PREDICTING SOFTWARE LAUNCH READINESS IN A COMPLEX PRODUCT

Abhinav Sharma

HCL Technologies, Welwyn Garden City, UK

ABSTRACT

A simple model used successfully for estimating and tracking software defects to predict launch readiness of software in a complex product is described in this paper. The model is based on tracking the number of defects estimated to be found, actually found and resolved to measure the quality of the product. Defect estimates can also help identify quality and process issues in the development and testing phases.

The defect estimation tracking method described here covers the whole project and is split into the three phases Initial Defect Estimates (based on historical data), Interim Revised Estimates (based on actual performance of the project) and Final Defect Tracking (based on testing still to do). The method is based on existing development processes of the team so is easier to implement and has been successfully applied in several projects.

KEYWORDS

Software Reliability Growth Model, Defect Estimation, Software Quality Tracking, Schedule Prediction

1. INTRODUCTION

Almost every project team wants to meet their schedule and cost targets. Delay in a project can waste a lot of resources and may even result in cancellation of the project. In some cases a failed project can also make a company go bankrupt [1]. Early projection of when a project would complete with a quality product also enables the rest of the supply chain to align with the delivery of the product.

This document describes the model used to predict launch readiness of software in a complex product. After providing some background the paper explains the model used to predict launch readiness. This is then explained with metrics from real projects. Finally, issues which need further research and current best practices to adopt are briefly discussed.

To maintain confidentiality, the company, product and project names are not used in the paper and the dates are changed as a further security measure.

Natarajan Meghanathan et al. (Eds) : ICCSEA, WiMoA, SPPR, GridCom, CSIA - 2017
pp. 127– 138, 2017. © CS & IT-CSCP 2017 DOI : 10.5121/csit.2017.71110

2. BACKGROUND

This paper covers complex projects with schedule ranging from six months to about two years. The complexity here is best explained by a) source lines of code (kSLOC) which was over a million, b) interaction of sub-systems which were mechanical, electronics and software and c) the developed software which included embedded, back-office and personal computer applications and low level drivers.

Low confidence in predictability of software launch readiness means that teams in different geographies, including manufacturing, marketing cannot plan to complete brochures and other marketing material to meet the launch date. To improve confidence in predicting launch readiness several software process improvements were initiated and a model to predict launch readiness was developed.

During development, product testing happens at different levels, like unit test, module test, system test, certification test and in different geographic locations. Any of these tests (excluding unit tests) can identify defects which are then included in the model. The reliability of the overall system is tracked using a different mechanism which is not covered in this paper. However, the software issues identified during system testing are treated as defects and are covered in the model.

3. PRIOR PROCESS IMPROVEMENTS

The model described here depends upon the important improvements in defect management which had been implemented within the team in earlier projects. These were:

- Defect Attributes,
- Defect Lifecycle, and
- Defect Tracking

3.1 Defect Attributes

The defect attributes were defined to manage the defects consistently across the teams. The key attributes relevant to this paper are listed below.

- Priority – business priority for fixing the defect as Critical, Major or Minor.
- Severity – severity of the defect from the customer's point of view as 1, 2, 3 and 4.
- State – defines the lifecycle state the defect is currently in. See section 4.2 for the states defined for the Defect Lifecycle used in the model.

3.2 Defect Lifecycle

The following main states of a defect were defined and then used consistently throughout the projects. The main states relevant to this model are listed below.

- New – the defect is created in this state.
- Assign – the defect is assigned to an engineer to resolve.

- Reject – the defect is rejected as invalid.
- No Action Planned (NAP) – the defect is accepted as valid but will not be fixed.
- Fixed – the defect has been fixed.

3.3 Defect Tracking

Trend charts were used in the earlier projects to track the number of defects created and resolved by week. Although the charts had basic information they provided a simple indicator on whether the rate of finding defects is slowing down or not and if the fix rate is keeping up and closing the gap or not.

4. ISSUES WITH EARLIER PROJECTS

The defect management process worked well but had limited business value in that it did not help in planning for the launch of the products. It was not possible to predict launch readiness of a product to plan related marketing and supply activities. This problem is illustrated by the following two charts.

Figure 3 below shows the Defects Projection and Tracking chart for an older project close to launch. The vertical axis shows the total number of defects found or fixed so far in the project and the horizontal axis is date (which has been changed for reasons of confidentiality). Due to the limited information the development team could not predict when the product would be ready for launch and even at the code freeze date had a significant number of open defects. The project launch in this case was delayed due to high level of open defects.

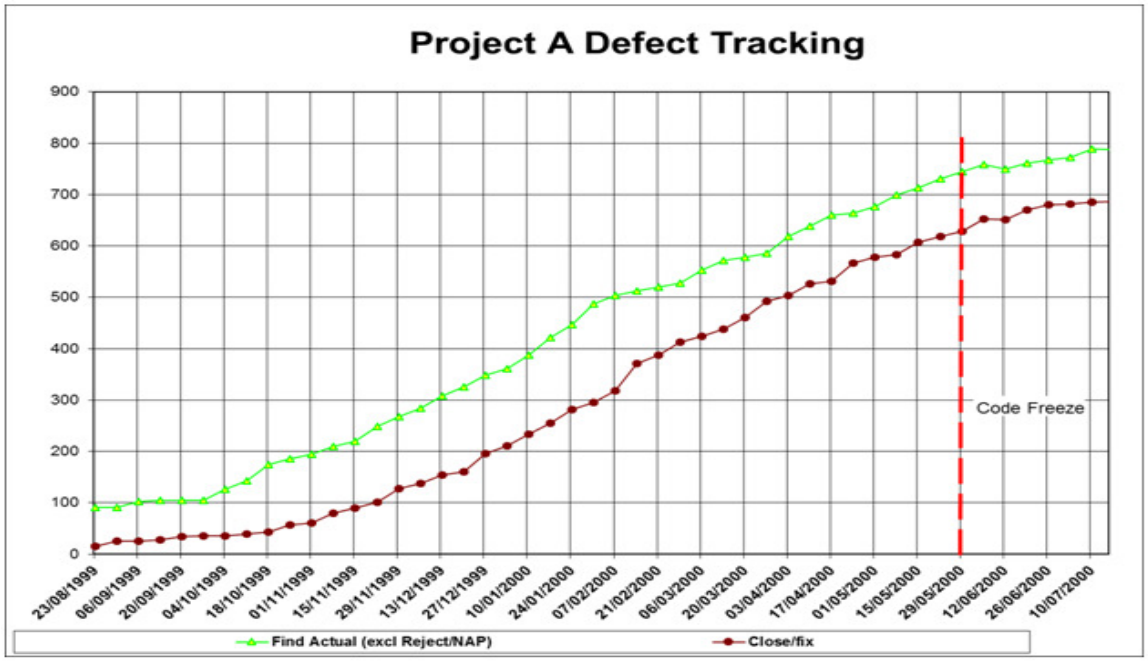

Figure 3 Project A – Defect Projection and Tracking

A few improvements were made to the earlier defect find and fix trend charts. The charts now had a target number of defects to be fixed and showed the maximum fix capability and minimum fix capability of the development team during the hardening period. This helped the defect tracking further by ensuring that the fix rate was kept close to the maximum fix capability.

The improved chart is shown in Figure 4 below. The vertical axis shows the total number of defects found or fixed so far in the project and the horizontal axis is date (which has been changed for reasons of confidentiality). The chart also shows that in the middle of the testing when the fix rate started to slow down corrective actions were taken to bring it closer to the maximum fix projection rate. The chart shows the corrective actions identified while tracking

- 2 spikes show where corrective actions to increase defect find rate were applied and
- one spike shows where corrective action to increase the defect fix rate during the system testing was applied.

Although the updated defect projection and tracking chart was more useful than before but the ability to predict launch readiness was still not there. Launch of this project was also delayed due to the large number of open defects. This eventually led to the development of the software launch readiness prediction model described in this paper.

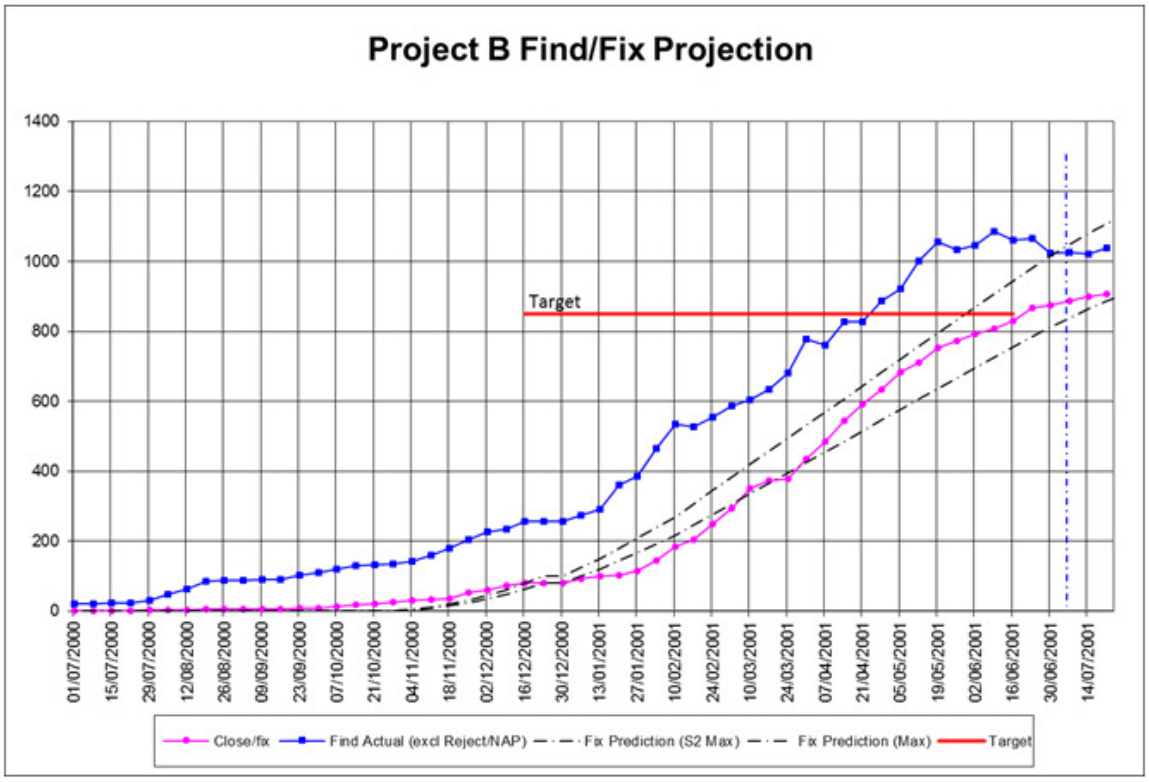

Figure 1 Project B - Defect Projection and Tracking

5. SOFTWARE LAUNCH READINESS PREDICTION MODEL

The following sections describe the key concepts used in the model and the software launch readiness prediction model in detail.

5.1 Concepts Used in the Model

In this model a defect is defined as an error in the software programme, that when executed under particular conditions will result in a failure. Failure means that a function of the software does not meet user requirements.

Reliability is usually defined as the probability that a system will operate without a failure for a specified time under specific operating conditions. Reliability is concerned with the time between failures or its reciprocal, the failure rate. In this model the reliability is tracked as Failure based, where, cumulative failures are recorded within a given time interval (of a week.)

Reliability can also be tracked as system shutdown and system reset rates separately. Any software related shutdowns and system resets are treated as high priority defects. So the assumption here is that if the defects are fixed in time then the rest of the reliability metrics will also get under control.

5.2 Software Development Lifecycle

The Model proposed here uses historical data in the Initial Phase so the software development processes used in the past projects are important. Minor changes in the processes can be ignored but significant process changes mean that the historical data may be of little use for projection.

The software development lifecycle used is also important to understand the scope of the model. For example,

Delivery: Incremental delivery? Iterative delivery? Software Reliability Growth Model (SRGM) are used in the model for projecting the defect find rate so it is important to pick the right model. Several SRGM differentiate between completion of implementation and start of system testing (hardening). Some SRGM do allow for test to start in parallel with implementation but the model used here assumes that over eighty percent of the development has been completed.

Defects: Lifecycle of a defect is important for the tracking part, for example, when are the defects raised and how are these counted? A forum is used to review and accept failures found in any testing as software defects using a consistent method of classifying, prioritising and allocating defects.

Review: Existing code review/inspection processes? The software quality and by inference estimate of defects depends on the quality and consistency with which review processes are used during development.

5.3 Defect Estimating and Tracking Approach

The approach in the model is based on three phases:

- The Initial Phase: Initial Defect Estimates & Fix Projection
- The Interim Phase: Revised Estimates & Update Fix Projection
- The Final Phase: Defect & Fix Projection with Test Tracking

5.3.1 The Initial Phase: Initial Defect Estimates & Fix Projection

Early in the development, estimate the total defects which will have to be fixed to meet the quality target. There are several methods to estimate defects and my recommendation is to first estimate the size and then use the size to predict the number of defects. This is because, given the project requirements, the 'size' of the project should not change. Also, if the size can be measured on delivery then it makes it easier to update the estimates. Unfortunately, there is no perfect metric for measuring size so for simplicity kSLOC (kilo SLOC) is used in the model for size. However, it is possible to estimate defects directly. In case the product size in kSLOC is estimated then using the historical data estimate the number of defects injected for every 1 kSLOC code developed is determined.

Now all this is put together:

- Given the total defects to find use the Rayleigh curve (or another SRGM [2]) to project defect find rate.
- From the available team estimate when engineers will complete implementation and start fixing defects.
- Determine average fix capability from historical data, as number of days to fix a defect per engineer.

This provides a chart showing the projected defect find and projected defect fix curves. Add the defect estimate and the launch build date as targets on the chart. Finally, add actual find and fix charts which will be populated regularly as part of tracking.

5.3.2 The Interim Phase: Revised Estimates & Update Fix Projection

Once majority of the development is complete, say 80% delivered using Earned Value Method (or another appropriate method), measure the actual size (kSLOC) delivered and defects raised to revise the defect estimate. This is the time to revise the estimates based on implementation already completed, testing started and implementation still to complete.

Now update the following:

- The total defect estimate,
- Fix capability of the team from fixes delivered so far (if significant defects have been delivered) and
- Team availability for the rest of the duration of the project.

And obtain the following from the defect tracking database:

- Defects already found and
- Defects already fixed

Update the chart with the revised find and fix projections. This chart allows for tracking the rate with which defects are being found and fixed and to take corrective actions as required to stay on schedule:

- If defects find rate is low then possible options are to review and improve the test plan, increase test resources etc.
- If defect fix rate is low then possible options are to review and improve the defect fix process, increase engineers allocated to fixing defects etc.

5.3.3 The Final Phase: Defect & Fix Projection with Test Tracking

Once significant testing has completed then switch to using the defects arrival rate from different tests and the amount of testing still to complete to estimate the remaining defects to find.

The first step is to review the total defect estimate made in the Initial Phase by comparing the defects already found and fixed with the estimates. From the defects found so far determine the rate with which different test groups were identifying defects. Using this rate of defect detection and remaining tests still to complete, determine the updated estimate of the remaining defects to find. Update the defect projection and tracking chart with the revised estimates providing the remaining defects to find.

6. CASE STUDY

The model has been applied to the projects following Spiral Model and Scrum adapted for the organisation. The model can also be applied to other lifecycles, for example, Waterfall lifecycle provided the historical data used for estimates and forecasts followed similar lifecycles.

6.1 Projects Selected for the Case Study

The projects were selected from two product families. All the selected projects involved changes in electronics, mechanicals and software. Software size for all the products was very similar and grew over time as new projects added more features. Newer features and thus projects tended to be more complex with higher interaction between different components.

The names and dates of the projects have been changed. The main discussion below is for project C. Project D is an earlier project whose data is used as historical data. The final section provides the charts from other projects where the same model was also used.

6.2 Initial Phase

In project C, the current project, the defects were estimated using the analogous estimate method where the defects in similar modules of an earlier software project (Project D) were used as a starting point. The %Injected factor was derived for each module using the cost factors defined in COCOMO [1] as a guide.

Table 1 Project C - Analogous Defect Estimates

Module	Project D	%Injected	Most Likely	Minimum	Maximum	Weighted Estimate
A	130	1.5	195	25	220	171
B	140	1.2	168	60	200	155
C	234	0.8	187	125	200	179
D	53	0.25	13	10	25	15
E	60	0.25	15	10	30	17
F	46	6	276	200	350	276
G	16	2	32	25	40	32
H	20	1	20	10	25	19
I	8	3	24	8	30	22
J	123	0.05	6	5	10	7
Total	830	Total	937		Total	892

The table above shows the number of defects found in Project D, estimated percent defects which will be injected in Project C which then provides the most likely estimate for the defects. The minimum, maximum and most likely estimates for the defects were derived in consultations with the respective SMEs. The weighted estimate is given by the following equation.

$$Weighted\ Estimate = \frac{Minimum + 4 * Most\ Likely + Maximum}{6}$$

6.3 Intermediate Phase

Reasonable calibration using data from the project is now possible as the project is about half way through the testing. Use the development teams' capabilities to fix defects so far to project the fix trend for the rest of the project duration. Similarly, use the existing defect find trend to project the defect find trend for the rest of the project duration.

The chart in Figure 1 shows the defect find and fix projections with the actual defects found and fixed in the early stages of system testing. The vertical axis shows the total number of defects found or fixed in the project to date and the horizontal axis is date (which has been changed for reasons of confidentiality).

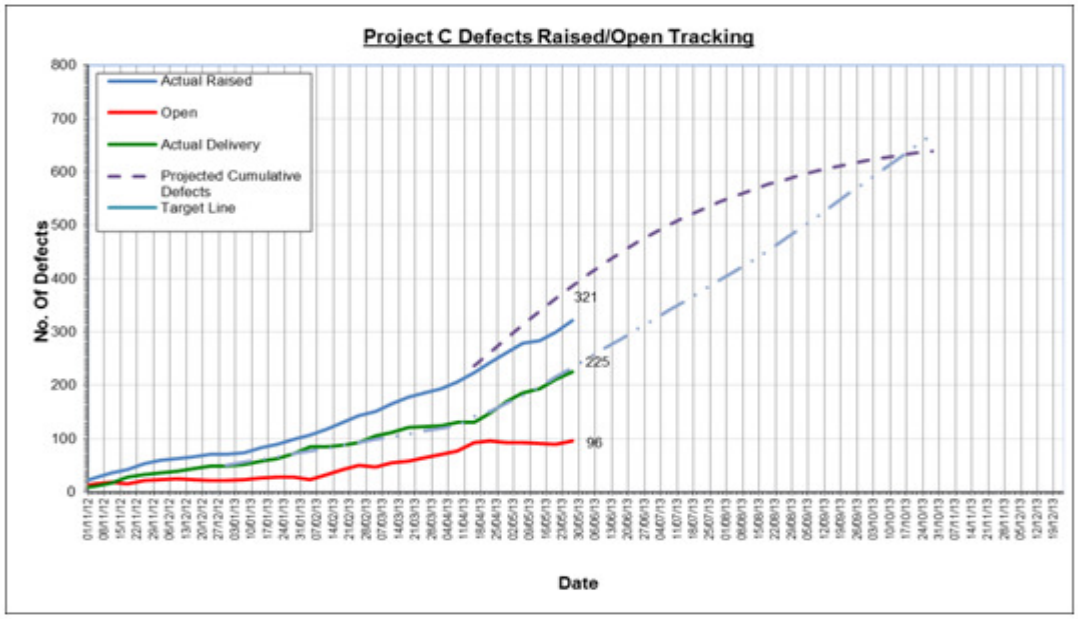

Figure 1 Project C - Defect Projection and Tracking

6.4 The Final Phase

The chart in Figure 2 shows the projected arrival rate of the defects based on the testing still to complete. The vertical axis shows the total number of defects found or fixed to date in the project and the horizontal axis is date (which has been changed for reasons of confidentiality).

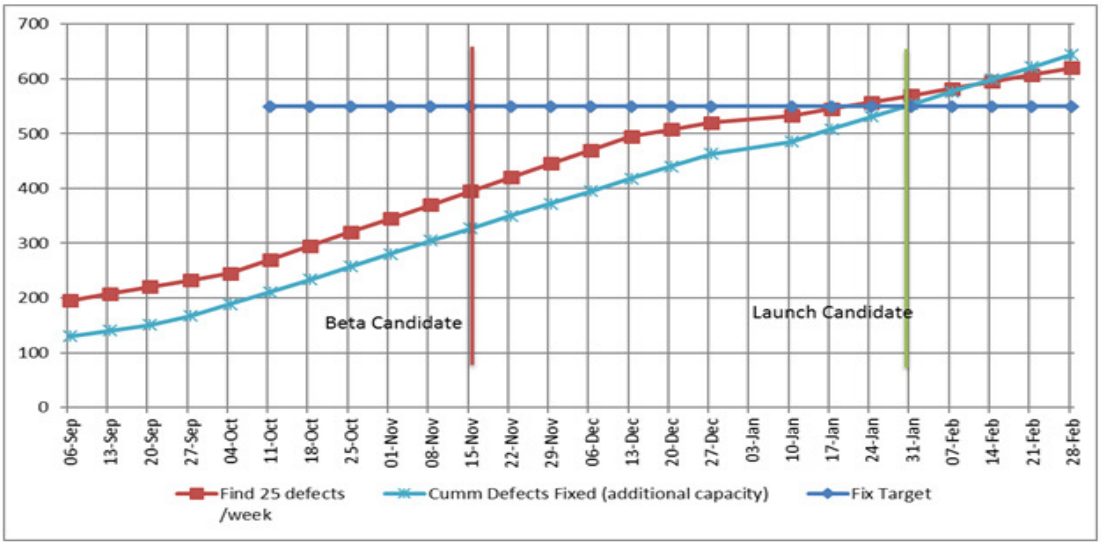

Figure 2 Revised Projection based on Test to Complete

For Project C, discussed earlier, the Table 2 below shows the initial estimates and the defects already found in testing which then provides the number of defects still to find. This can also be used as a sanity check for the estimate of defects still to find based on the defect arrival rate from different tests in progress.

Table 2 Review of Estimates in the Final Phase

Module	Project D	%Injected	Most Likely	Minimum	Maximum	Weighted Estimate	Project C Found	Still to Find
A	130	1.5	195	25	220	171	89	82
B	140	1.2	168	60	200	155	125	30
C	234	0.8	187	125	200	179	145	34
D	53	0.25	13	10	25	15	11	4
E	60	0.25	15	10	30	17	12	5
F	46	6	276	200	350	276	190	86
G	16	2	32	25	40	32	18	14
H	20	1	20	10	25	19	16	3
I	8	3	24	8	30	22	17	5
J	123	0.05	6	5	10	7	2	5
Total	830	Total	937		Total	892	625	267

7. APPLYING THE MODEL

The method described in this paper of defect find and fix tracking is fairly simple but there are some assumptions which need to be resolved before the model can be successfully applied. The main assumptions which the model relies on are described below.

7.1 Development Process

The method assumes consistency between development processes used in the current and the previous projects for the Initial Estimation Phase. From this method's point of view, this includes having standard review process, definitions of defects, priority, severity etc. Benefits of process improvements take time to percolate through the system so these should only be considered after their first successful implementation.

7.2 Historical Data

The model assumes that historical data from previous projects is present and applies to the current project. In absence of any historical data industry standard metrics will have to be used in the beginning.

7.3 Tool Calibration

The models and techniques mentioned in this document have been developed under specific environment and need to be calibrated for the organisation.

7.4 Initial Estimate of Total Defects

It is assumed that the team can estimate defects reasonably reliably. Developers find it difficult to estimate number of defects but the method presented here requires the initial estimate of total defects which will be found during development and test. One method which has been found to work better is a mix of analogy and work break-down system (WBS). In this method break the system into smaller sub-systems (WBS items) and then compare the new project with the defects fixed in similar projects delivered in the past.

7.5 Consistent Allocation of Defects

An important assumption is that the team is disciplined in using the related processes throughout the software development lifecycle. Relation between failure and defect is often unclear so a forum which is consistent in its analysis of the incoming failures may help the team. This forum also needs to ensure that duplicate defects are not assigned but are rejected instead.

8. FURTHER RESEARCH WORK

Research in SRGM has included adaptations to existing models to take differences in development and testing processes, for example, developing and testing phases in parallel, restarting test after fixing defects, into account. Artificial intelligence is also being used, to improve learning from the historical database which can then be used to predict schedule and quality of future projects.

Constructive Cost Modelling [1] (COCOMO) was developed by Barry Boehm and is used by several researchers and tool vendors, for example, COSTAR [7], Cost Xpert[4]. Software Lifecycle Management by QSM [3] uses historical data, Raleigh distribution to manage complete software planning and tracking tool SLIM-Suite. Bayesian Belief Network is another way to predict software (or product) quality (and risks) which AgenaRisk [6] is using.
Further research is required in the following main areas.

- Reliable early defect estimates with limited information
- Machine learning techniques to improve estimates and predictions
- Data mining of the defects database for estimates and predictions

9. CONCLUSIONS

Delivering reliable software on schedule is a concern in all development organizations. With increase in size and complexity, several vendors and research institutes are looking into tools and methods on how to improve predictability in software development. Most of these tools are expensive and require significant effort to learn and normalize for the development teams to start getting benefits.

The method described in this paper is simple and practical and any team, disciplined to use processes consistently can start using it with little additional effort. However, initially, the team will need some historical data to base their estimates and predictions on. The method uses tools and concepts readily available to all. The method has been successfully applied in several projects in the past with excellent results. The method can be easily enhanced as more and more data is collected within the development teams without tying them to an external vendor.

REFERENCES

[1] Robert N. Charette, Why Software Fails, 2008, http://spectrum.ieee.org/computing/software/why-software-fails, last accessed on 11th August 2017.

[2] Reliability Growth Model, http://www.ece.uvic.ca/~itraore/seng426-07/notes/qual07-8.pdf, last accessed 11th August 2017.

[3] Barry W Boehm et al, Software Cost Estimation with COCOMO II, 2000, ISBN-10: 0137025769

[4] CoStar, http://www.softstarsystems.com/

[5] Cost Xpert, http://www.costxpert.com/

[6] Software Lifecycle Management (SLIM), http://www.qsm.com

[7] AgenaRisk, http://www.agenarisk.com

[8] P.K. Kapur, D.N. Goswami, Amit Bardhan, Ompal Singh, Flexible software reliability growth model with testing effort dependent learning process, Applied Mathematical Modelling, Volume 32, Issue 7, July 2008, Pages 1298-1307

[9] M.Xie, G.Y.Hong, C.Wohlin A Practical Method for the Estimation of Software Reliability Growth in the Early Stage of Testing, http://www.wohlin.eu/issre97.pdf last accessed 11th August 2017.

AUTHOR

Abhinav Sharma is a PMP® (PMI certified) professional with more than 25 years of proven track record in Product/Program/Project Management and development. Main products covered Wearable IoT, Consumer products, Multi-function printers and embedded software. Proven track record of delivery to schedule of large, complex, multi-site, multi-discipline projects and implementing lean/agile/process improvement initiatives.