each of these categories has a relation to a general software development sub-process. The categories are:

Defect-type	**Associated –process**
Function	Design
Interface	Low Level Design
Checking	Low Level Design and Code
Assignment	Code
Timing/Serialisation	Low Level Design
Build/Package/Merge	Library Tools
Documentation	Publications
Algorithm	Low Level Design

- Function defect:
  A defect is one that affects significant capability, end-user interfaces, port interfaces, interface with hardware architecture, or global data structure(s) and should require a formal design change.

- Assignment defect:
  A defect indicates a few lines of code, such as initialisation of control block or data structure.
- Interface defect:
  Are defect in interfacing with other components, modules, device drivers via macros, call statements, control blocks or parameter lists.
- Checking defect:
  Are defects in program logic that has failed to properly validate data and values before they are used.
- Timing/Serialisation defects:
  These are defects that are corrected by improved management of shared and real-time resources.
- Build/Package/Merge defects:
  Describe the defects that occur due to mistakes in library systems, management of changes, or version control.
- Documentation defects:
  Defects that can affect both publications and maintenance notes.
- Algorithm defects:
  Are defects that include efficiency or correctness problems that affect the task and can be fixed by (re) implementing the algorithm or local data structure without the need for requesting a design change. Defect on e.g. efficiency or correctness.

It is possible to determine the maturity of the process by the type of defects you find when developing a product.
Given a development process one can derive expected behaviour. For instance, in most development processes where a design in conducted prior to coding and testing, the function defect should be fund early in the process and ideally very few in system test. Thus, the function defects should be diminishing through the process. On the other hand it is understandable that more timing and serialisation defects are found during system test. Assignment and interface defects can have profiles that peak at unit-test and integration test, respectively. Essentially, the defect type distribution changes with time, and the distribution provides an indication of where the development is logically.
The change in the distribution of the defect type thus provides a measure of the progress of the product though the process. At the same time, it provides a means to validate if a development is logically at the same place as it physically.

10.3 Change Control and Reuse of software

To be defined

11 Other Event Lifecycles (optimalisations)

11.1 Lifecycle with person, Role and Task allocation

11.1.1 The Concept

An event is controlled within one of three contexts within the event handler:

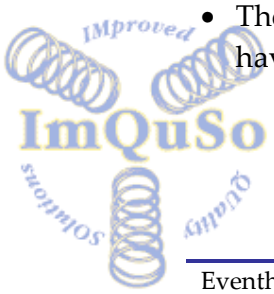
- **Managerial context**
  On this level it is decided what should be done with the event. E.g. in case it has to be solved, the developer and time of implementation will be defined. The project manager is responsible for these decisions, although he may assign a special person to do the actual work.
- **Functional context**
  This is where the event is controlled while it is being worked at. Each event is assigned to an owner. The owner of the event is responsible for the progress of the event in this context.
- **Working context**
  This is where tasks, assigned to specific developers, are performed for well-defined activities.

When an event is entered it starts in the managerial context. There, an owner is assigned. After that, an event goes back and forth between the managerial and the functional context. At the managerial level it is decided what actions are needed for an event. The event is transferred to the functional level, where the actions are controlled. After all actions are concluded, the event is transferred back to the managerial context. There is decided which further actions are needed. In principle, this process iterates until it is concluded at the managerial level, meaning that the event can be considered as finished. In practice, we will define a specific lifecycle in this document.

11.1.2 Roles

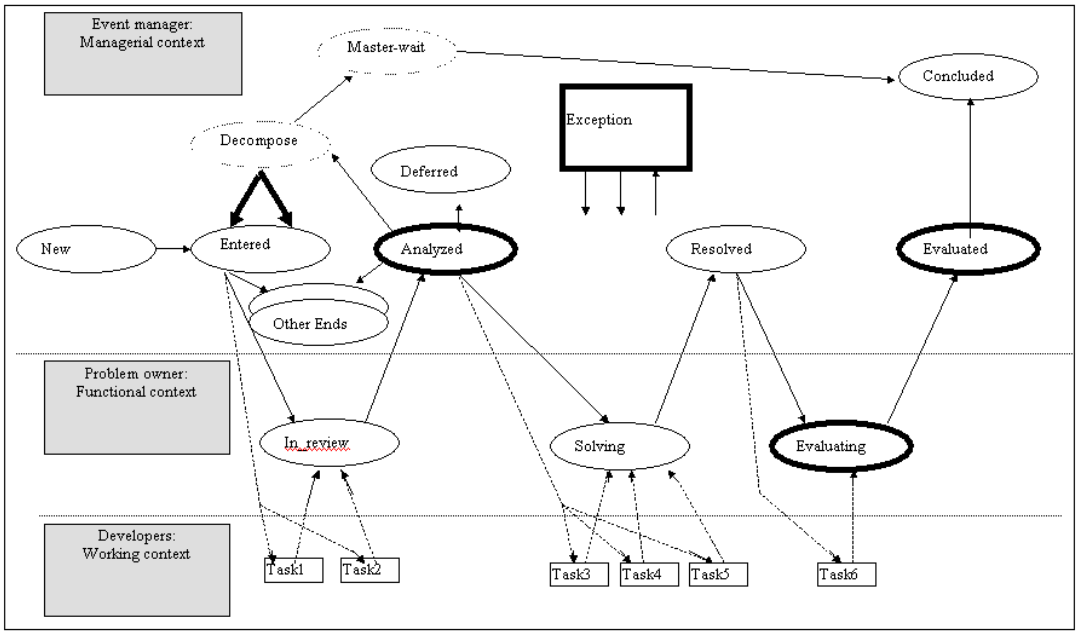
Within the event lifecycle the following roles can be distinguished:

- The **event manager** controls the flow of the events in the lifecycle. He makes state transitions and assigns tasks.
- The **problem owner** is responsible for the progress of the event in particular states where tasks are assigned to developers. The owner makes the state transition to the managerial level.
- The **developer** is responsible for the actual work of assigned tasks. Developers have no influence on the lifecycle of the event.



11.1.3 Event Life Cycle

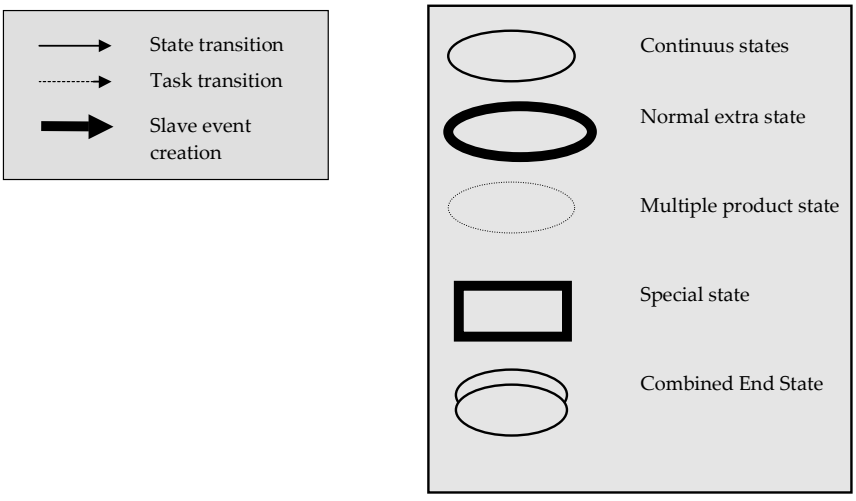
The 'file rouge' of the event tracking tool is the event life-cycle. It describes the states in which an event can reside, the possible transitions between the different states, and the tasks and responsibilities the user has.



Notes:

- 'Exception' state: input: from any state one can reach this state; outputs: from this state the event manager can reach any state. If the state transition reaches 'In_Review', 'Assigned' or 'Evaluating', tasks need to be defined as well.
- Task assignments are always associated with a state transition.

Legenda

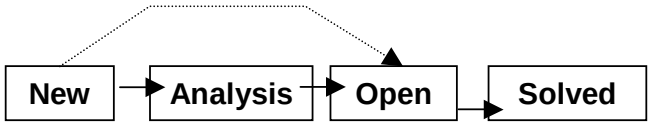


11.2 Capacity problem solutions

At some periods in time there can be more events then capacity to solve them. In that case it is nice if we can make a distinction between the events that have been selected to be resolved and the events that are people actually work on to resolve them.

This can be done with an attribute in the Open state (see …) but can also be implemented with an extra state in the lifecycle.
For the state solution we have the controlled and the uncontrolled solution.

Given the nest stages in or exemplarity lifecycle



In the **uncontrolled solution** we add a state Assigned that implies that an event is assigned to a developer/resolver to work on. As soon as he actually starts to work he moves the event to the Open state indication that the work has begun.

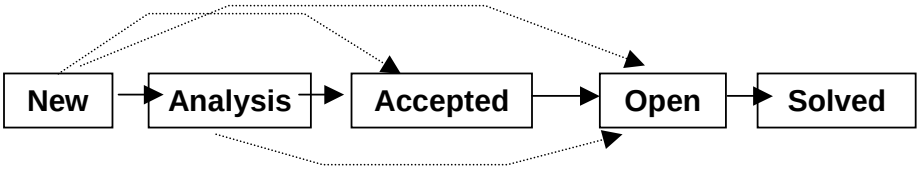
Advantages

- Single assigned of the CBB

Disadvantages

- Control on the priority of which the events will be resolved is at the developer/resolver
- Discipline needed to change the state of the event when work is started.

In the **controlled solution** we add a state Accepted that implies that the CCB has acknowledged that the event has to be solved but has no capacity to start the actual work. The CCB has to move the event to the Open state and allocate the event to a developer/resolver at the moment the capacity becomes available.



Advantages

- The CCB decides based in the available capacity of that moment who should work on what event
- CCB has full control over the flow.

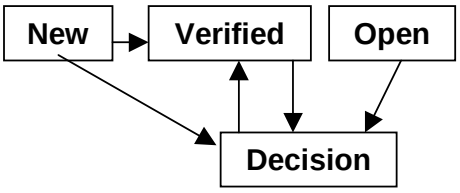
Disadvantages

- An extra transition step is required from the CCB

11.3 Problem Analysis

The CCB members in most organisations are key persons with a huge workload. If the number of new entered events is more than the CBB can handle a filter should be applied so that the CCB can focus on the important decisions.

The filter can be that all new events are verified to so that only real problems reach the CCB. If this is combined with the "Analysis" state then the efficiency can even be improved a little bit more.



In this case the persons who do the event verification and analysis should operate independently of the CCB. They should pickup any new event immediately when it is entered in the system, do their thing and shift the event to the Verified state. This type of role allocation (opposite to person allocation) should be supported by the tool to be able to implement this.

12 Change Tracking References

- Configuration Management Yellow Pages
  originally by: André van der Hoek
  now available at http://www.cmcrossroads.com/
  The best site for Configuration and Change management.
- ????

*

ImQuSo Improved Quality Solutions.
Po Box 169
5540 AD REUSEL, The Netherlands

Doc. ImQuSo-WP-2008-007 Rev. 0.5 2009-04-15