 THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE EUROPEAN UNION'S SEVENTH FRAMEWORK PROGRAM (FP7/2007-2013) FOR CRYSTAL – CRITICAL SYSTEM ENGINEERING ACCELERATION JOINT UNDERTAKING UNDER GRANT AGREEMENT N° 332830 AND FROM SPECIFIC NATIONAL PROGRAMS AND / OR FUNDING AUTHORITIES.


**CR**itical S**YST**em Engineering **A**cce**L**eration

# System engineering performance analysis report
# D400.020

## DOCUMENT INFORMATION

| | |
|---|---|
| **Project** | CRYSTAL |
| **Grant Agreement No.** | ARTEMIS-2012-1-332830 |
| **Deliverable Title** | D400.020 |
| **Deliverable No.** | 021 |
| **Dissemination Level** | CO |
| **Nature** | R |
| **Document Version** | V1.00 |
| **Date** | 2014-04-29 |
| **Contact** | S. Roubtsov |
| **Organization** | TU/e |
| **Phone** | +31 40 247 2720 |
| **E-Mail** | s.roubtsov@tue.nl |

## AUTHORS TABLE

| Name | Company | E-Mail |
|---|---|---|
| S. Roubtsov | Eindhoven University of Technology (TUE) | s.roubtsov@tue.nl |
| Ilse van Binsbergen | Philips | ilse.van.binsbergen@philips.com |

## CHANGE HISTORY

| Version | Date | Reason for Change | Pages Affected |
|---|---|---|---|
| 0.5 | 16.04.2014 | First draft submission | all |
| 0.6 | 24.04.2014 | Reviewers' remarks incorporated | 7,12, 13,14,31 |
| 1.0 | 25.04.2014 | Final version | 7,8,9 |
| | | | |
| | | | |
| | | | |
| | | | |

# CONTENT

## Content of Figures



## Content of Tables

# 1 Introduction

## 1.1 Overview

This document describes the performance evaluation of the product engineering process at Philips Healthcare. It includes the definitions of performance indicators used, the results of their measurement on a chosen baseline project, as well as the analysis of the results.

The report describes two sets of performance indicators we have chosen, namely, *Phases Caused vs. Found KPIs and Defect Management process throughput KPIs*. The KPI measurements performed on the baseline project reflect the current state of the product engineering process at Philips Healthcare.

The measurement results, obtained for different groups of defects as well as different phases of the defect management process, are analysed. The KPIs' relevance to the expected improvement of the product engineering process is discussed.

As a result of this study, the chosen KPIs are to be used (with possible extensions) in future projects to measure the improvement of the product engineering process due to applying advanced engineering methods.

## 1.2 Role of deliverable

### 1.2.1 Purpose and goals

The purpose of this study is to develop *the approach* as to how *to assess the efficiency of applying advanced engineering methods to the product development process.*

The goal of the current stage of the study is to determine the *Key Performance Indicators (KPI)* by means of which the *state of the product engineering process can be evaluated and compared between projects.*

The goal of this report is to describe the results of 'zero-measurement', that is, the evaluation by means of chosen key performance indicators of the *current state of the product engineering process* at Philips Healthcare. In the future, during the duration of the Crystal project, the results of 'zero-measurement' are to be compared with the results of the measurements for newer projects to be developed with use of advanced engineering methods, such as Agile development process.

### 1.2.2 Scope

The scope of this report is narrowed to a sample already finished project. Also for the zero-measurement we decided to focus on the data which can be obtained from one particular source, namely, Product Defect Management process.

## 1.3 Relationship to other CRYSTAL Documents

This document is part of the deliverables within work package WP4.0 Coordination Healthcare, which belongs to the Healthcare domain (SP4). It is an output of Task 4 SP Quality Management.

## 1.4 Structure of this document

Chapter Measurement (2) describes the system setup (2.1), definitions of measurements performed (2.2), as well as a project chosen for the measurements (2.3).

Chapter 3 describes the results of the 'zero-measurement' and its analysis (3.1 and 3.2). Chapter 4 concludes the report.

# 2 Measurement approach

Due to the scope of the study, the first step was to find the means and tools capable to retrieve the data related to the Defect Management process. The choice of possible KPIs to use also depends on the available data. That is why we start with the description of a measurement system setup. After that, this chapter describes the types of data used for the measurements, and the KPIs we have chosen. Finally, the data selected for the baseline project is described.

## 2.1 System Setup

To perform the described in the previous section 'zero-measurements' we used the measurement system setup which consists of two interconnected parts (Figure 2-1): the QlikView tool with the underlying Information Platform and the FRASR tool chain. In the following sub sections we describe the both.

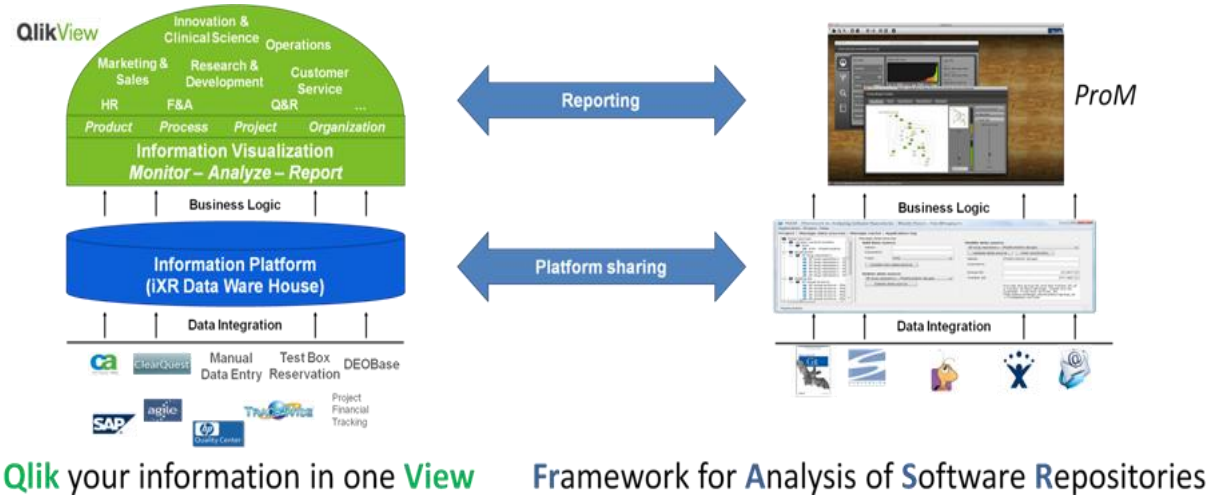



Figure 2-1: System Setup

### 2.1.1 QlikView description

QlikView tool has access to different aspects of product related data at Philips Healthcare. QlikView is described in [Crystal D_403_901, 2014] Crystal project report.

### 2.1.2 FRASR description

FRASR (FRamework for Analyzing Software Repositories) is a software repository mining tool [FRASR, 2011]. The main feature of FRASR, which distinguishes it from similar tools, is its ability to retrieve information from different sources: bug tracking systems, source code repositories, email archives, etc, and integrate ("bind") them in a flexible semantic way. Also FRASR possesses an elaborate set of filtering capabilities. The goal is to use all this data for better understating software development process by answering "Software engineering questions". FRASR itself doesn't answer those questions but rather provides input for process mining and analysis tools such as ProM [ProM, 2014] and Disco [Disco, 2014].

ProM makes use of a wide variety of process mining and analysis techniques and algorithms implemented with extensive visualisation capabilities. However, its data integration features are more limited. That is why, within this project, we make use of both tools in a tool chain.

Additionally, we used the Disco [Disco, 2014] tool, which possesses some useful features with regard to workflow modelling and performance visualisation.

### 2.1.3 QlikView and FRASR+ProM interoperability

The information for the FRASR+ProM tool chain is provided by QlikView. Within this project, we intend to use QlikView output obtained from the following sources of the underlying Information Platform (Figure 2-1) described in [Crystal D_403_901, 2014]:

- IBM Rational ClearQuest – an application lifecycle management (ALM) software for change and defect tracking.
- IBM Clarity - a resource management solution that integrates budgeting, planning, reporting, consolidations, analytics.
- IBM Rational ClearCase - a software configuration management solution that provides version control, workspace management, and build auditing.

QlikView is able to prepare the information from the mentioned above sources as files with comma-separated values of different fields. These files can be used either by FRASR, when we need its source sharing and filtering capabilities ("Platform Sharing" arrow in Figure 2-1), or directly by process mining analysis and visualisation tools from the ProM family ("Reporting" arrow in Figure 2-1).

## 2.2 Measurements definition

The following sub sections introduce the data to measure and describe the KPIs we have chosen to measure as the "zero-measurement".

### 2.2.1 Data

For "zero-measurement" we have decided to concentrate on the data from the Clear Quest Defect Management system. QlikView's underlying database provides more than 80 different fields, among which we have chosen the fields specified in Table 2-1. We consider these data most interesting as they reflect a variety of aspects of the product development, e.g., it's phases, components, the duration of defect resolution, the stages where defects occur and resolved, as well as the severity and types of the defects.

| Field | Description | Possible values |
|---|---|---|
| *DefectId* | Unique identifier of a CQ record | e.g. CVPRJ00005775 |
| *Date* | Time stamp when defect was put into the CQ system | e.g. 4/14/2010 9:06:47 AM |
| *CausedInPhase* | System engineering phase where defect has been caused | REQ – requirements;DES – design; IMP – implementation; INT – integration; VER - subsystem verification; VAL – system verification/validation; OPER - operations; FIELD -production |
| *Component* | Where defect has been found | e.g., Software; Testware; Mechanics; Electronics; Service documentation; Product requirements; Product documentation; Project documentation; Maquet; particular parts of hardware: Magnus reverse tabletop; Rail cover |
| *FoundInPhase* | System engineering phase where defect has been found | same as for CausedInPhase |
| *RootCause* | result of defect root cause analysis | Same as in CausedInPhase with more granularity, e.g., IMP: Exception handling; REQ: Functional req; DES: Physical design; |
| *Severity* | Severity of the defect | 1-Critical; 2-Major; 3-Average; 4-Minor; |
| *ProblemType* | Differentiates true defects from new feature (change) requests | PR- problem request; CR - change request |

Table 2-1: Data fields

### 2.2.2 Defect Classification, Phases Caused vs. Found Key Performance Indicator

As the first measurement we have chosen the Phases Caused vs. Found Key Performance Indicator [CV X-Ray, 2012] to get more insights, in which development phase defects have been introduced (caused) and found and how how far away from each other these phaser are.

The goal of this KPI is “To reduce the costs of appraisal and rework by finding defects as early as possible. (shifting the curve of the Phase Found towards the curve of the Phase Caused in Figure 2-3) and by reducing the number of defects. (Improvement of Product Realization process and the quality and reliability of products)”.

Thus, the rationale of this KPI is that its improvement should lead to both decreasing the hight of the curves in Figure 2-2 as well as closing the gap between them moving Defect Found curve to the left.

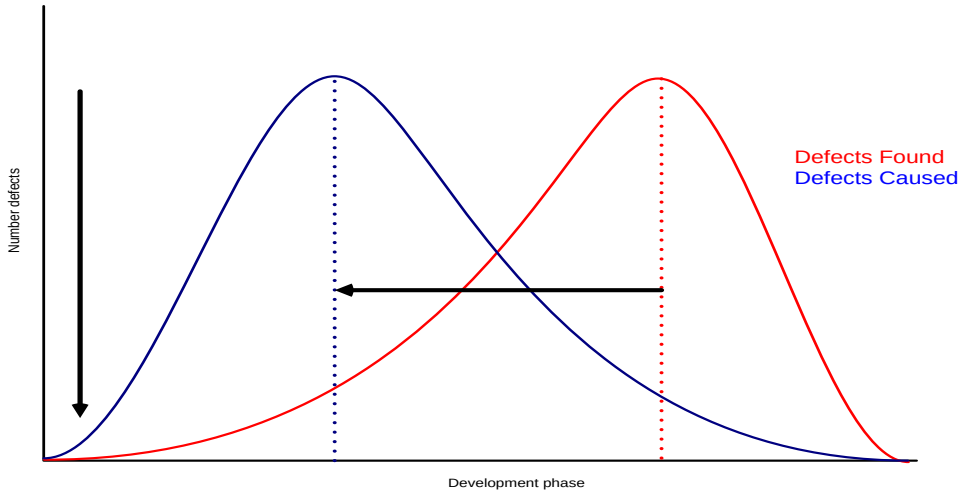



Figure 2-2: Phases Caused vs. Found Key KPI rationale: black arrows show the direction of the improvement

The KPI is defined in [CV X-Ray, 2012] as follows:

“Count the number of defects per development phase, in which the defects are caused and count the number of defects per development phase, in which the defects are found.

These values can be plot into two different graphs.

1. The total number of defects, caused in the specified phases and the total number of defects found in the specified phases. (Figure 2-3)

2. A matrix (defect classification map) presentation showing in which phases the defects, caused in a particular phase, have been found (see example presentation in Figure 2-4).

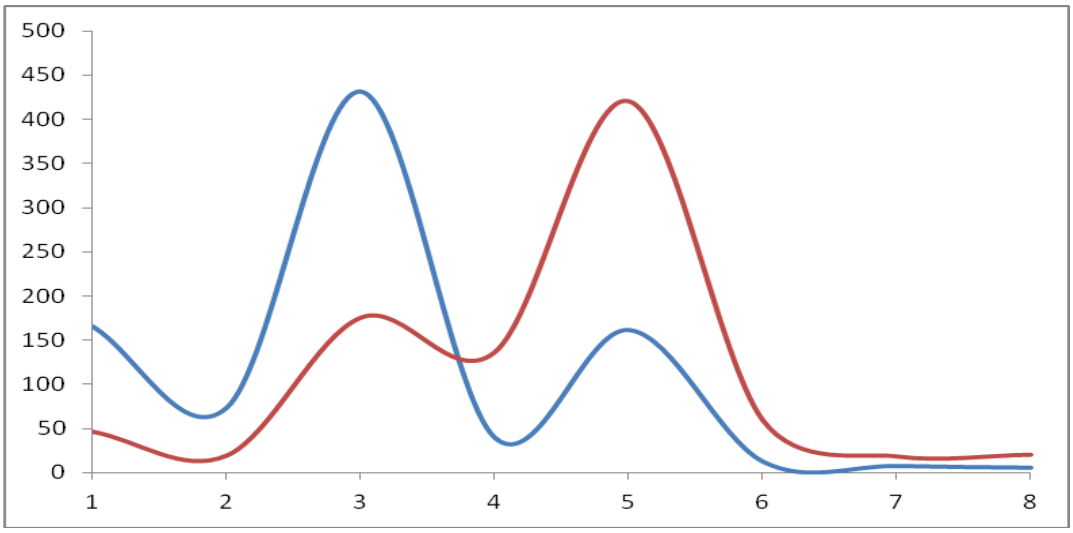
Colour requirements for this map representation are:

- Unused matrix fields: Blue
- The matrix fields for which apply:

  Phase caused = Phase found: Green

- The matrix fields of the two phases found directly following the phase, in which the defect is caused: Yellow
- Other matrix fields: Red”

The KPI description above refers to the following product engineering phases: REQ – requirements; DES – design; IMP – implementation; INT – integration; VER - subsystem verification; VAL – system verification/validation; OPER - operations; FIELD - production.




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
| Phase Caused | # | # | # | # | # | # | # | # |
| Phase Found | # | # | # | # | # | # | # | # |

Figure 2-3: Phase Defect Caused versus Defect Found Defect distribution in phases: REQ – requirements; DES – design; IMP – implementation; INT – integration; VER - subsystem verification; VAL – system verification/validation; OPER - operations; FIELD -production problem requests

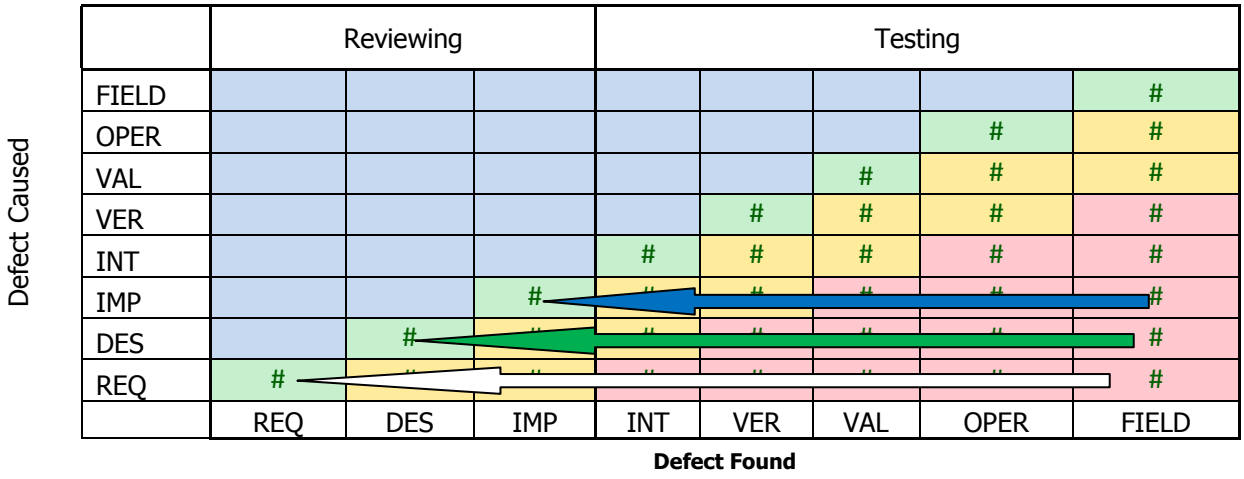



Figure 2-4: Defect classification map

The question remains as to how can this indicator reflect the improvement of the product development process? And by what means? The arrows in Figure 2-4 indicate the expected (and arrowheads – ideal) results of applying CRYSTAL engineering methods: as the methods aim at early defect detection, the numbers of defects are expected to migrate towards the 'green diagonal' should the corresponding engineering methods be used [Crystal D_403_901, 2014].

- White arrow to the requirements Defect Found phase: by using the engineering methods of Requirements validation and visualization.
- Blue arrow to the implementation Defect Found phase: by means of engineering methods Model driven development and code generation

- Green arrow from requirements to the design phase for Defect caused: by using the engineering methods such as hardware simulation in MatLab and software test with in-the-loop simulation.

As we can see the general idea of applying the above engineering methods is to move the numbers of defects found Testing to Reviewing phases.

In this project we use a derivative from this KPI which shows more adherences to the requirements to KPIs. According to Kitchenham [Kitchenham, 1996, p. 103] the main requirements are:

- "**Quantifiability**: KPI has to be a number.
- **Sensitivity**: Sensitivity expresses how much the performance must change before the change can be detected. Therefore, a sensitive indicator is able to detect even minor changes in performance.
- **Linearity**: Linearity indicates the extent to which process performance changes are congruent with the value of a certain indicator. Or, conversely, a small change in the business process performance should lead to a small change in the value of a corresponding performance indicator, whereas an ample performance rise should also lead to strong change in the level of the performance indicator[1].
- **Reliability**: A reliable performance indicator is free of measurement errors. To illustrate, if a certain business process has to be rated through a given performance indicator by different experts, the results should not depend on the subjective evaluation of an individual.
- **Efficiency**: Since the measurement itself requires human, financial and physical resources it must be worth the effort from a cost/benefit point of view.
- **Improvement-orientation**: Performance indicators should emphasize improvement rather than conformity with instructions. Therefore, measuring billing errors, number of safety violations, data entry errors and the like do not create an atmosphere where feedback sessions are viewed in a positive, constructive light…"

From the point of view of these requirements, the Phases Caused vs. Found KPI still needs to be made *quantifiable*. In this way, also *sensitivity and linearity* could be achieved. As far as the rest of the requirements concerns, it's *efficient*, because the required data is already available from QlikView and its *improvement orientation* is proven by the previous discussion in the beginning of this sub section.

*Reliability* is harder to prove as it depends upon error prone expert analysis of the phase that caused the defect. However, it is not completely subjective because, if this parameter is set up with a mistake, we assume that as a result the corresponding defect cannot be resolved properly until the caused phase is identified correctly. Indeed, to resolve the defect we need to return to the corresponding phase. In assumption that in such a case the corresponding field in the defect management system is corrected (such possibility exists), we conclude that the reliability if this KPI is acceptable.

We propose the following derivative KPIs, developing which, apart from the mentioned above requirements, we also have taken into account the possibility to compare different projects. In particular, this means that the KPIs have to be in ***relative values making them independent on the size of the project.***

- Percentage of defects found in the red zone:

$$InR = \sum\#\text{def of defects found in the red zone}/\sum\text{Total \# def}*100\%,$$

This KPI ranges from 0% (no defects in red zone) to 100% (all defects in red zone). It is sensitive to process improvement and the more defects are removed from the red zone the higher the percentage (linearity).

- Average distance from the green zone:

$$AvD = (\sum d_i*\sum\#\text{def with } d_i)/\sum\text{Total \# def},$$

where di =0,1,..,7 is the distance between the product engineering phases where a defect was causes and found

[1] As it follows from this definition linearity does not necessarily mean strict linear dependency in mathematical sense, just 'big change in process leads to big value change' and the other way around.

AvD ranges from 0 (all defects are in green zone) to 7 (all defects are caused at REQ stage and found at FIELD stage). The KPI decreases proportionally to the number of defects moved closer to the diagonal in Figure 2-4 following the extent to which the defect management process improves.

It may be noticed that not all defects are equal with regard to their cause-effect contributions. The above KPIs taking into account per defect averages, although quantifiable, hide those differences (as well as the initial Phases Caused vs. Found indicator does). A possible way to take into account different types of defects is to further separate them into groups. The data contained in the defect management database allows for this. We measured the above KPIs for the whole dataset as well as for the selected values of several data fields (see Table 2-1):

- PR and CR problem types defects
- Defects of Critical and Major Severity
- Defects in component types:
  - Software
  - Mechanics and Electronics (in different domains)
  - Documentation

If necessary, this grouping may be extended by other types of information in the fields present in the defect database. Still, cause-effect contributions of selected groups of defects need to be assessed by using additional research. It may require analysis of different data sources, for example, Clarity resource management database, which is out of the scope of this report.

### 2.2.3 Defect Management Process KPI

As the second measurement we have chosen the duration of the defect management process [ClearQuest, 2012]. Currently there are three different defect management processes in place, one for older projects, compliant to Defect Management Procedure, and two - for newer ones, defect management of which follows Product Defect Management Procedure and Engineering Issue Defect Management Procedure. As we do not intend to use for our zero-measurement newer, mostly still unfinished projects, we do not show their defect management workflows here.

Figure 2-5 shows the state diagram describing the defect workflow from its submission to the following closing states: *Validated, RejectAccepted, Forwarded and Duplicate*. Numbers on arrows indicate actions performed during the transitions between the corresponding states. The list of all possible actions and the corresponding is show in the table in Annex I. We refer to those actions further in section 3.2.3 while discussing the durations of different defect management process transitions and states.

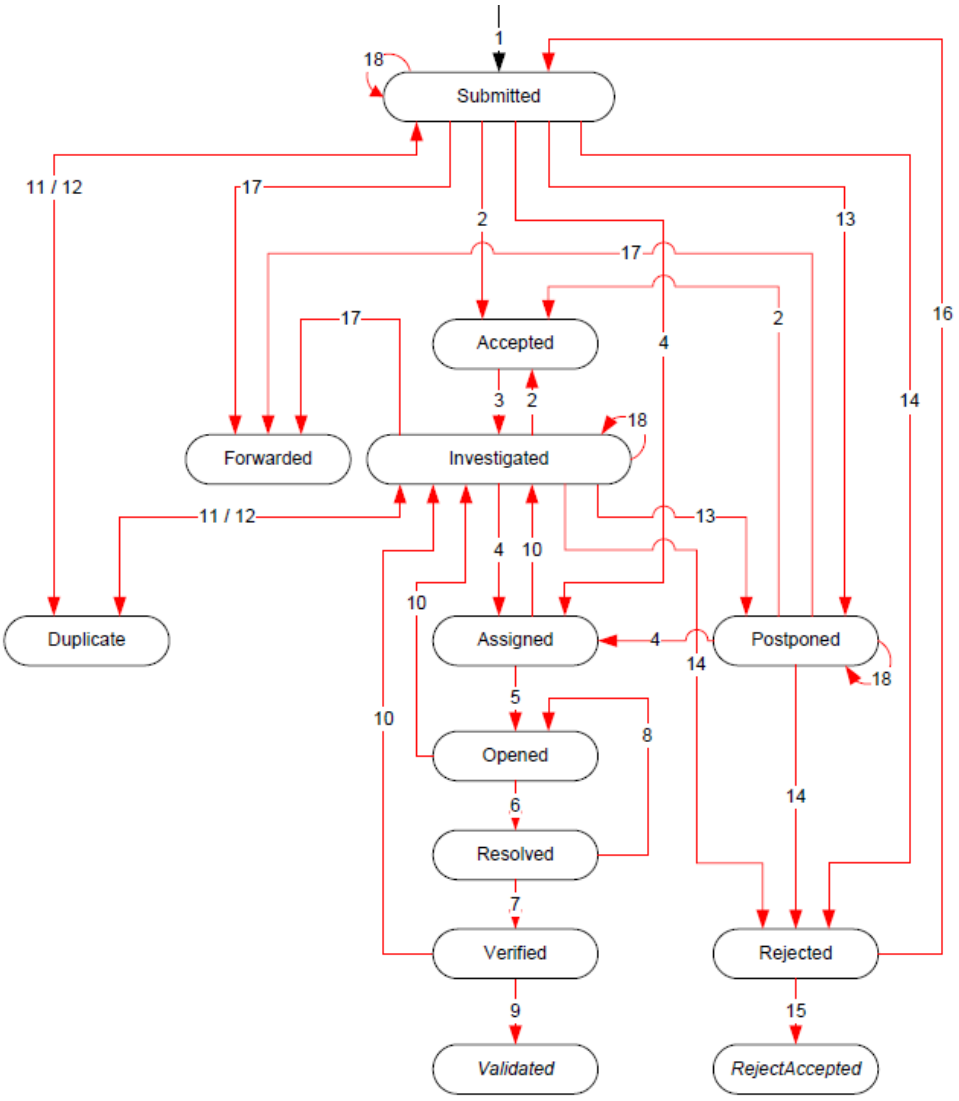



Figure 2-5: Defect Management Procedure workflow.

The idea of measurement is the following:

We need to quantify the duration of individual steps and find the most time consuming ones. This measurement should reflect the defect management from a different perspective than the first one described in the previous sub section:

Rather than shortening the gap between defect caused and defect found stages, it should ***facilitate the shortening the duration of the defect found stage.***

We propose the following KPIs:

- Mean defect resolution time:

| $RT\mu = \sum Total\ resolution\ time / Total\ \#\ def$ |
|---|

Where *resolution time* is counted for all *complete cases*, that is, the ones from the start and ended in *Validated* or *RejectAccepted*, *Forwarded*, or *Duplicate* state (Figure 2-5)

- Median resolution time:

| RT~ = statistical median[2]: 50/50% complete defect cases have faster/slower resolution times |
|---|

The specified KPIs are *quantifiable*. The *sensitivity* and *linearity* depend on the statistical distribution of the defects. Under the assumption that this distribution is of the same type for all projects, those two requirements should hold.

*Reliability* is assured since the KPIs are based on the objective data (time within which the defect is closed).

Finally, the KPIs can be used to measure the *improvement* of the same project as well as to compare different projects in similar domains. The goal is to decrease the KPIs values in order to decrease the duration of the Defect Found phase. On the other hand, it has to be noticed that these KPIs reflect the ***throughput duration***, not the actual resolution time of particular defects. This is because the data taken solely from the Clear Quest database do not reflect resource management data present in the Clarity database only. Thus, *the improvement orientation of these KPIs may be limited and requires further assessment.*

Apart from total duration of the defect management process, the KPIs can be used to measure the duration of its different stages. QlikView categorizes them as shown in Table 2-2:

| Defect Open/Closed | Category | State |
|---|---|---|
| Open | New | Submitted |
| Open | New | Accepted |
| Open | New | Investigated |
| Open | Engineering | Assigned |
| Open | Engineering | Opened |
| Open | Verification | Verified |
| Open | Verification | Resolved |
| Open | Verification | Rejected |
| Closed | Closed | Duplicate |
| Closed | Closed | Forwarded |
| Closed | Closed | RejectAccepted |
| Closed | Closed | Validated |

Table 2-2: Categorization of defect management process stages

RT$\mu$ *and RT*~ can be measured for:

- Defect management process as a whole (final states: Validated, RejectAccepted, Forwarded, Duplicate) as well as for each final state separately
- The stage Defect Open to assess which share of the whole defect management process the actual defect resolution part constitutes.
- Categories: New, Engineering, Verification, Closed separately

Also, for every item above the duration of defect management process separately for Critical, Major, Average and Minor severity defects can be measured. This way we can assess how fast the severe defects are resolved with respect to average values.

[2] The statistical median value for discrete probability distributions, see http://en.wikipedia.org/wiki/Median

## 2.3 Project under measurement description

### 2.3.1 Baseline project for the KPI measurement

For the Phases Caused vs. Found KPI we have chosen a finished Mechatronic project with functionality similar to the current Mechatronic project with multi-axis movements. This project serves as a baseline for zero-measurements.

This project uses the waterfall method and did not apply engineering methods such as Requirements validation and visualization, Model driven development and code generation, and Hardware simulation in MatLab and software test with in-the-loop simulation.

### 2.3.2 Project data snapshot for the Phases Caused vs. Found KPI measurement

We used the current state defect management data exported from Clear Quest as a csv-file.

It contains 1091 unique defect identifiers, all of which except one are currently closed.

We used the *FoundInPhase* field (Table 2-1) to identify the phase where defect had been found. As far as the Phase Caused concerns, there are two field candidates: CausedInPhase and RootCause. Our analysis revealed that there was significant difference between the information in these two fields. For instance, for 157 cases where *FoundInPhase and CausedInPhase* both indicate the verification phase, the *RootCause* field shows only 27 values containing the VER phase identifier.

Further, defect management process analysis indicated that *CausedInPhase* is filled in when the defect is submitted, *RootCause* is the result of the investigation step in the defect management workflow (Figure 2-5). The reason of discrepancy may be that at the submission stage the location of the defect cause cannot be properly identified. Thus, we opted to use the *RootCause* as the field for Phase caused indication.

The analysis of data representativeness reveals that all necessary for the KPI measurement are filled except *RootCause* and *FoundInPhase* are sometimes left blank or filled with not specified values, e.g. “not applicable” or “other”. We filtered out those fields. That left us with 920 defect cases out of 1091 or 84%, which is still representative enough for further KPI calculations.

### 2.3.3 Project history data for Defect Flow mining

The defect management history file provided by QlikView’s database contains 1430 unique defects related to the Baseline project. This is 23% more that the number of unique defects (1091) in the file reflecting the current state. Using ProM shows, we observe that these ‘extra’ defects correspond to the case variants for which defects don’t reach one of the final sates of defect management workflow in Figure 2-5. According to the explanation by the developers, those are the defects moved from Baseline to other projects. We just excluded those cases from our measurements.

# 3 Measurement results and analysis

In this chapter we present the results of measurements described in section 2.2. Also we provide some analysis results.

## 3.1 Phases Caused vs. Found KPI

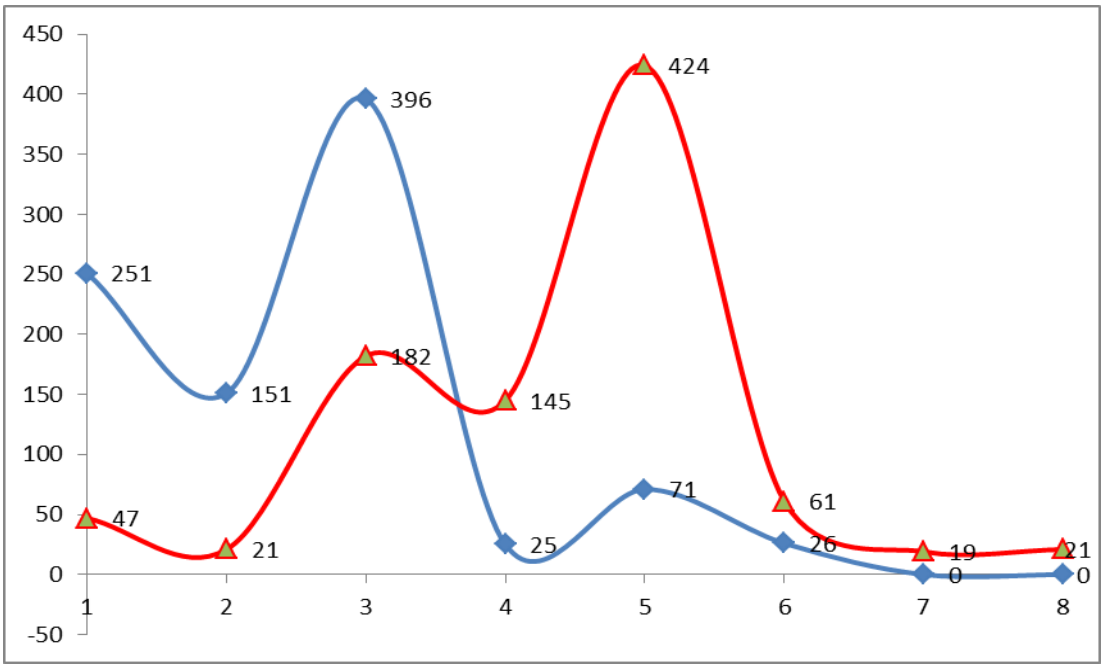
### 3.1.1 Results

First we measured the Phases Caused vs. Found KPI as it described in section 2.2.2. A defect distribution graph and a defect classification map for the Baseline project are shown in Figure 3-1 and Figure 3-2 correspondingly.

The following figures represent the results for

- PR and CR problem types (Figure 3-3 and Figure 3-4; Figure 3-5 and Figure 3-6, correspondingly)
- Defects of Critical and Major Severity combined (Figure 3-7 and Figure 3-8)
- Component types:
  - Software (Figure 3-9 and Figure 3-10)
  - Mechanics and Electronic (Figure 3-11 and Figure 3-12)
  - Documentation defects (e.g. in Service documentation, Product requirements documentation, Project documentation) (Figure 3-13 and Figure 3-14)

Table 3-1 contains the results of measurements Phases Caused vs. Found KPI values for the Baseline project for whole data and chosen fields.



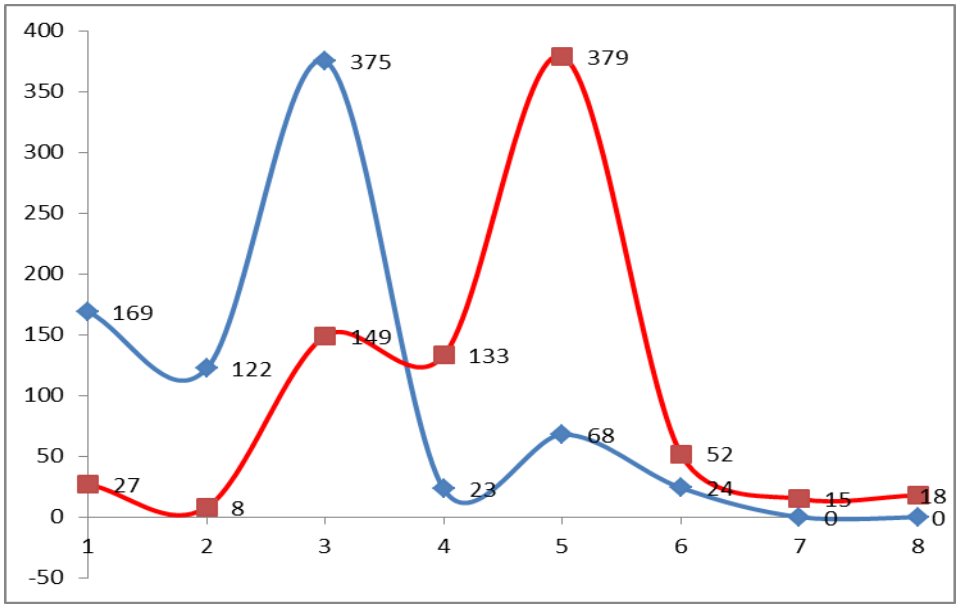

| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 251 | 151 | 396 | 25 | 71 | 26 | 0 | 0 |
| Phase Found | 47 | 21 | 182 | 145 | 424 | 61 | 19 | 21 |

Figure 3-1: Phase Defect Caused versus Defect Found Defect distribution for Baseline project

| Defect Caused | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 3 | 0 | 0 |
| VER | | | | | 38 | 4 | 2 | 0 |
| INT | | | | 6 | 10 | 1 | 0 | 3 |
| IMP | | | 95 | 65 | 199 | 27 | 5 | 4 |
| DES | | 13 | 35 | 25 | 53 | 10 | 8 | 5 |
| REQ | 39 | 8 | 33 | 36 | 107 | 16 | 4 | 8 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |

**Defect Found**

Figure 3-2: Defect classification map for the Baseline project

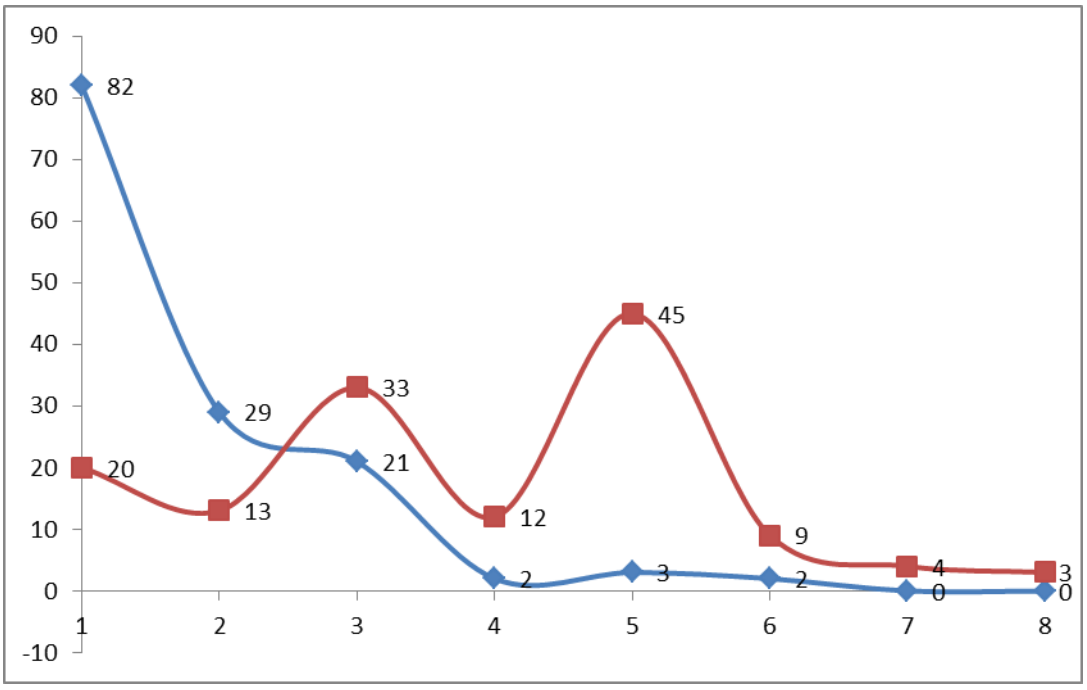



| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 169 | 122 | 375 | 23 | 68 | 24 | 0 | 0 |
| Phase Found | 27 | 8 | 149 | 133 | 379 | 52 | 15 | 18 |

Figure 3-3: Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Problem type PR (Problem Request) only

| Defect Caused | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 3 | 0 | 0 |
| VER | | | | | 37 | 4 | 2 | 0 |
| INT | | | | 6 | 10 | 0 | 0 | 3 |
| IMP | | | 90 | 63 | 188 | 24 | 5 | 4 |
| DES | | 4 | 27 | 24 | 46 | 10 | 5 | 5 |
| REQ | 22 | 4 | 16 | 27 | 81 | 11 | 3 | 5 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
| | Defect found | | | | | | | |

Figure 3-4: Defect classification map for the Baseline project. Problem type PR (Problem Request) only



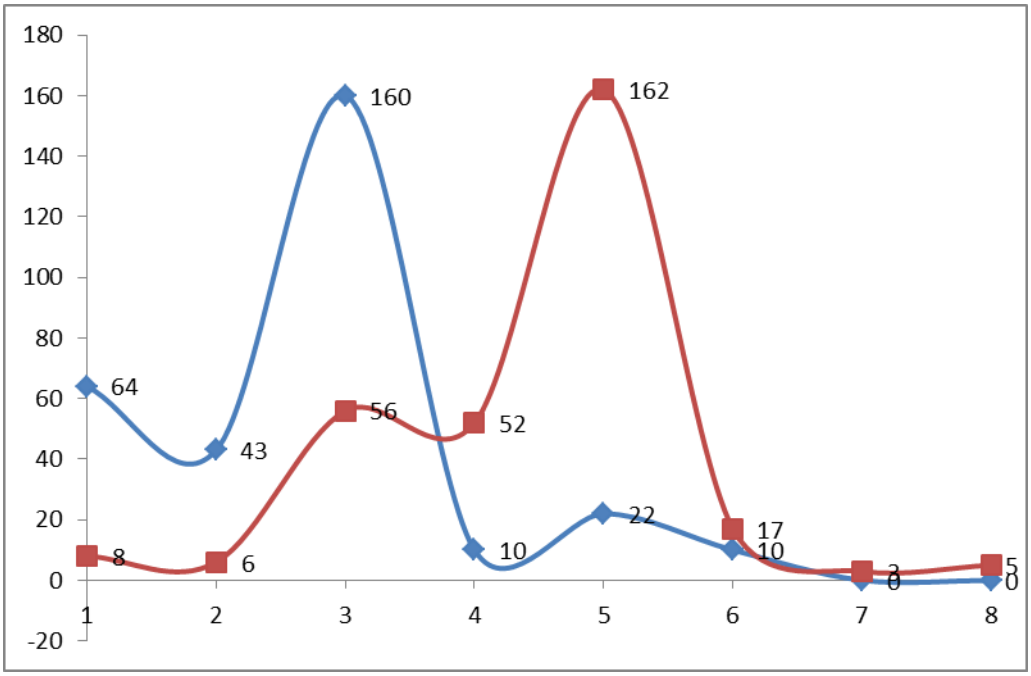

| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 82 | 29 | 21 | 2 | 3 | 2 | 0 | 0 |
| Phase Found | 20 | 13 | 33 | 12 | 45 | 9 | 4 | 3 |

Figure 3-5: Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Problem type CR (Change Request) only

| Defect Caused | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 0 | 0 | 0 |
| VER | | | | | 1 | 0 | 0 | 0 |
| INT | | | | 0 | 0 | 1 | 0 | 0 |
| IMP | | | 5 | 2 | 11 | 3 | 0 | 0 |
| DES | | 9 | 8 | 1 | 7 | 0 | 3 | 0 |
| REQ | 17 | 4 | 17 | 9 | 26 | 5 | 1 | 3 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |

**Defect Found**

Figure 3-6: Defect classification map for the Baseline project. Problem type CR (Change Request) only



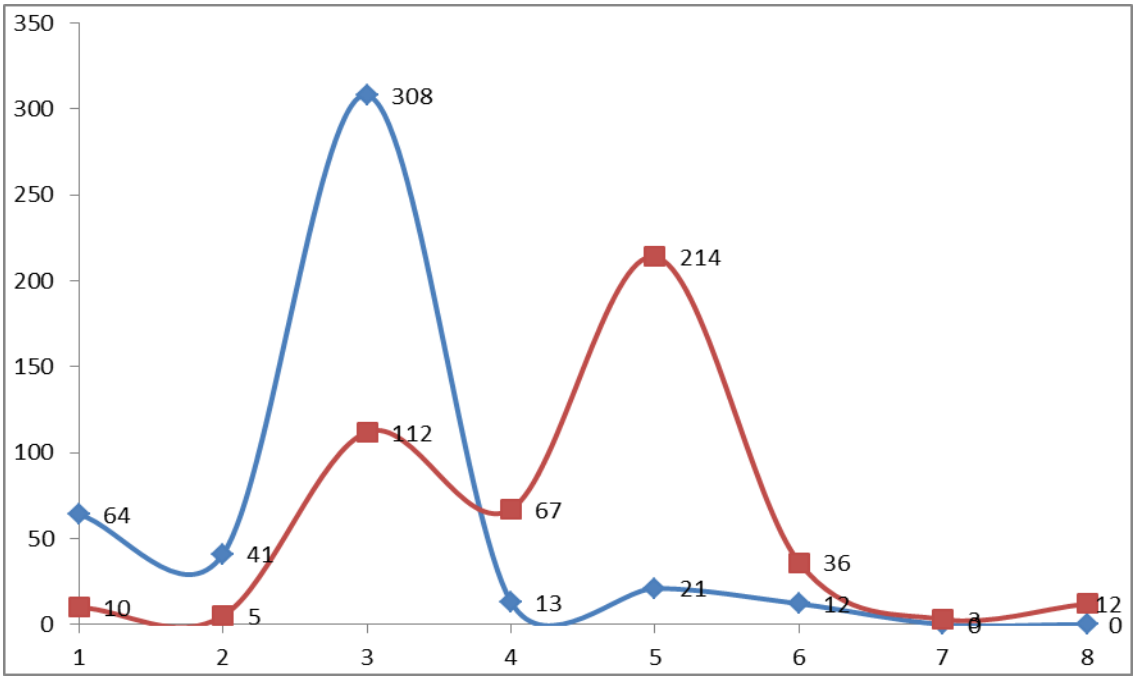

| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 64 | 43 | 160 | 10 | 22 | 10 | 0 | 0 |
| Phase Found | 8 | 6 | 56 | 52 | 162 | 17 | 3 | 5 |

Figure 3-7: Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Critical and Major Severity defects

| Defect Caused | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 0 | 0 | 0 |
| VER | | | | | 12 | 1 | 0 | 0 |
| INT | | | | 3 | 1 | 1 | 0 | 1 |
| IMP | | | 31 | 27 | 89 | 10 | 1 | 2 |
| DES | | 4 | 11 | 9 | 16 | 1 | 2 | 0 |
| REQ | 8 | 2 | 3 | 10 | 35 | 4 | 0 | 2 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |

**Defect found**

Figure 3-8: Defect classification map for the Baseline project. Critical and Major Severity defects

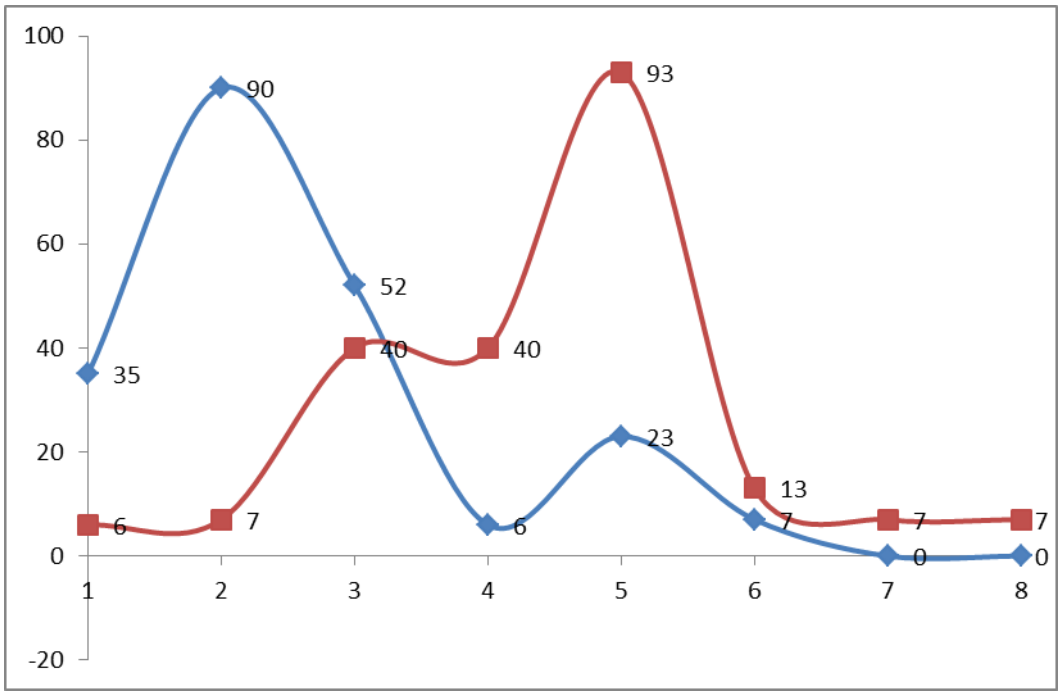



| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 64 | 41 | 308 | 13 | 21 | 12 | 0 | 0 |
| Phase Found | 10 | 5 | 112 | 67 | 214 | 36 | 3 | 12 |

Figure 3-9: : Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Defects in Software Components

| Defect Caused | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 2 | 0 | 0 |
| VER | | | | | 15 | 1 | 0 | 0 |
| INT | | | | 4 | 2 | 0 | 0 | 3 |
| IMP | | | 76 | 47 | 157 | 22 | 2 | 4 |
| DES | | 5 | 9 | 7 | 12 | 6 | 1 | 0 |
| REQ | 9 | 0 | 18 | 7 | 20 | 5 | 0 | 5 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
| | | | | Defect Found | | | | |

Figure 3-10: Defect classification map for the Baseline project. Defects in Software Components




| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 35 | 90 | 52 | 6 | 23 | 7 | 0 | 0 |
| Phase Found | 6 | 7 | 40 | 40 | 93 | 13 | 7 | 7 |

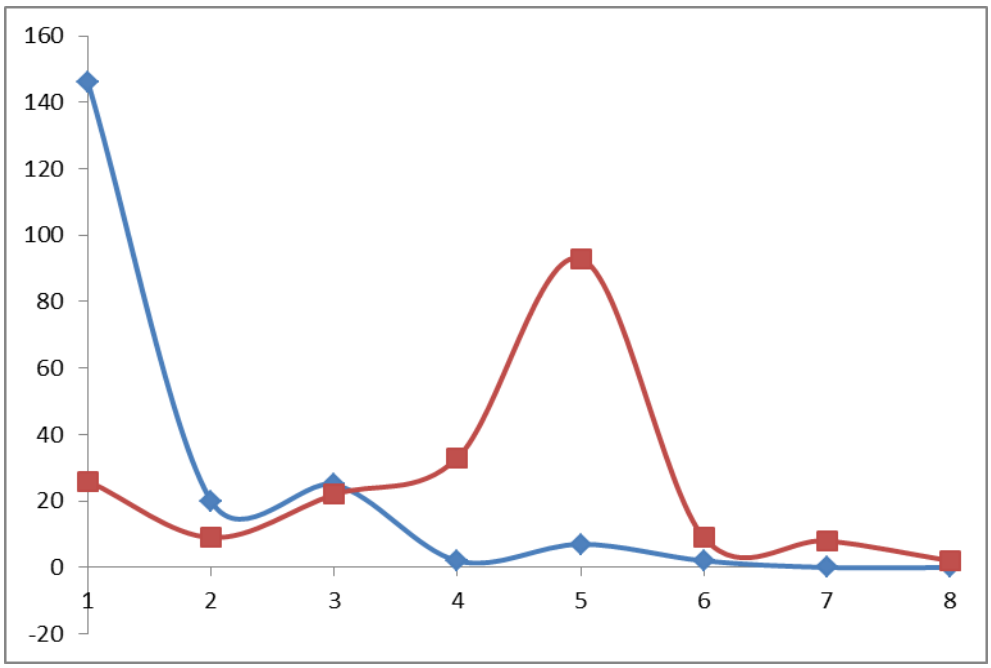
Figure 3-11: Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Defects in Hardware Components

| | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 1 | 0 | 0 |
| VER | | | | | 12 | 2 | 0 | 0 |
| INT | | | | 2 | 3 | 1 | 0 | 0 |
| IMP | | | 10 | 14 | 26 | 1 | 1 | 0 |
| DES | | 4 | 21 | 16 | 36 | 3 | 5 | 5 |
| REQ | 6 | 3 | 4 | 4 | 11 | 5 | 1 | 1 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |

Defect Caused

Defect Found

Figure 3-12: Defect classification map for the Baseline project. Defects in Hardware Components




| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |
|---|---|---|---|---|---|---|---|---|
| Phase Caused | 146 | 20 | 25 | 2 | 7 | 2 | 0 | 0 |
| Phase Found | 26 | 9 | 22 | 33 | 93 | 9 | 8 | 2 |

Figure 3-13: Phase Defect Caused versus Defect Found Defect distribution for Baseline project. Defects in components identified as Documentation

| | Reviewing | | | Testing | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| FIELD | | | | | | | | 0 |
| OPER | | | | | | | 0 | 0 |
| VAL | | | | | | 0 | 0 | 0 |
| VER | | | | | 2 | 0 | 1 | 0 |
| INT | | | | 0 | 1 | 0 | 0 | 0 |
| IMP | | | 4 | 2 | 13 | 3 | 2 | 0 |
| DES | | 4 | 5 | 2 | 5 | 1 | 2 | 0 |
| REQ | 24 | 5 | 11 | 24 | 72 | 5 | 3 | 2 |
| | REQ | DES | IMP | INT | VER | VAL | OPER | FIELD |

**Defect Caused**

**Defect Found**

Figure 3-14: Defect classification map for the Baseline project. Defects in components identified as Documentation

### 3.1.2 Analysis

The following list items describe the analysis of the most interesting results of the measurements we have presented in the previous subsection.

***Distributions***

- Superimposed, the Phase Caused and Phase Found curves represent a typical 'double hump' distribution. In most of the calculated distributions Phase Caused absolute maximum corresponds to the implementation phase. Phase Found has its absolute maximum at the verification phase.
- Taken separately, both curves have several local maximums; the second most significant one for Phase Caused is at the requirements phase. Phase Found has its second maximum at the implementation phase.
- PR (problem request) distributions behave a similar way as the general ones do, which is not surprising as problem requests contribute the lion share to the whole number of defects.
- CR (change request) Phase Caused distribution behaves differently: its maximum corresponds to the requirements stage. This is also understandable since change request are mostly about product feature enhancements needed to be introduced via requirements engineering.
- Critical and major defects have similar distributions as the general ones.
- The distributions for the software defects are similar to the general distributions with relatively higher share of defects found at the implementation phase with respect to ones found at the verification phase (ratios of found defects are 112/214=0.52 and 182/424=0.42 defects, correspondingly).
- The Phase Caused distribution for the hardware (mechanics and electronics) defects is different: it has its maximum at the design phase and the second maximum at the requirements phase.
- The defects recognised as different 'documentation defects' have the maximum at the requirements phase.

***KPI values***

Table 3-1 shows the results of the InR and AvD KPI measurements for the Baseline project.

| | Whole data | Problem type: PR | Problem type: CR | Critical&Major | Software | Hardware | Documents |
|---|---|---|---|---|---|---|---|
| InR | 33% | 31% | 43% | 29% | 20% | 37% | 62% |
| AvD | 2.02 | 1.97 | 2.27 | 1.96 | 1.72 | 2.09 | 2.75 |

Table 3-1: InR and AvD KPI values for the Baseline project for whole data and chosen fields.

- Percentage of defects in the red zone is 33%, that is, one third of defects are found too late in the development process.
- Average distance from the green zone is 2.02 (defects in average are found ~2 phases later.)
- The number of problem requests in the red zone (31%) is slightly lower in comparison to the general average.
- Critical and Major defects are discovered a little earlier as well (InR = 29%, AvD=1.96). However the difference is not that big.
- Software defects are in average found faster than hardware defects (InR=20% vs 37%; AvD=1.72 vs. 2.09 correspondingly)
- Change requests are issued relatively later in the development process.
- With respect to the software and hardware components, defects in documentation components are found relatively late: almost two thirds of them and in average almost 3 development phases later.

## 3.2 Defect management process KPI

The second set of KPIs related to the duration of the defect management process is described in subsection 2.2.3. This section describes the results of its measurements.

### 3.2.1 Defect management process mining

As the first step, using FRASR (filtering) and Disco (visualisation) tools, we performed process mining in order to confirm that defect management process in reality complies with the procedure prescribed in [ClearQuest, 2012] and shown in Figure 2-5. Figure 3-15 shows the defect management process as we discover it during process mining.

Comparison between Figure 2-5 and Figure 3-15 confirms that the real process in general follows the prescribed procedure, that is:

- Final states: *Validated, RejectAccepted, Forwarded* indeed do not have transitions back to the states corresponding to the *Open* defect category. *Duplicate* state is an allowed exception.
- All transitions present in Figure 3-15 are indeed allowed by the procedure (except just 2 cases with a defect going from the *Accepted* state to the state *Assigned* directly, bypassing the *Investigated* state.)

However, there are three differences distinguishing the real process from the procedure:

- *Postponed* state is not used. [ClearQuest, 2012] describes this state as follows “The Defect is delayed and a decision on handling the Defect will be taken in the future.” According to the Philips expert’s explanation, the policy in place was to avoid this state as much as possible and after investigation either to accept or reject the defect.
- Small fraction of cases (35) starts with the Investigated state bypassing state Submitted. Expert’s opinion on that was a human error: the investigated defect has been forwarded from another project not following the normal submission procedure.

- Every one of possible states has self-loop transitions during which the state of a defect doesn't change. Analysis of the defect management practice with the developers reveals that during those self-loops the person responsible at that moment for the defect handling quite often performs filling in multiple fields in the defect record. This activity is indicated in the field called "Action" as "Modify". In the *Forwarded* state, an undocumented action "UpdateFwdInfo" may also be performed.

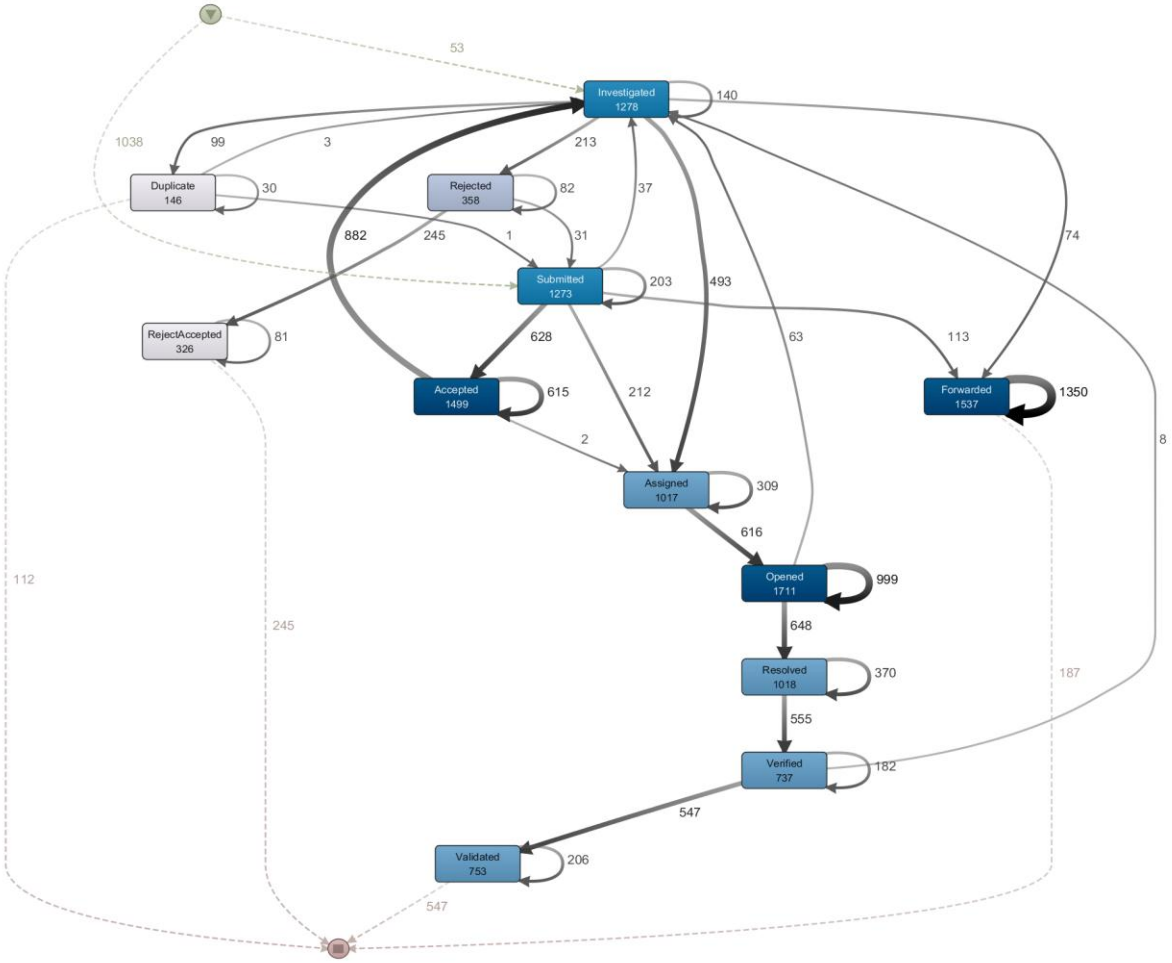



Figure 3-15: Defect management process discovered for the Baseline project. Frequency view of the Disco tool [Disco, 2014]: Nodes show states of a defect. Arrows indicate frequencies of the transitions between states.

- However, not all self-loops are caused by modification of the defect record. The defect very often stays in one of the final states when the undocumented action "Repair" takes place. No other fields are changed as a result of this action. An example if one of such cases is shown in Table 3-2.

| Action | Date | DefectId | CausedInPhase | FoundInPhase | State |
|---|---|---|---|---|---|
| Submit | 26-4-2010 8:25:55 | CVPRJ00015676 | | Integration | Submitted |
| Assign | 28-4-2010 8:49:10 | CVPRJ00015676 | Design | Integration | Assigned |
| Open | 13-7-2010 13:37:25 | CVPRJ00015676 | Design | Integration | Opened |
| Resolve | 13-7-2010 13:38:53 | CVPRJ00015676 | Design | Integration | Resolved |
| Verify | 30-7-2010 12:06:34 | CVPRJ00015676 | Design | Integration | Verified |
| Validate | 1-8-2010 17:17:43 | CVPRJ00015676 | Design | Integration | Validated |
| Repair | 6-11-2010 12:20:11 | CVPRJ00015676 | Design | Integration | Validated |

Table 3-2: An example of the case with Repair action in final state.

Further we performed filtering out of rarely occurring transitions. Figure 3-16 indicates the normal defect resolution process from *Submitted* to *Validated* as the most frequent path (thick arrows).

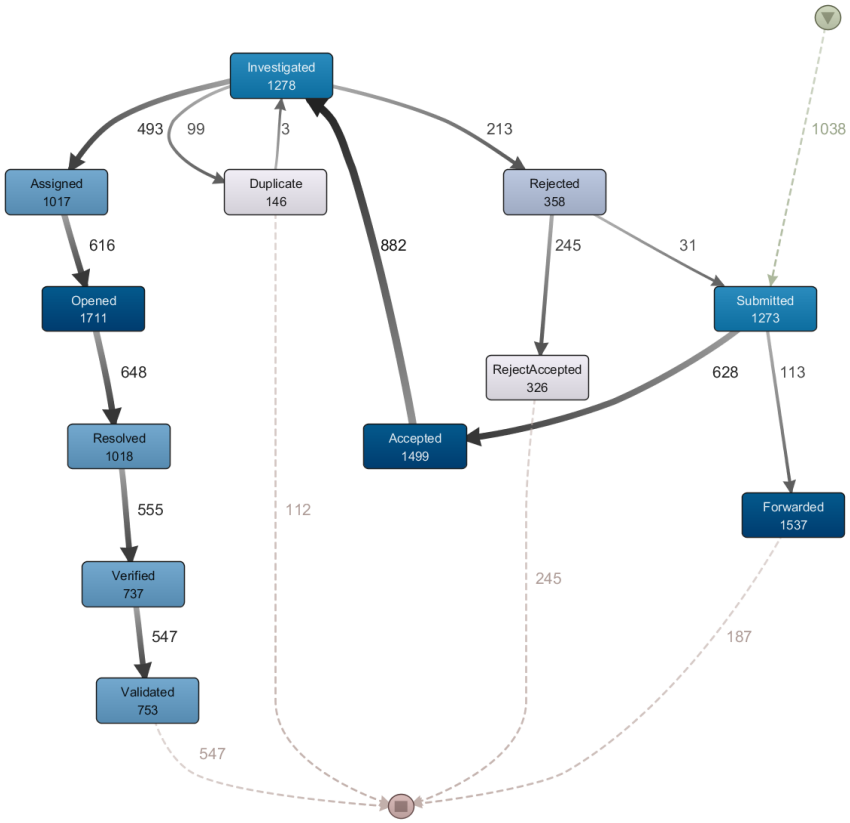



Figure 3-16: Defect management process discovered for the Baseline project. Less frequent paths are filtered out. Arrow thickness indicates the frequency of cases following the corresponding transition. The depth of a node colour indicates the frequency of the state occurrence.

### 3.2.2 Results of the KPI measurement

Figure 3-17 shows the 'performance view' on the same process as shown for case frequencies in Figure 3-15. The labels on the transitions show their median duration. The thickness of the arrows indicates the relative median duration of the corresponding transitions. Figure 3-18 shows case duration distribution for the Baseline project.

These views clearly indicate the following:

- The distribution in Figure 3-18 is highly right-skewed. This means that few very long cases would influence the values of the mean KPI *RTμ*. As a result, the median KPI *RT~* appears to be more representative.
- The self-loops at final states *RejectAccepted*, *Duplicate* and *Validated* are particularly lengthy. This is because of the described in 3.2.1 mass occurrence of the "Repair" action, which is performed after a long time when the defect was already in the final state (Table 2-1). According to the response of Philips experts: "Repair actions are performed by the support group and have no impact on the content of ClearQuest we are using. Due to this nature the action is not part of the specification. In principle it is wrong that this information is forwarded to Qlikview." For the state *Forwarded* this action occurs interchangeably with also undocumented action "UpdateFwdInfo". According to the experts, this action ''is used to update reference information (not the actual defect)", when the defect

is finalised and also can be filtered out. Thus, the decision has been made to filter out all the states from the Baseline project history database with those two actions.

- Comparison to the Figure 3-15 shows that the final state *Forwarded*, although not very long in time (median 3.4 days), occurs very often so that cases with a *Forwarded* final state tend to effect averages significantly. Investigation has shown that the duration of that state is affected both the above mentioned “Repair” action and also undocumented action called “UpdateFwdInfo”

Taking into account the above observations, we repeated the Defect Management Process discovery. The result is shown in Figure 3-19. Comparing with the Figure 3-17 we can see the following:

- The durations of the Open stage phases are comparable now with the ones for the Closed stage states
- The remaining lengthy self-loop is for the *RejectAccepted* state. This is due to the documented “ApproveReject” action performed by the defect owner. Thus, it is in the scope of the developers’ activities.
- The second remaining self-loop exists for the *Validated* state due to legitimate *CloneToDefect* action. It is out of the developer’s scope because it is performed by the Change Control Board (CCB).

Next we calculated the mean and median resolution time *RTμ and RT~* (subsection 2.2.3) for the whole project as well as for the problem request defect type. Further we separated the KPI values between defects still in *Open* states and *Closed* defects (Table 2-2). The latter we further separated into *Duplicate*, *Validated*, *Forwarded* and *RejectAccepted* states. Although it was an initial goal to aggregate open states into categories *New*, *Engineering* and *Verification*, is has proven to be impossible because too many transitions were directed (and allowed) backwards from the following to the previous states. Thus, is was possible that the particular defects might go through states in loops and the truncation of such defect life cycle paths would be misleading.

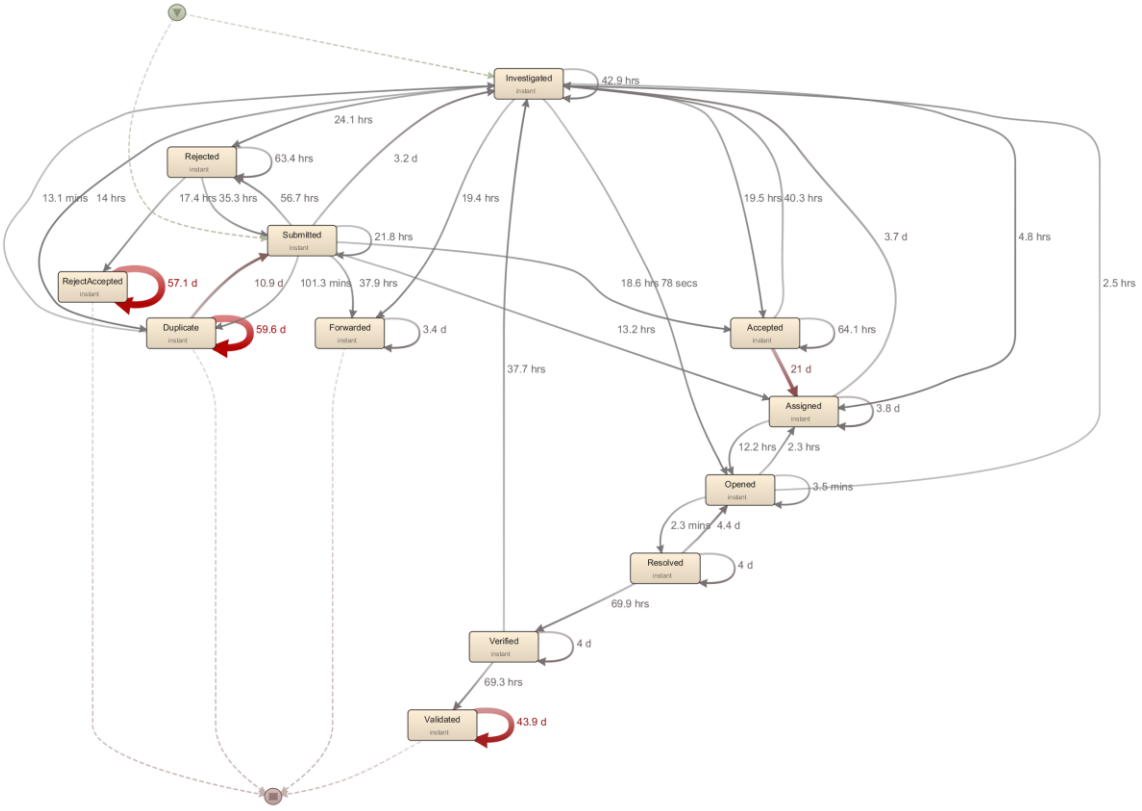



Figure 3-17: Defect management process discovered for the Baseline project. Performance view of the Disco tool: the transition labels indicate their median duration.

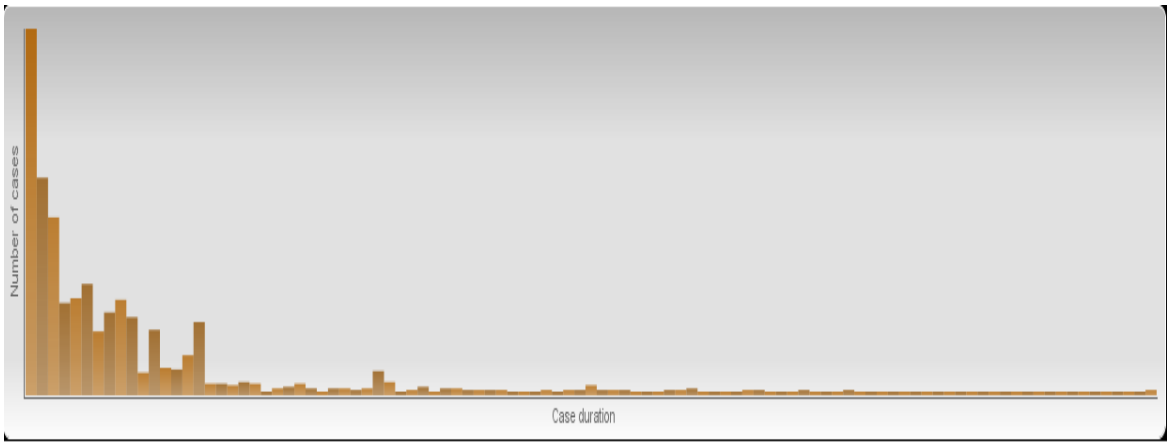



Figure 3-18: Case duration distribution for the Baseline project

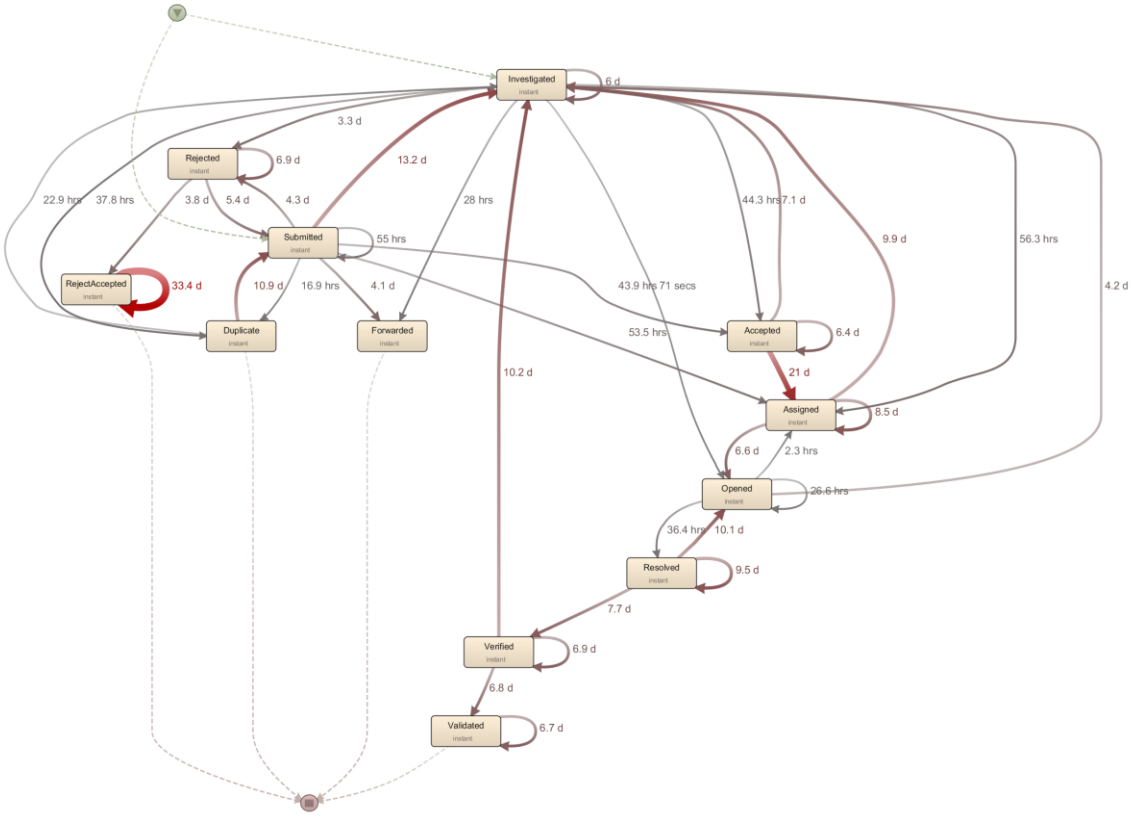



Figure 3-19: Defect management process discovered for the Baseline project. Performance view with "Repair" and "UpdateFwnInfo" states filtered out.

Table 3-3 shows the results of *RTμ* and *RT~* KPI calculations for the entire project including the KPI values for Critical and Major defects, Table 3-4 – for the PR defects only.

| 1091 | | Defect Flow Process KPIs | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Defect Open/ Closed | State | Critical&Major (370) | | | | For Open/Closed | | Total | |
| | | RTμ | | RT~ | | RTμ | RT~ | RTμ | RT~ |
| Open | Submitted | 24.8 | 30.9 | 9.9 | 17.3 | 33.4 | 15 | 38.7 | 21.9 |
| Open | Accepted | | | | | | | | |
| Open | Investigated | | | | | | | | |
| Open | Assigned | | | | | | | | |
| Open | Opened | | | | | | | | |
| Open | Verified | | | | | | | | |
| Open | Resolved | | | | | | | | |
| Open | Rejected | | | | | | | | |
| Closed | Duplicate (111) | 14.4 | | 4.9 | | 23.6 | 16.6 | | |
| Closed | Forwarded(188) | 8.3 | | 4.5 | | 10 | 5 | | |
| Closed | RejectAccepted(245) | 21.5 | | 13.1 | | 30.1 | 15.2 | | |
| Closed | Validated(547) | 49.2 | | 30.9 | | 55.5 | 36.5 | | |

Table 3-3: Calculation results of *RTμ and RT~* KPIs for the Baseline project.

| 904 | | Defect Flow Process KPIs | | | |
|---|---|---|---|---|---|
| Defect Open/ Closed | State | For Open/Closed | | Total | |
| | | RTμ | RT~ | RTμ | RT~ |
| Open | Submitted | 30.3 | 13.7 | 35.1 | 21 |
| Open | Accepted | | | | |
| Open | Investigated | | | | |
| Open | Assigned | | | | |
| Open | Opened | | | | |
| Open | Verified | | | | |
| Open | Resolved | | | | |
| Open | Rejected | | | | |
| Closed | Duplicate (96) | 21.4 | 14.4 | | |
| Closed | Forwarded(143) | 8.9 | 4.1 | | |
| Closed | RejectAccepted(196) | 26.2 | 15.1 | | |
| Closed | Validated(469) | 49.5 | 35.7 | | |

Table 3-4: Calculation results of *RTμ and RT~* KPIs for the Baseline project. PR-defects only.

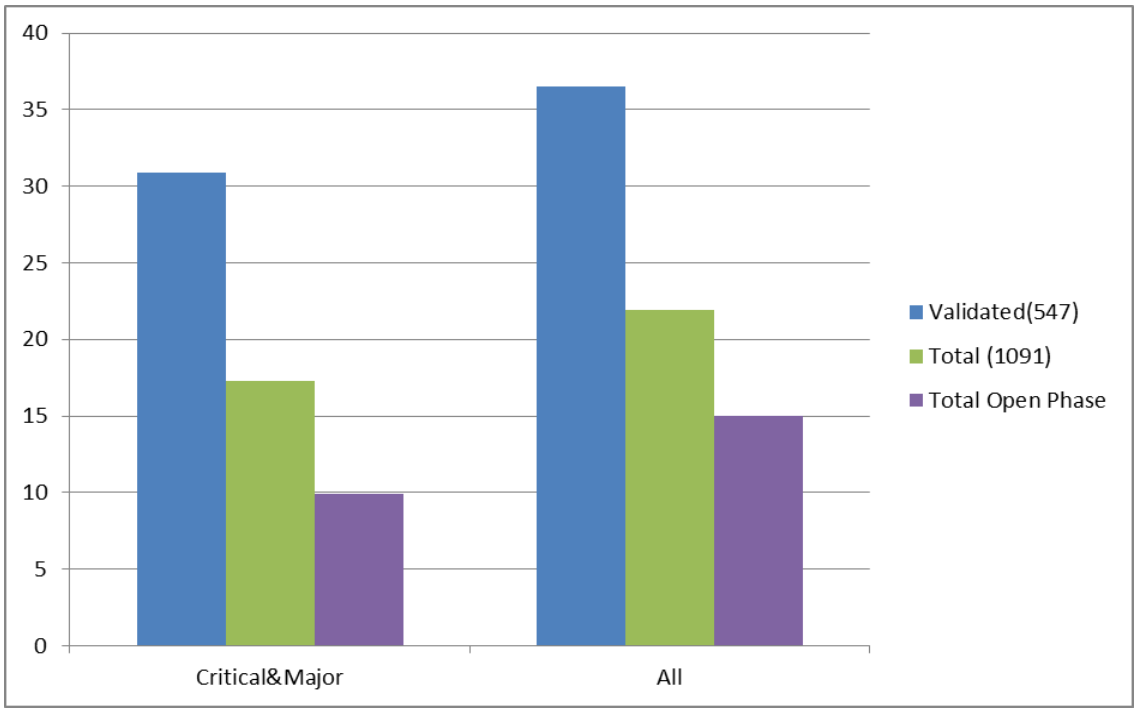



Figure 3-20 shows the comparison of the Total median resolution time RT~ KPI for all cases with the one for the cases of critical and major defects. Critical and major defects are resolved faster both in total time and validated defects. The open stage for them is also shorter.

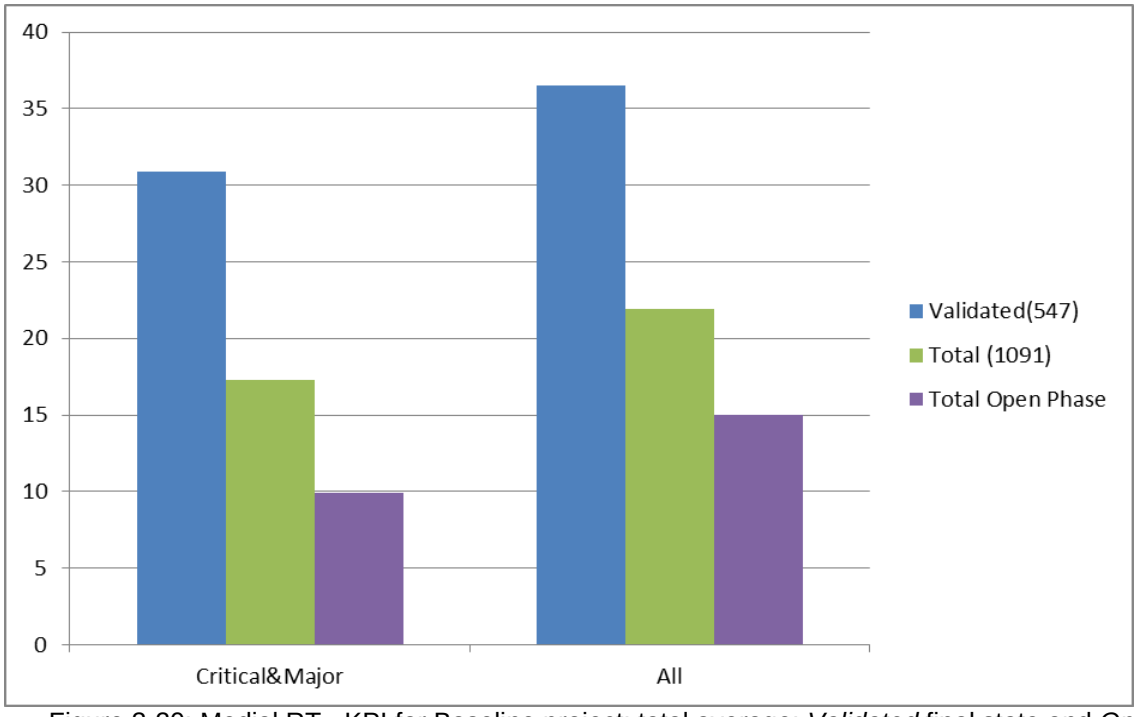



Figure 3-20: Medial RT~ KPI for Baseline project: total average; *Validated* final state and *Open* stage.

### 3.2.3 Analysis

The following list shows the results of the analysis for calculated above KPI values:

- Average median defect resolution time is about 22 days. Mean is 35 days.

- Problem requests are resolved roughly within the same time
- Critical and major defects are resolved relatively faster: 17 days on median average.
- Among defect transitions in defect Open phases, the transition from *Accepted* to *Assigned* is particularly long; median duration is 21 days (Figure 3-19). According to [ClearQuest, 2012] this action is performed by CCB – Change Control Board, that is, it is outside the scope of the development team
- On the other hand, the investigation phase, which is within the scope of the development team, also takes quit a long time: 9.9 days for the transition from the *Assigned* state and 6 days inside the state (Figure 3-19).

All in all, the duration of the transitions and states managed outside the development team (final state *Validated* in particular) or related to the organisation of the Defect Management Process (e.g., the transition from Accepted to *Assigned* and the *RejectAccepted* ) contribute significantly to the duration of the whole defect resolution process.

This means, that improvement of the Defect Management Process reflected by the median RT~ KPI is influenced both by advanced engineering techniques used to enhance actual defect resolution and improvement of the process organisation.

For the goals of the Crystal project, the former should be the focus of the KPIs.

The solution as to how to enhance this KPI might be to separate the two. For instance, we could use the throughput KPIs for partial traces, which can be improved (shorten) by engineering methods, e.g. from Open to *Resolved*, inside *Resolved*, and to *Verified.* However, it is difficult to separate them because of multiple back loops from cut out states (Figure 3-21).

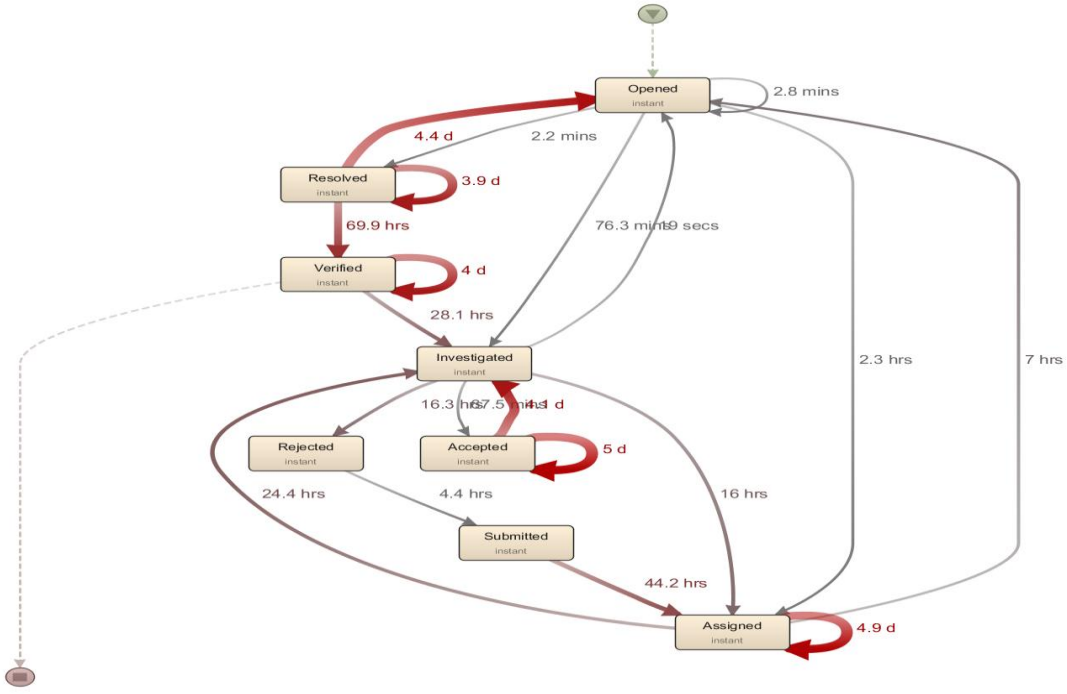



Figure 3-21: Defect traces truncated between *Opened* and *Verified* states.

Another possible modifications of this KPI may consider the number and median duration of rework transactions such as Verified -> Investigated or Assigned -> Investigated. The higher the quality of the defect management, the less rework is required.

To conclude, we advocate the usage of median throughput resolution time RT~ as a KPI for Defect Management Process is justified as improvement oriented. We expect the implementation of engineering methods described in sub section 2.2.2 to reduce the throughput time since early defect resolution is known to be beneficial as more defects could be resolved at early stages of the development process when their resolution would require less time.

Indeed, the KPIs and engineering methods described in 2.2.2 are aimed at closing the gap between Defect Caused and Defect Found curves by moving the latter towards the former. As a result, the defect resolution process could occur (much) earlier in the development process reducing the resolution throughput time reflected by the KPIs described in sub section 2.2.3 in this sub section.

# 4 Conclusions

In this report we described the first results of the evaluation of the product engineering process at Philips Healthcare we performed within the Crystal project. To perform the evaluation we defined two sets of key performance indicators: first, the Phases Caused vs. Found KPIs, and, second, the Defect Management Process KPIs.

We used the KPIs to perform ‘zero-measurements’ on the already finished Baseline project, which has been executed without applying engineering methods accelerating product development process.

The Phases Caused vs. Found KPIs measure the percentage of late found defects and the average distance between phases Caused and Found. The measurements have shown that about one third of the defects in the project were discovered too late, on average two product development phases later than they were introduced. Among different types of defects software defects are discovered relatively faster than hardware defects, critical and major defects are found slightly faster than in average. The highest difference between Phases Caused vs. Found is for change requests and defects in documentation.

Defect Management Process KPIs measure the mean and median duration of defect cases. The measurements for the baseline projects have revealed that the average median duration is 22 days. Critical and major defects are resolved about 1.3 times faster than in average. The analysis shows that the chosen KPI can be influenced both by the improvement in the engineering methods used to resolve the defects and the improvement of the process organisation. The degree to which these two can influence the KPI needs further investigation.

Later in the duration of the project we intend to compare the results of ‘zero-measurements’ with the measurements for the newer projects developed with the use of Agile and accelerating engineering methods.

# 5 Terms, Abbreviations and Definitions

Please add additional terms, abbreviations and definitions for your deliverable.

| | |
|---|---|
| CRYSTAL | **CR**itical S**YST**em Engineering **A**cce**L**eration |
| R | Report |
| PU | Public |
| PP | Restricted to other program participants (including the JU). |
| RE | Restricted to a group specified by the consortium (including the JU). |
| CO | Confidential, only for members of the consortium (including the JU). |
| WP | Work Package |
| SP | Subproject |
| KPI | Key Performance Indicator |
| PR | Problem request |
| FRASR | FRamework for Analyzing Software Repositories |
| CR | Change request |
| CCB | Change Control Board |
| REQ | Requirements engineering phase of the product development process |
| DES | Product design phase of the product development process |
| IMP | Implementation phase of the product development process |
| INT | Product integration phase of the product development process |
| VER | Subsystem verification phase of the product development process |
| VAL | System verification/validation phase of the product development process |
| OPER | Operations phase of the product development process |
| FIELD | Production phase of the product development process |

Table 5-1: Terms, Abbreviations and Definitions

# 6 References

Please add citations in this section.


| | |
|---|---|
| [Author, Year] | Authors; *Title*; Publication data (document reference) |
| [FRASR, 2011] | Wouter Poncin, Alexander Serebrenik, Mark van den Brand; *Process Mining Software Repositories*. CSMR 2011: 5-14 |
| [ProM, 2014] | Process mining. Website. http://www.processmining.org/. Last visited 15/04/2014 |
| [Disco, 2014] | Process mining for professionals. Fluxicon website: https://fluxicon.com/disco/. Last visited 15/04/2104 |
| [Crystal D_403_901, 2014] | Use Case Development Report UC403 Motion control of patient table and X-ray beam positioning |
| [CV X-Ray, 2012] | CV X-Ray; Measurement definition Prj_All_01 Defect Classification. Phases Caused vs Found; Philips medical Systems, Best, 2012 |
| [ClearQuest, 2012] | Ouweland J. v. d. ClearQuest For Philips Healthcare. PhilipsMedical sysytems, 2012. |
| [Kitchenham, 1996] | Kitchenham, B. (1996) Software Metrics: Measurement for Software Process Improvement (Cambridge, MA, Blackwell). |

# 7 Annexes

## 7.1 Annex I: Actions performed during Defect Management Process

| | Action name | Access Control | Owner | States | |
|---|---|---|---|---|---|
| | | | | Source | Destination |
| 1. | Submit | User | - | - | Submitted |
| 2. | Accept | CCB,<br>Lead Engineer | (Lead) Engineer | Submitted,<br>Investigated,<br>Postponed | Accepted |
| 3. | Investigate | (Lead) Engineer | (Lead) Engineer | Accepted | Investigated |
| 4. | Assign | CCB,<br>Lead Engineer | - | Submitted,<br>Investigated,<br>Postponed | Assigned |
| 5. | Open | (Lead) Engineer | (Lead) Engineer | Assigned | Opened |
| 6. | Resolve | (Lead) Engineer | Tester | Opened | Resolved |
| 7. | Verify | Tester | System Tester | Resolved | Verified |
| 8. | ReOpen | Tester | (Lead) Engineer | Resolved | Opened |
| 9. | Validate | System Tester | - | Verified | Validated |
| 10. | DecisionRequired | (Lead) Engineer | (Lead) Engineer | Assigned,<br>Opened | Investigated |
| | | System Tester | System Tester | Verified | |
| 11. | Duplicate | CCB | - | Submitted,<br>Investigated | Duplicate |
| 12. | Unduplicate | CCB | - | Duplicate | Submitted or<br>Investigated |
| 13. | Postpone | CCB | - | Submitted,<br>Investigated | Postponed |
| 14. | Reject | CCB | Submitter | Submitted,<br>Investigated,<br>Postponed | Rejected |
| 15. | ApproveReject | Owner | - | Rejected | Reject-<br>Accepted |
| 16. | RefuseReject | Owner | - | Rejected | Submitted |
| 17. | Forward | CCB | - | Submitted,<br>Investigated,<br>Postponed | Forwarded |
| 18. | Move | CCB,<br>Lead Engineer | - | Submitted,<br>Investigated,<br>Postponed | Submitted,<br>Investigated or<br>Postponed |