Release Metrics: When to Stop Testing with a Clear Conscience

John A Fodeh
Release Metrics: When to Stop Testing with a Clear Conscience

John A. Fodeh

John A. Fodeh graduated from the Technical University of Denmark in 1998 (MSc.). Despite his 3 years of experience he has been widely around in the field of testing, especially concerning test methodologies, process improvement and tools. He is currently a Test Manager at B-K Medical and is responsible for test-tools and methods. He has also been a key person in a SPI project concerning improved basis for the release decision. Mr. Fodeh has previously given presentations at a number of national conferences and at EuroSPI2000.
Release Metrics: When to Stop Testing With a Clear Conscience

John A. Fodeh
B-K Medical
Sandtoften 9, 2820 Gentofte
Denmark
Tel.: +45 45970174
Fax.: +45 45970199
jaf@bkmed.dk

And

Niels Bruun Svendsen
B-K Medical
Sandtoften 9, 2820 Gentofte
Denmark
Tel.: +45 45970157
Fax.: +45 45970199
nbs@bkmed.dk

Abstract
This paper presents the experiences gained and lessons learned from introducing metrics to support the release decision. When developing software systems, finding the right release moment is undoubtedly of highest importance. Finding this moment is a matter of being able to estimate the cost of release as well as the cost of postponing the release.

B-K Medical has applied a process improvement experiment with the purpose of basing the release decision on objective metrics rather than intuition and “gut-feel”. In this paper, we present this experiment and its outcome.

Introduction
When developing software systems the inevitable management question is: "When is the system ready for release?". On the bottom line, the answer to when to release a new product for production and sales is a matter of being able to determine the balance between time-to-market and quality. Do you ship now and win market shares or do you postpone the release for a higher quality product? In order to estimate the economical impact of releasing a product, the release decision must be based on quantitative arguments and consequences.

This paper will share the experiences gained and the lessons learned from introducing metric based release decision support.

For setting up a metrics program, the "Goal-Driven Software Measurement” method was applied. Apart from leading to a well-defined set of metrics, the impact on the organisation was remarkable. The metrics program resulted in a “Release Form”, i.e. a data sheet containing a set of metrics collected during the system test phase together with other relevant information needed for assessing the product's readiness for release.

A number of metrics included in the developed “Release Form” have been applied in multiple product releases and the results evaluated after each release. This paper will highlight the benefits found as well as the problems encountered. Furthermore, it will put emphasis on the experienced effect of introducing and working with these metrics, that has been seen in the organisation.

The work presented in this paper is a result of a Process Improvement Experiment (PIE) applied at B-K Medical and supported by the European Commission’s ESSI program. The full experiment is titled “WHEN” (project no. 27498). The PIE has three objectives, of which two are related to release decision support, the topic of this paper.
Company Context

B-K Medical develops, produces and markets ultrasound systems for medical diagnostic imaging. The systems are sold throughout the world with the major markets being Europe, USA and Asia. B-K Medical has 250 employees with 170 located in Denmark. The development department consists of 60 employees where 20 are involved in software development. B-K Medical is ISO 9001 certified and most of the products have FDA market clearance and conform to the CE-Medical Device Directive. Therefore, internal as well as external audits are performed accordingly. No formal assessment against a model has been performed. Nevertheless, an informal self-assessment using the BootCheck tool from ESI has been performed. This assessment gave maturity ratings between 2.5 and 3.25, indicating some areas in need of improvement to get to the Defined (3) level, and a general lack of metrics as required in the Managed (4) level.

Project Initiation

The work on release decision metrics was initiated by the management group. It has its basis in the release meetings held within the management group, where the decision is made whether to release a new product or not. This decision was found to be based too much on subjective input and too little on objective data. Therefore, a goal was set to improve the basis for the release decision and thereby give objective support on when to release. A brainstorm on needed data and information to give an improved support for the release decision was held with the management group. The brainstorm was used as input for the work to be done and to get the attention and support of the management group on the fact that the improvement work was started and that it was their problems and wishes being addressed.

In order to measure the impact of the improvement, a questionnaire was designed based on the output of the management brainstorm. The questionnaire was applied after the release of three products, with one release being prior to the PIE, one being half way through and one being at the end of the improvement work.

The questionnaire consists of the following 7 questions:

1. How would you characterise the basis for release decision in general?
2. How was the remaining known errors and their consequences presented?
3. How was the presentation of how much that had been tested?
4. How was the presentation of how thorough the user evaluation was?
5. How was the estimate on remaining unknown errors?
6. How was the estimate on remaining unknown safety errors?
7. How was the post-release plan presented?

Each question is given a score. The five rating levels are:

1. Non existent
2. Weak (Very subjective)
3. Fair (Subjective, but well argumented)
4. Very Good (Mainly objective)
5. That’s how to do it (Objective, based on solid data)

The initial average score was 2.4, indicating that the basis for release decision was largely subjective. The results obtained during the WHEN project can be seen in The Results section.
Setting up a Metrics Program

At the time of the WHEN project definition a number of metrics were identified. In addition, the brainstorm with the management group gave input to additional measurements that could be applied. It was decided to test if the planned metrics and techniques were optimal for supporting the objectives of the experiment. The “Goal-Driven Software Measurement” method was selected to set up the relation between the objectives and the measurements to be implemented. This handbook, developed by SEI (Software Engineering Institute) at Carnegie Mellon University, ref. [2], delivers a formalised and structured method for decomposing the Business Goals of the PIE into a set of metrics and clarifying the dependencies between the metrics as well as the actions needed for collecting them.

The method builds on the GQM (Goal Question Metric) method by Basili and Rombach, and extends the GQM with a phase that guides the user from Business Goals through Sub-Goals to Measurement Goals. In overview, the method can be illustrated as shown in Figure 1.

![Figure 1: Goal-Driven Software Measurement method](image)

Figure 1: Goal-Driven Software Measurement method

The work on Goal-Driven Software Measurement was conducted as a series of workshops involving the newly formed system test group and an external mentor. The system test group consisted of a senior test manager, a senior SW engineer and two test managers. The first step on the way from Business goals to Sub-Goal was to ask questions concerning the process involved. As shown in Figure 2, this raised another question, i.e. which process are we talking about? It was found that despite the fact that work is performed according to an ISO9000 certified quality system, the process definitions were either lacking or not detailed enough and the terms used were not defined. The “Goal-Driven Software Measurement” handbook uses a concept called “Mental Models”. Mental Models are the perception of procedures, processes and practices in the mind of the user. Models like that can work when only one person is using the model (although the model has a tendency to change according to the current situation). The problem occurs when more people are involved and only Mental Models exist, because there is at least one Mental Model per person involved.
Ask questions to the process involved

Which Process?

Mental Model

Defined Process

**Figure 2: Evolving Mental Model to process definitions**

Evolving those Mental Models and writing them down in process definitions took quite some time. However, the discussions afterwards could be aimed at continuing the Goal-Driven Software Measurement process, instead of discussing the proper use of terms and which sub-processes existed.

Having reached the point where a number of Sub-Goals were defined we were ready to apply the GQ(i)M part of the process. The (i) part stands for indicator and is an addition to the GQM that we found valuable. The idea is to make sketches of the desired presentation of the measurement. It provides context, gives “life” to the measurements and generates a number of additional discussions and ideas. Examples of indicators can be seen in Figure 3.

Several measurements were defined using the Goal-Driven Software Measurement method. A number of these were selected as our release decision metrics. It was noticeable that some Sub-Goals did not directly result in measurements, but rather pointed out the need for templates, checklists etc.

**Figure 3: Use of indicators to provide context**
The final step was to prepare a plan that addressed the identified actions needed for both implementing the measures and completing the templates and checklists. This plan set the framework for the PIE activities and established a reference for further improvements of the processes.

The Release Decision Metrics

The selected release decision metrics can be grouped into the following 4 main groups:

- **System Status**
  - The number of known remaining errors and their consequences
  - System stability data
- **Estimate on remaining errors**
  - Altogether and sorted according to severity
- **Test Coverage**
  - Test execution coverage
  - User evaluation coverage
  - Code coverage
- **Post release plan**

Based on these, a Release Template was developed (implemented as an MS Word template). When filled, the template is referred to as a “Release Form”.

The Release Template

In the following, the developed Release Template is shown (Figure 4) together with an explanation of its contents. The release template is basically a data-sheet containing the gathered metrics regarding the state of the system to be released. The data sheet is usually delivered to the management group some hours, or the day before the actual release meeting, so the contents of the sheet can be studied in advance.

In the template, the actual value of each metric is shown together with a target value and a reference value (i.e. the actual value from a former release of the same or another product). At this time, targets have only been set for a few of the metrics. It is planned to add further target values as we obtain the data to base the target values on.

In addition to the metrics table, the release template includes two charts. One showing the stability of the system for each of the builds during the system test phase (Figure 5) and the other showing the error trend based on the error detection rate (Figure 6). The error trend has been one of the key metrics introduced and will be described further in the Error Trending section.

Additional information to the template is normally given through footnotes.

**Test Coverage**

The test coverage data include the information concerning the progress and completeness of the testing. A low value reveals insufficient testing effort and the risk of potential latent defects. It is planned to extend this section with code coverage data for quantifying the portion of the code that is exercised by testing, thereby showing the thoroughness of the applied testing techniques. If there were any requirements not covered by the planned tests an explanation is given in a footnote.

**System Stability**

The system stability section delivers vital information about the reliability of the software. The data shows the mean number of operations between failures, equivalent to the widely spread Mean Time Between Failures (MTBF) metric. This measure is collected by means of a software tool. The task of this tool is to exercise the target system with random input and record the
A reference is given to the stability chart giving a graphical presentation of the mean number of random operations between failures, as a function of the build number. The chart visualises the progress in the system's stability throughout the development and up to release. The chart contains two limits; the lower limit is the entry criterion for system testing, while the higher limit is release stability criterion. In this way, it is straightforward to confirm if the system's stability is adequate for release.

**System Status**

Test system status section contains statistics regarding the problems reported during the system-testing phase. The total number of problems reported is shown and categorised in closed (fixed and verified) and open problems. The open problems are sorted according to their severity. These data deliver a snapshot of the system state at release time, making it possible to take into account the risk and consequence of releasing the system. E.g. if the data reveals a large number of open high-severity or non-verified problems, then it clearly shows that releasing the system at this moment is high-risk decision. The metrics of this section are collected by applying the appropriate queries in the database for problem reports.

**User Evaluation**

User feedback during the development is undoubtedly of major importance. The user evaluation section presents relevant data collected during the user evaluation activities. At this time, this section only contains a summary of the raised problem reports and their classification. It is planned to extend this section with information about covered applications, user types, countries, etc.

**Post Release Plan**

The post release plan section contains an overview of the activities to be performed after the release of the product together with the responsibilities, schedules and the date for the subsequent release. The post release plan makes it possible to speed-up the detection and correction of post release errors, thereby reducing the number of systems requiring a filed update. Furthermore, the post release plan sends a clear signal that the project is not ended with the release of the product. This helps preventing management from allocating all resources to new projects just after release. Instead, efficient planning in the transition phase between projects can be made.

The post release plan is prepared by the project manager in collaboration with the test manager involved.

---

**System Test Release Form – 12 May 2000**

<table>
<thead>
<tr>
<th>Ultrasound Scanner 4101 version 1.00 (Build 20)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
</tr>
<tr>
<td>Test Coverage</td>
</tr>
<tr>
<td>Percentage of executed vs. planned test scripts</td>
</tr>
</tbody>
</table>

| System Stability (Figure 5) | |
|-----------------------------|
| Mean number of operations between failures | 104600 | 80000 | 78650 |

| System Status 12-May-2000, 09:45 | |
|-------------------------------|
| Number of raised problem reports | 206 | 403 |
| Number of closed problems | 166 | 239 |
| Number of problem corrections to be verified | 3 | 0 | 54 |
Number of known problems in Build 20

<table>
<thead>
<tr>
<th></th>
<th>37</th>
<th>0</th>
<th>110</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of <strong>safety</strong> errors</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of <strong>functional</strong> problems</td>
<td>11</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Number of <strong>discrepancies</strong></td>
<td>19</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Number of <strong>change requests</strong></td>
<td>7</td>
<td>48</td>
<td></td>
</tr>
</tbody>
</table>

**Error Trend** (Figure 6)

| Estimated total number of problems after release (first ½ year) | 8  | 5  | 30 |
| Estimated number of **safety** errors | 0.4 | 0  | 0.6 |

**User Evaluation**

| Number of raised problem reports | 22 | 56 |
| Number of **safety** errors | 0  | 0  |
| Number of **functional** problems | 6  | 14 |
| Number of **discrepancies** | 5  | 14 |
| Number of **change requests** | 11 | 28 |

**Post Release Plan**

The following activities/issues will be addressed in the next release in the following priority:

1. Addition of Russian language.
2. Improved calculation package.
3. User defined TGC settings
4. Export to PC facility
5. Correction of the following recognised errors:
   - PR1638: Corrupted images on floppy
   - PR1634: Ob/Gyn report - values missing in report.
   - PR1481: Error in Italian text-strings concerning calculations.
   - PR1388: TGC-sliders do not work when zooming in.
6. Verification of implemented problem reports

**Allocated resources**

- Software: CAW+TOJ + JR
- HW: JPB + OMS + HED + LLH + MON
- Mech: N/A.
- System Test: MOH + JAF + SEM + JRO
- Application Specialist: BF

**Planned date of 2'end release**

23-jun-2000

**Planned date of 3'end release**

28-sep-2000

**Test Personal**

| Test Manager | MOH |
| Test Technician | CAW |
| Test Technician | JAF |
| Test Technician | LKA |
| Test Technician | MOH |
| Test Technician | NiN |

**Application Specialist’s and User Evaluation**

| Application Specialist | BF |
| Application Specialist | GK |
Figure 5: Stability Trend

Error Trend for 4101 (since build 4)

- Estimated Remaining Errors: 8
- Estimated Remaining Days: 5
- Uncertainty (standard deviation) ±: ± 2.08

Cumulative Errors

<table>
<thead>
<tr>
<th>Effort (Test Days)</th>
<th>Fitted Trend</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Error Trend

Error Trending

For estimating the number of remaining unknown errors, error trending is used. Error trending is based on the graph showing the accumulated number of reported errors (y-axis) as a function of test effort (x-axis), as shown in Figure 6. The test effort is expressed in terms of test days. A test day is the effort equivalent to a typical (8 hours) work day of a single tester. The test effort data is gathered by means of the work-hour registration system, while the data concerning the found errors are gathered by searching through the problem report database.

The dots in the graph represent the reported errors, while the line going through the dots is the best-fit line (mathematical least square) based on a Weibull function. This line is extrapolated, providing a predictive evolution of the error finding rate.
As noticeable, the graph is S-shaped and can be divided into three sections; the first is the section with the slight slope at the beginning, the second is the mid-section with the linear-like slope, the third is the section where the graph flattens out. This S-shape is found to correlate with empirical data from software projects. At system test initiation, the error finding rate is low (as the functionality of the software is often restricted to few areas). The error finding rate increases with the addition of new functionality and the introduction of new errors during the correction of already found errors. Entering the third section, the error finding rate begins to decrease, as it becomes harder to find new errors. Ultimately, the graph flattens. Finding further errors at this stage require huge test effort and shows that the software is possibly ready for release (or at least that the limitation of the applied testing technique has been reached).

More details on the error trending can be found in ref. [1] and the results obtained by using it are discussed in The Results section.

**The Results**

Measurable results have been obtained on the quality of the release decision support. Equally, on the precision of the Error trend based estimate of the number of remaining unknown errors. The quality of the release decision support has been measured by means of the previously mentioned questionnaire. Figure 7 shows how the rating of each of the 7 questions has evolved through the 3 releases. Included is also an average of the 7 questions for each of the 3 releases. It is seen that the average score has increased from 2.4 to 3.6. With the level definitions in mind, this means that the basis for release decision has shifted from largely subjective to mainly objective.

![Release Decision Support](image)

**Figure 7: Result of Release Decision questionnaire**

One of the major improvements is the estimate on remaining unknown errors. This estimate is based on the error trend. The results of the error trend based estimates compared with actual number of errors found can be seen in the table below. What we conclude from this, is that the error trend based estimate is an optimistic estimate. It is not high precision, but it is fairly
consistent and far more realistic than a subjective estimate. The experience is that the error trend based estimate is nearly always received as being high, i.e. “Do we really have that many errors left?”. In that case, it is important to notice that so far the estimate always has been too low.

<table>
<thead>
<tr>
<th></th>
<th>Trend based estimate</th>
<th>Reported after release¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-WHEN release</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>2100 product 1 - release</td>
<td>17</td>
<td>35</td>
</tr>
<tr>
<td>2100 product 2 - release</td>
<td>31</td>
<td>56</td>
</tr>
</tbody>
</table>

¹All reports counts, including change requests.

**Lessons Learned**

*Metrics are valuable in planning and decision making*

Clearly, there is a substantial pressure to maximise the profit by releasing early. On the other hand, the economical losses of releasing a poor-quality product as well as the damages in goodwill and reputation may inflect irreversible damage. In the lack of metrics to support the release decision, the state of the system to be released is vague, often resulting in an unnecessary delay of the release.

In this respect, the metrics used for supporting the release decision have shown their value. By giving the management group a more objective release decision basis, a higher degree of freedom in their decision has been obtained. A visible effect has been that management has decided not to delay releases in order to reduce the number of unknown defects at time of release, but to focus on a post-release plan to bring down the impact of post-release errors (See The Release Template section).

In the planning phase, the metrics have also shown their strength. The ability to give a qualified guess on the effort size of a system test project 9 months ahead, by use of the SW development time to system test time ratio, is convincing. During the system test, the error trend has given input to the planning of the remaining amount of test and needed resources for both testing and error correction.

Furthermore, metrics have also taken the role of a common reference. Especially the stability and error trends gave the common reference for discussion of system state, i.e. a simple graph gives the common basis for discussing system state, which is understood and accepted by top-management, project management, developers, testers and QA staff.

*Metrics demand maturity or the will to mature*

Working with defining relevant metrics we soon discovered that there was a need for clear definitions of the processes to base the measures on. In other words, for the metrics to be relevant a certain level of maturity is required. We did not initially have that level of maturity but we used the work on metrics to trigger and drive the improvements of process definitions. We experienced major benefits from that work especially in terms of job motivation, as there is no longer any need for spending time on the general way of performing regular routines, instead more effort can be put into solving the specific task at hand. Moreover, when spending time on the process it is to improve it, instead on figuring what the process is.

As much of the work done was focused on the system test phase, the major impact has been seen in the system test group. The results obtained as well as the discussion generated during the PIE has helped greatly in forming a dynamic and committed group that considers metric-supported process improvement a vital part of the process.

**Conclusions**

Incorporating metrics into the development process delivers an effective tool for planning, monitoring, predicting and following up. In particular for finding the proper timing for release, possessing the right metrics has shown a tremendous value. These metrics provide insight into
the state and quality of the software system making it possible to base the release decision on solid data and well-calculated risk rather than intuition and gut-feel.

The conclusion on the use of Goal-Driven Software Measurement to drive the definition of a metrics program is that it can be highly recommended. Although it involved far more work than initially anticipated, it was undoubtedly worthwhile. Looking back, it was a necessary step for bringing up the level of maturity to where measurements start to make sense. Starting out without the awareness of missing process definition etc., the Goal-Driven Software Measurement was a perfect trigger of the needed improvement actions. Especially in the system test group, the work completed with Goal-Driven Software Measurement had helped establishing a solid infrastructure consisting of well-defined, functional and efficient processes.

In relation to the release decision support, a clear and positive effect has been seen. The greatest positive effect has been seen for the error trend based estimation of number of remaining errors and for the post release plan. These improvements have also triggered an interest in other metrics based on available data. An example of this is the calculation of the general cost of delaying release and comparing that with the cost of field update of the SW. It showed that the cost of updating the SW on all scanners in the field, 6 months after release, equals the loss of delaying the release by only 10 days. The substantial cost of delaying the release shows the enormous pressure to release early and emphasises the importance of choosing the right release time, as the consequences of a "premature" release may be unrecoverable. The developed release template will without doubt be used on future releases. It will be enhanced with code coverage and an improved user feedback section. It will also evolve towards defining release criteria by defining more target values.

In a broader sense, this work has helped establishing process improvement as a natural part of daily life in the development department.

References
Release Metrics

WHEN to stop testing with a clear conscience

John A. Fodeh
B-K Medical A/S
www.bkmed.com/when
email: jaf@bkmed.dk or nbs@bkmed.dk

This work was supported by the EU Commission’s ESSI program as part of project 27498 - WHEN

© 2001 B-K Medical A/S
Contents

Background
Setting up a Metrics Program
Release Decision Metrics
Lessons Learned
Background

Wish to Do
Improved basis for release decision

ESSI Program

WHEN
PIE - 27498

Need to Do
Higher complexity products
New FDA guidelines
Shorter time-to-market
The Release Decision

• When is a new product ready for release to production and sales?

• Risk evaluation based on:
  – Many sources
  – Complex information
  – Often subjective
The Basis for Release Decision

• Evaluation questionnaire
  – 7 questions based on brain-storm
  – 5 levels:
    1. Non existent
    2. Weak (very subjective)
    3. Fair (subjective, but well argumented)
    4. Very good (mainly objective)
    5. That’s how to do it (objective, based on solid data)

• Evaluation after each release
  – Initial evaluation gave score 2.4
Setting up a Metric Program

Goal-Driven Software Measurement - A Guidebook (Park, Goethert, Florac 1996)

- Structured and formalized method.
- 10 steps guide you from business goals to measurements
- Based on ideas of Basili and Rombach
- Templates, examples and checklists


© 2001 B-K Medical A/S
Goal-Driven SW Measurement

From business goals to sub-goals

Ask questions to the process involved

Which Process?

Evolving mental models provide context and focus

Mental Model

Defined Process
Goal-Driven SW Measurement
From sub-goals to measurements

GQ(I)M translates informal goals into executable measurement structures
Setting up a Metrics Program

Conclusions:

- Logical mapping of business goals to metrics
- Effective for assessing where you are ("if you don’t know where you are a map won’t help" - Humphrey)
- Highlights shortcomings in applied processes
- A reference for further improvements
- Understanding and evolving Mental Models required a demanding effort
- Large amount of data was quickly generated
- Far more work than anticipated
Release Decision Metrics

- **System Status**
  - Known remaining errors and their consequences
  - System stability

- **Estimate on Remaining Errors**
  - Altogether and sorted according to severity

- **Test Coverage**
  - Test execution coverage
  - User evaluation coverage
  - Code coverage

- **Post Release plan**
Release Form

• **Metrics**
  – Data gathered during system test
  – Snapshot of the system state
  – Reliable estimates

• **Post release plan**
  – Well calculated risk
  – Enable sooner release
  – Effective transition between projects
  – Prepare organisation
Demo

Now you are the project manager!
# Release Form

## System Test Release Form - 21 Nov. 2001

### System 4101 version 1.0 (Build 19)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Target</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage of executed vs. planned scripts</td>
<td>84%</td>
<td>100%</td>
<td>66%</td>
</tr>
<tr>
<td>Code coverage</td>
<td>76%</td>
<td>&gt; 95%</td>
<td>80%</td>
</tr>
<tr>
<td>System Stability (see stability chart)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean number of random operations between failures</td>
<td>43447</td>
<td>80000</td>
<td>78400</td>
</tr>
</tbody>
</table>

## System Status - 21 Nov. 2001 - 09:45

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of raised problem reports</td>
<td>603</td>
<td></td>
<td>761</td>
</tr>
<tr>
<td>Number of closed PRs</td>
<td>291</td>
<td></td>
<td>578</td>
</tr>
<tr>
<td>Number of corrected problems to be verified</td>
<td>108</td>
<td>&lt; 10</td>
<td>72</td>
</tr>
<tr>
<td>Number of known open problems:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of critical problems</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Number of functional problems</td>
<td>78</td>
<td>&lt; 5</td>
<td>20</td>
</tr>
<tr>
<td>Number of discrepancies</td>
<td>87</td>
<td>&lt; 10</td>
<td>42</td>
</tr>
<tr>
<td>Number of change requests</td>
<td>38</td>
<td>&lt; 10</td>
<td>48</td>
</tr>
</tbody>
</table>
## Error Trend
(see error trend chart)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Target</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated post release problems (total in first ½ year)</td>
<td>30</td>
<td>&lt; 6</td>
<td>17 (35)</td>
</tr>
<tr>
<td>Estimated critical post release problems</td>
<td>0.6</td>
<td>0</td>
<td>0.4 (1)</td>
</tr>
</tbody>
</table>

## User Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
<th>Target</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of raised problem reports</td>
<td>54</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>Number of critical problems</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of functional problems</td>
<td>28</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Number of discrepancies</td>
<td>5</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Number of change requests</td>
<td>21</td>
<td>28</td>
<td>28</td>
</tr>
</tbody>
</table>
# Post Release Plan

The following issues will be addressed:

1. Addition of Russian language
2. Improved calculation package
3. User defined TGC settings
4. Export to PC utility

5. Correction of the following PRs:
   - PR 1638: Corrupted images with high compression.
   - PR 1634: Missing OB/Gyn values in report.
   - PR 1481: Errors in Italian text-strings in setup page.
   - PR 1388: TGC-sliders do not function while zooming in.

6. Verification of all implemented PRs

## Allocated resources:

- **Software:** CAW + TOJ + JR + KED
- **Hardware:** JPB + OMS + HED + MON
- **Mechanics:** N/A.
- **System Test:** MOH + JAF + SEM + JRO
- **Application specialist:** BF

## Planned dates

<table>
<thead>
<tr>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned date of 2’end release</td>
<td>13-Dec-01</td>
</tr>
<tr>
<td>Planned date of 3’erd release</td>
<td>09-Jan-02</td>
</tr>
</tbody>
</table>
Release Form - Stability Trend

Stability Trend for 4101

- Release Criterion
- Entry Criterion

Mean number of random operations between Failures

Build no.

- Random operations
- Start Criterion
- Release Stability Criterion

© 2001 B-K Medical A/S
Release Form - Error Trend

Error Trend for 4101

Estimated Remaining Errors: 30
Estimated Remaining Days: 71 (until 5 errors remaining)
Uncertainty (standard deviation) is: +/- 3.8
Demo

Release or not???
Results

Release Decision Support

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1: How would you characterize the basis for release decision in general</td>
<td></td>
</tr>
<tr>
<td>Q2: How was the remaining known errors and their consequences presented</td>
<td></td>
</tr>
<tr>
<td>Q3: How was the presentation of how much that had been tested</td>
<td></td>
</tr>
<tr>
<td>Q4: How was the presentation of how thorough the user evaluation was</td>
<td></td>
</tr>
<tr>
<td>Q5: How was the estimate on remaining unknown errors</td>
<td></td>
</tr>
<tr>
<td>Q6: How was the estimate on remaining unknown safety errors</td>
<td></td>
</tr>
<tr>
<td>Q7: How was a post-release plan presented (regarding further testing and further releases)</td>
<td></td>
</tr>
</tbody>
</table>

© 2001 B-K Medical A/S
Lessons Learned

• **Metrics are valuable for planning and decision making**
  – Easy to communicate
  – Deliver solid arguments
  – Insight into the state and quality of the system
  – Effective tool for planning, monitoring, predicting and following-up
  – A common frame of reference

• **Metrics demand maturity (or the will to mature)**
  – Require well-defined processes
  – Require a supporting infrastructure
Release Metrics

WHEN to stop testing with a clear conscience

John A. Fodeh
B-K Medical A/S
www.bkmed.com/when
email: jaf@bkmed.dk or nbs@bkmed.dk

Thanks for listening!