

EIGHTEENTH ANNUAL
PACIFIC NORTHWEST
SOFTWARE QUALITY
CONFERENCE

OCTOBER 17 - 18, 2000

Oregon Convention Center
Portland, Oregon

Permission to copy without fee all or part of this material, except copyrighted material as noted,
is granted provided that the copies are not made or distributed for commercial use.

When Will We Be Done Testing?

Software Defect Arrival Modeling Using the Weibull Distribution

Erik Simmons
Intel Corporation
JF1-46
2111 NE 25th Ave.
Hillsboro, OR 97214-5961
erik.simmons@intel.com

Version 1.3, 8/7/00

Prepared for the 18th Annual Pacific Northwest Software Quality Conference

Key Words: Software Defect Arrival; Weibull; Software Testing

Author Biography

Erik Simmons has 15 years experience in multiple aspects of software and quality engineering. He holds a Masters degree in mathematical modeling and a Bachelors degree in applied mathematics from Humboldt State University in California. Erik currently works as Platform Quality Engineer in the Platform Quality Methods group, part of the Corporate Quality Network at Intel Corporation.

Abstract

One of the most common yet vexing questions asked of Software Quality Assurance managers and testers is "When will we be done testing?" Product engineering and marketing groups have a vested interest in knowing when the software under test will be at an acceptable level of quality. While this question is not at all easy to answer, modeling the arrival of defects during testing can provide clues such as predictions of when a given percentage of the estimated total defects will be found, or the time at which the rate of newly arriving defects will be below a given threshold. The Weibull distribution serves as an excellent model for software defect arrival. Three case studies based on actual projects are provided.

Introduction

Software Quality Assurance managers, testers, and others involved in software system testing are often confronted with questions from product engineering and marketing groups concerning when the software will exit the test process at an acceptable level of quality. This information is essential to planning product launches, staffing and training support teams, and similar activities. An accurate prediction of the defect arrival rate over time is one part of the information required to get beyond the obvious answer “when enough of the defects are gone”. Of course, your answer might be “when the release date arrives”, but that’s another problem.

The Weibull distribution was created in 1937 by Waloddi Weibull. Since the 1950s, it has been used to model phenomena in a wide range of disciplines. Weibull analysis has a large following today, where it is used for modeling hardware failures, analyzing radar clutter, predicting warranty and support costs, forecasting spare parts levels, and many other purposes [Abernethy98]. The Weibull distribution has tremendous flexibility. It can accommodate decreasing, constant, and increasing failure rates. Few if any other models can match its combination of breadth and simplicity.

The Weibull distribution has been shown to describe the defect arrival pattern of software projects [Kan95], [Lyu95], [Putnam92]. The process for modeling defect arrival with the Weibull distribution involves several steps. First, an estimate of the total number of defects in the software must be made. This estimate is then used to determine the cumulative proportion of defects that arrive each time period. Estimates of the parameters of the Weibull distribution are then obtained from a regression line fit to a mathematical transformation of this data.

In this paper, the parameters of the Weibull distribution were estimated using Microsoft[®] Excel[®], though any spreadsheet or statistical software package could be used. The first parameter of the distribution is given by the slope of the fitted line, and the second is calculated with a simple formula. Once these parameter estimates are known, inferences about the future defect arrival patterns can be made.

The projected defect arrival rate can be used to predict the time at which a given percentage of the estimated total defects will be located, or when the number of newly discovered defects per time period is likely to be below a given threshold. These predictions can be used to validate the viability of a release date or to track progress towards a given quality goal. For example, if a release criterion for a software system states that 95% of the total estimated defects must be removed, the analysis described here can be used to predict the date at which that milestone will be met, given the project’s current staffing and effort.

The Weibull Distribution

The form of the Weibull distribution used for this analysis is the two-parameter Weibull. The cumulative distribution function (CDF) of the two-parameter Weibull is given by [Abernethy98] as:

$$F(t) = 1 - e^{-(t/\eta)^\beta}$$

In this formula, β is called the shape parameter, and η is called the characteristic life (the point at which 63.2% of the failures have occurred)¹. Figure 1 shows examples of the Weibull probability density function (PDF) with $\eta = 10$ and $\beta = .5, 1, 2, 3,$ and 4 .

¹ To see this, substitute $t = \eta$ in the Weibull CDF to get $F(t) = 1 - 1/e = .632$, regardless of β .

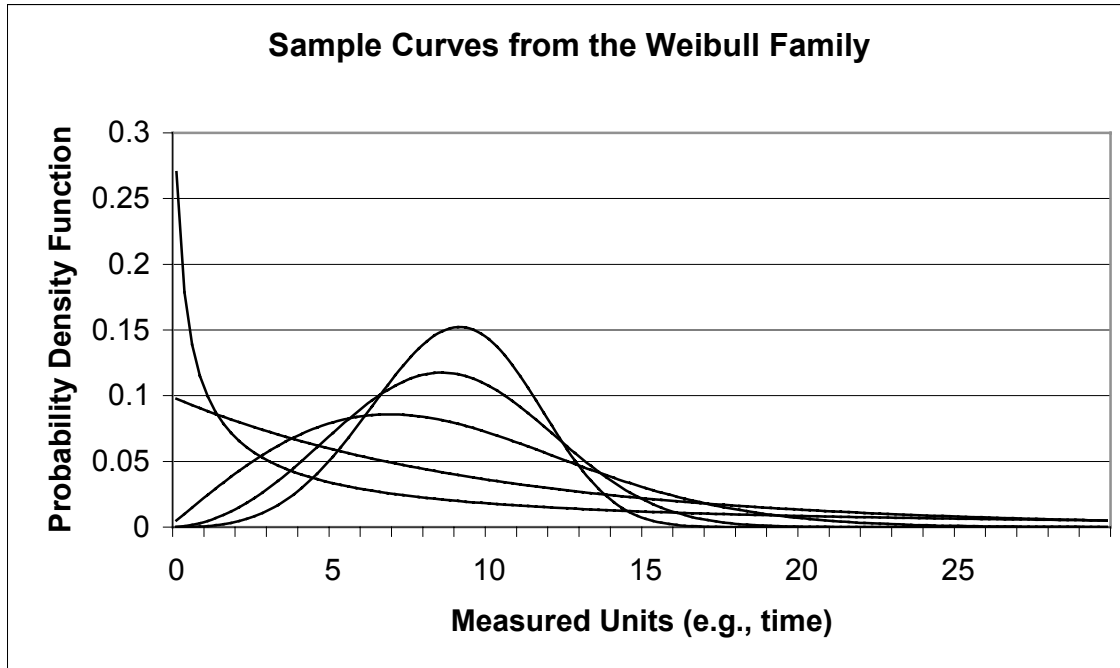


Figure 1

In reliability applications, the measured quantity is time, duty cycles, miles, or similar quantities associated with failure. When β is less than one, reliability increases with time (known as infant mortality). When β is equal to one, the Weibull distribution reduces to the Exponential distribution, implying random failure. When β is greater than one, reliability decreases with time (wear-out).

Modeling Defect Arrival Data

Defect arrival modeling is a subset of software reliability modeling. There are two general classifications for software reliability models [Lyu95], depending on the type of data used:

1. Failures per time period
2. Time between failures

Distinction between two types of time period is provided in [Musa87], where the authors note that software reliability can be modeled in an execution time domain or a calendar time domain. The execution time domain is generally accepted as superior to the calendar time domain. However, calendar domain data is more commonly collected.

In an execution time domain, time is measured in such a way that it represents the processor time used by the software under test. This measurement can be as fine as CPU seconds, or may be a coarse measurement such as the cumulative number of hours the system is under test by each tester. For example, 40 hours of accumulated time by the test team is equal to one tester-week, regardless of the elapsed calendar time required.

Modeling defect arrival in a calendar time domain ignores changes in test effort² and simply records the number of failures or the time between failures against elapsed calendar time.

² This could be caused by changing the number of testers, imposing automated test suites instead of manual testing, etc.

Since most managers think more easily in the calendar time domain, a means of translating between the execution time and calendar time domains is helpful. It is also important to understand the difference between time under test and time under normal use – that is, how many hours of typical field use are equivalent to one hour of test time³.

The model presented in this paper uses failures per time period, measured in a calendar time domain as the number of failures per week. Although not as precise as more sophisticated measures of execution time like CPU seconds or even failures per 40 hours of testing, calendar time data is relatively easy to measure and provides acceptable results within the model as long as the test effort is reasonably consistent. The data in the case studies contains events such as the addition of a tester, occasional vacation and sick days, etc., but there are no large-scale changes like moving from manual to automated testing, doubling the number of testers, or an extended hiatus in testing.

Weibull Model Assumptions

The assumptions of the Weibull model are (after [Lyu95]):

1. The rate of defect detection is proportional to the current defect content of the software
2. The rate of defect detection remains constant over the intervals between defect arrivals
3. Defects are corrected instantaneously, without introducing additional defects
4. Testing occurs in a way that is similar to the way the software will be operated
5. All defects are equally likely to be encountered
6. All defects are independent
7. There is a fixed, finite number of defects in the software at the start of testing
8. The time to arrival of a defect follows a Weibull distribution
9. The number of defects detected in a testing interval is independent of the number detected in other testing intervals for any finite collection of intervals

These assumptions are often violated in the realm of software testing. Despite such violations, the robustness of the Weibull distribution allows good results to be obtained under most circumstances.

Assumption 2 can be violated by changes in the test plan, changes to the number of testers, introduction of automated testing, and similar factors. Assumption 3 is violated 100% of the time in the case of instantaneous correction, and about 1/6 of the time for perfect correction [Jones97]. Assumption 4 depends on an accurate operational profile, which may be difficult to prove. Assumption 5 is violated when defects are masked by other defects, or more generally when one failure mode covers another. Assumption 6 is commonly violated in software, where dependent defects abound.

Fitting the Weibull Distribution to Defect Arrival Data

The steps to fit the two-parameter Weibull distribution to defect arrival data are:

1. Obtain an estimate of the number of defects in the software
2. Calculate the cumulative proportion of total defects arriving each period
3. Transform the data to obtain a linear form
4. Fit a least-squares line to the data
5. If the fit is acceptable, use the line to obtain estimates of β and η .
6. Plot the Weibull distribution versus the actual data

³ For example, how many hours would a typical user need to accumulate to match the amount of use placed on a word processing program in one hour by an automated test suite?

The parameters can be re-estimated each period by adding the new data point and repeating steps 2 through 6. Each step of the modeling process is described below.

Step 1: Obtain an Estimate of the Number of Defects in the Software

There are several different methods for obtaining this estimate. Historical data can be used when it exists. Several commercial applications provide estimates of total defects given various inputs. In many cases, estimates are derived from the total lines of source code or the Function Point count of the application. In one approach, an estimated Function Point count is 'backfired' from the total logical statements in the source code. Tables containing the average logical statements per Function Point for most common languages are available [Jones97]. Once the estimated Function Point count is known, the number of defects present can be estimated using an assumed or known defect potential per Function Point. If desired, a range of estimates (high, expected, and low) for the total number of defects can be created and used in subsequent steps.

Step 2: Calculate the Cumulative Proportion of Total Defects Arriving Each Period

This is accomplished by dividing the cumulative total of located defects for each time period by the estimated total defects. For example, assuming an estimate of 1200 total defects measured week by week:

Table 1

Week	New Defects	Cumulative Proportion F(t)
1	5	0.004
2	10	0.013
3	27	0.035
4	82	0.103
5	94	0.182
6	61	0.233
7	77	0.297
8	111	0.389

Step 3: Transform the Data to Obtain a Linear Form

In order to fit a linear regression line, the CDF of the two-parameter Weibull must be rearranged to obtain a linear relation. After some algebraic manipulation, the Weibull can be written in the following form:

$$\ln\left(\ln\left(\frac{1}{1-F(t)}\right)\right) = \beta \ln(t) - \beta \ln(\eta)$$

So the cumulative proportion F(t) is transformed to ln(ln(1/(1-F(t))), and Weeks are transformed to ln(Weeks). The required transformations the data from Table 1 are given in Table 2.

Table 2

ln(Week)	ln(ln(1/(1-F(t))))
0	-5.47855
0.693147	-4.37574
1.098612	-3.33465
1.386294	-2.21576
1.609438	-1.60701
1.791759	-1.32947
1.94591	-1.04434
2.079442	-0.70739

For ease of plotting, the y-coordinate is typically rescaled to zero by subtracting the first y value from all y values as shown in Table 3. This step is optional, and is just done to have all the plotted y values be non-negative.

Table 3

ln(Week)	$\ln(\ln(1/(1-F(t)))) + 5.47855$
0	0
0.693147	1.102808
1.098612	2.143905
1.386294	3.262797
1.609438	3.871539
1.791759	4.149079
1.94591	4.434213
2.079442	4.771166

Step 4: Fit a Least-Squares Line to the Data

Now that the data has been transformed, plot the data and fit a least-squares regression line (see Figure 2).

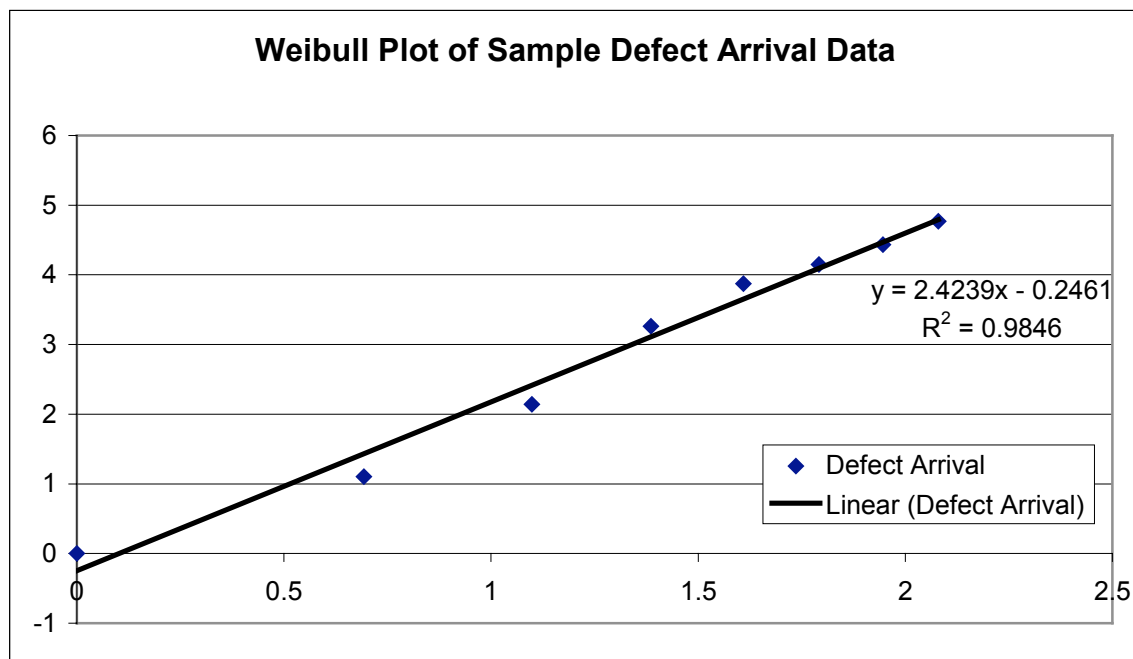


Figure 2

Step 5: If the fit is acceptable, use the line to obtain estimates of β and η

The correlation coefficient of the fit is .9846, so this fit is acceptable as about 98% of the variability in the data is explained by the line. The estimate of β can be read from the equation for the line as 2.4239. The value of η can be calculated using the fact that η is the 63.2 percentile of the distribution, that is, the time at which 63.2% of observations have occurred. This means that when $t = \eta$, (remember we added 5.47855 to rescale all y values in Step 3)

$$\ln\left(\ln\left(\frac{1}{1-.632}\right)\right) + 5.47855 = 2.4239\ln(\eta) - .2461$$

Solving this equation, $\eta = 10.6$ weeks. These calculations can be automated easily in Excel.

Step 6: Plot the Weibull Distribution versus the Actual Data

This step is actually optional, as inferences can be made from the Weibull plot constructed in the previous step. However, because of the significant rescaling and transformation of the data, it is a good idea to return the plot to normal space before presenting it.

The plot makes use of the WEIBULL function in Excel, which has the following arguments: WEIBULL(data, β , η , Cumulative?). In this function, the data values are the time periods (in this case, 1 through 8), and the ‘Cumulative?’ parameter determines whether the resulting plot is cumulative (the CDF) or not (the PDF). Figure 3 and Figure 4 show examples of both types for the sample data:

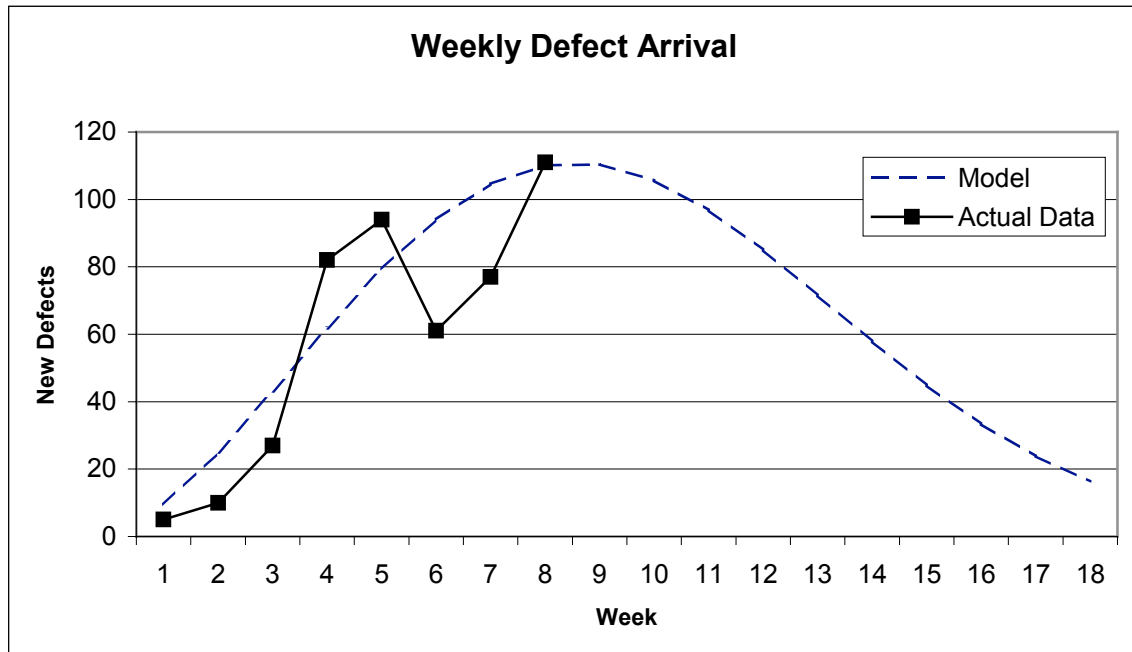


Figure 3

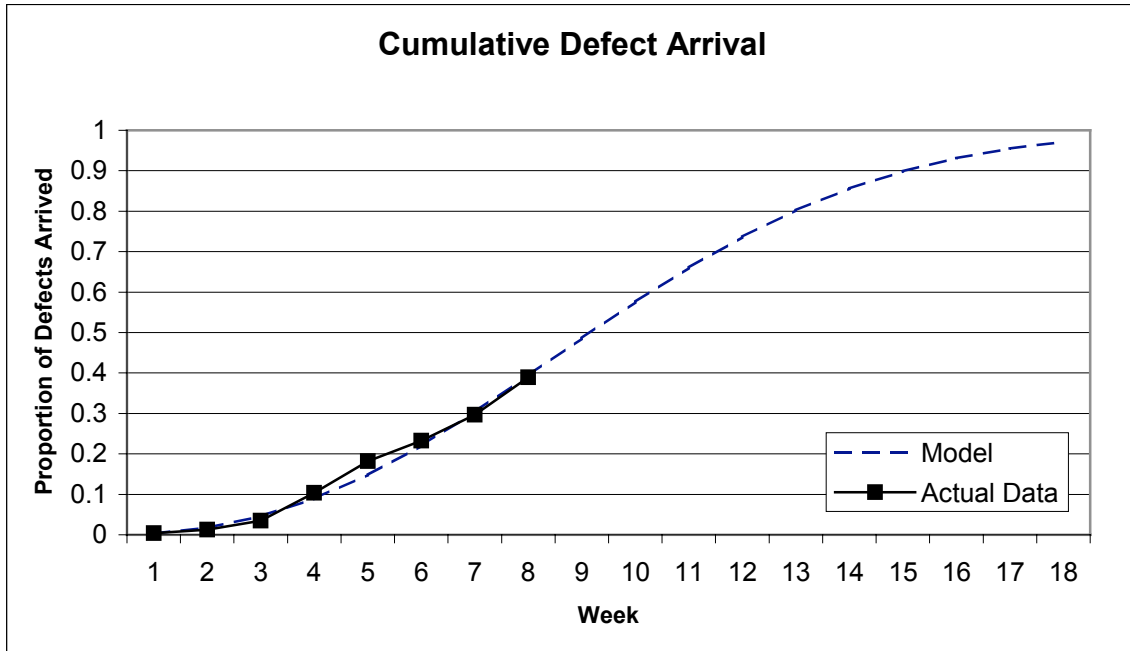


Figure 4

There is a danger in presenting cumulative measures. On projects with a large number of defects, cumulative charts can mask even reasonably big trends, precisely when they are of greatest interest. For this reason, the weekly arrival chart is normally a better indication of defect arrival trend. An example of this problem is given in the second case study.

Results

Three case studies from actual projects completed over the last several years are given below. The fits are performed on non-cosmetic (Projects A and B) or High/Medium priority defects (Project C), depending on the type of data available. In all three cases, the fits including cosmetic and low priority defects were not as good. One possible reason for this is that most testers tend to focus more on the significant and severe defects first, and focus on less serious or cosmetic issues only when few remaining serious issues can be found.

Project A

Project A was a traditional two-tier client/server application, written by a team of four developers of varying experience and one tester. Testing was manual, according to a well-written test plan. The Weibull distribution was fit post-hoc using the 1008 non-cosmetic defects located during testing.

The model's fit is shown in Figure 5. There are two failure modes evident in the plot. Note that the decision to isolate distinct failure modes must be made based on knowledge of the process used to create the system, not just on anomalies in the data. If there is a solid process or engineering reason why a failure mode should be separated, it is appropriate as long as the new failure mode adds value and not just complexity to the overall model.

The first failure mode, lasting three weeks, reflects a period when integration defects caused blocking failures that prevented full system execution. This is 'infant mortality', since $\beta = .19$, far less than one. The longer the system ran, the more reliable it became (that is, if an integration

defect did not cause a nearly immediate problem, the software could be executed much more fully).

The second failure mode represents system testing once blocking was removed. For the second failure mode, $\beta = 1.8$ and η can be calculated to be 17.1. There are noticeable deviations from the model for this failure mode. Examining the data, one might wonder if there are really three failure modes contained within the period after week 4: a pattern of shallow slope, followed by a steeper slope, then shallow again. Is there a reason to separate the steep slope from the two shallower slopes?

In manufacturing environments, this shallow-steep-shallow pattern in Weibull plots is associated with batch defects where a known, stable failure mode suddenly changes due to defective materials, equipment out of tolerance, etc. A similar situation in software engineering is when a module of code is produced with an uncharacteristically high error content. This is precisely what occurred on Project A in week 6. After exploring this further, the model using four failure modes was not significantly better than the one with only two modes for purposes of estimates and projections, so the simpler two-failure-mode model was used.

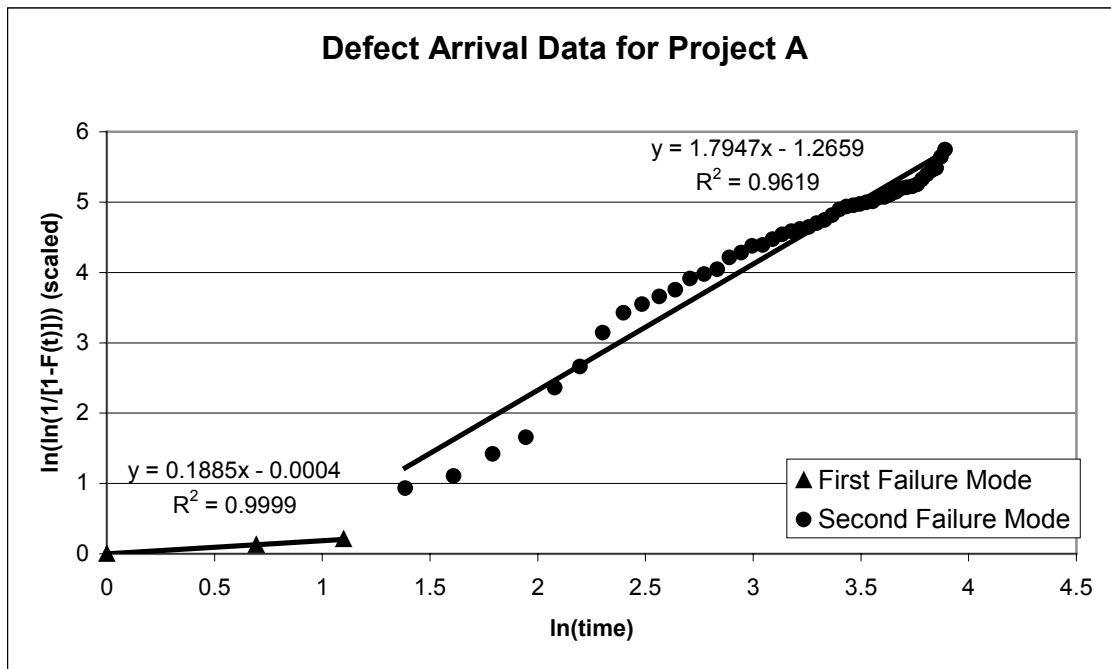


Figure 5

The actual weekly defect arrival versus two models for Project A is shown in Figure 6. In this figure, the effects of the integration and blocking defects can be seen, as can the large spike in defect arrival after the integration of a large and defect-prone module in week 6. The final model fits the data quite well after the spike in defects has past.

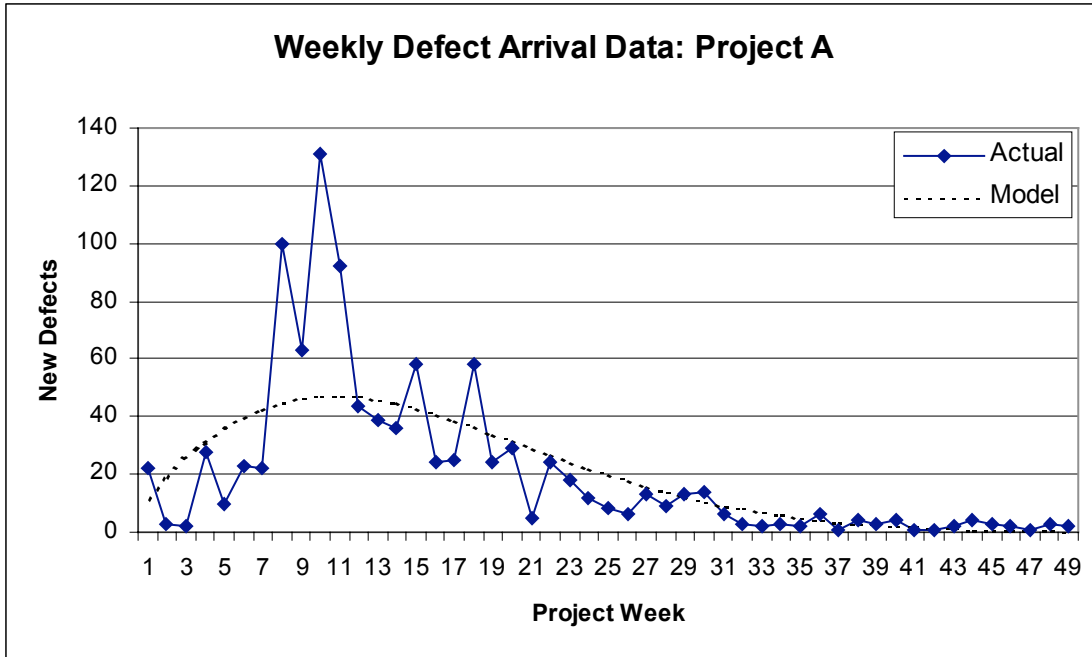


Figure 6

Project B

Project B was a Web-enabled three-tier application built using distributed objects. It was written by a team of 8 developers, and there were between 1 and 2.5 testers on the project at any given time. The development team was made up of moderately experienced to highly experienced individuals. Testing was manual, based on an extensive, formally inspected test plan.

The system was integrated in three approximately equal stages. The integration cycles were all relatively smooth, and there were few regressions.

The Weibull distribution was fit using an initial estimate of 1000 non-cosmetic defects, but the quality proved to be better than that estimate. The total defect estimate was adjusted to 800 near week 18 of the project. The linear fit given the actual 739 defects is shown in Figure 7. There are three failure modes present. The first failure mode represents integration defects blocking execution (as in Project A). The second failure mode represents system testing and shows the characteristic undulations of the 'build rhythm' of an iterative project. This failure mode has $\beta = 3.26$ and η about 24.5 weeks.

The final failure mode corresponds to customer acceptance testing. During this three-week period, the equivalent of several additional fulltime testers were evaluating the product at the customer site, increasing the number of defects located per week.

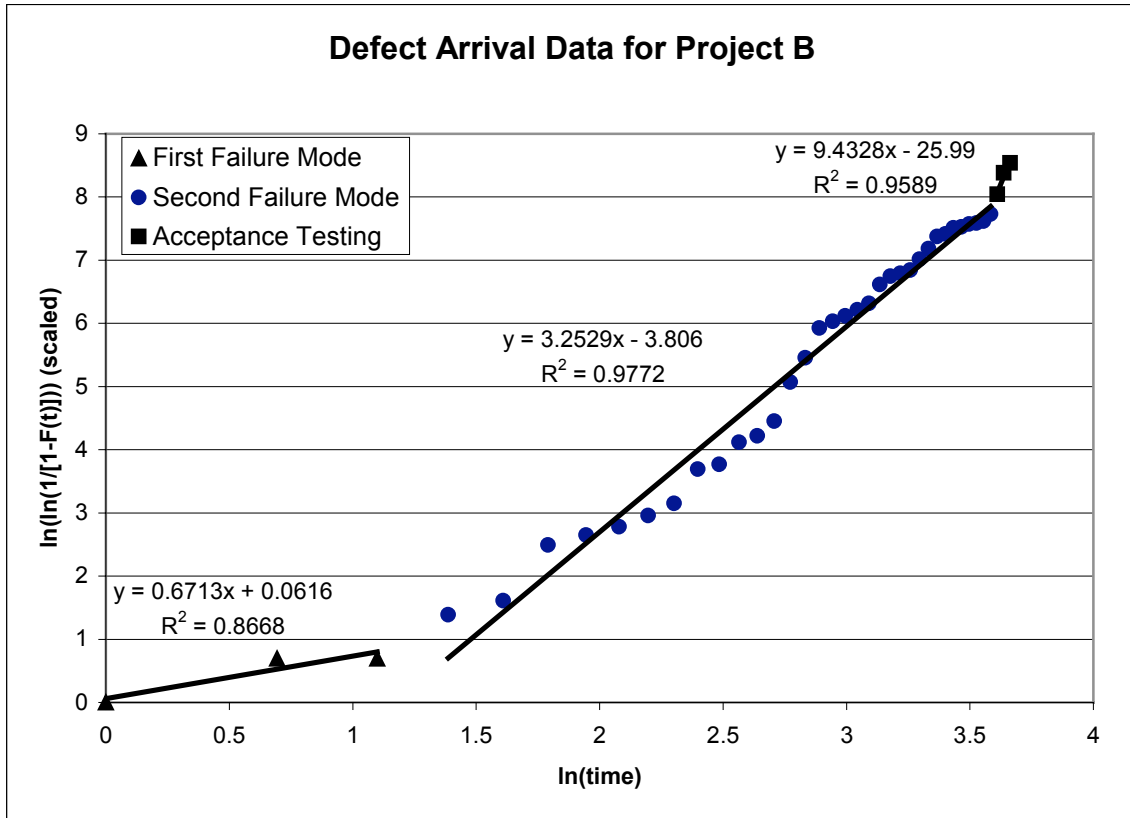


Figure 7

The actual weekly defect arrival versus the model for Project B is shown in Figure 8. The effects of initial integration and blocking defects can be seen on weekly defect counts through week 6. It is interesting to note that with each new integration, the height of the spike in defect arrival is reduced and the time between integrations is shorter – a comforting trend. The spike in defects represented by acceptance testing (weeks 36-38) is also apparent, but might be missed in casual observation without the benefit of the data in Figure 7.

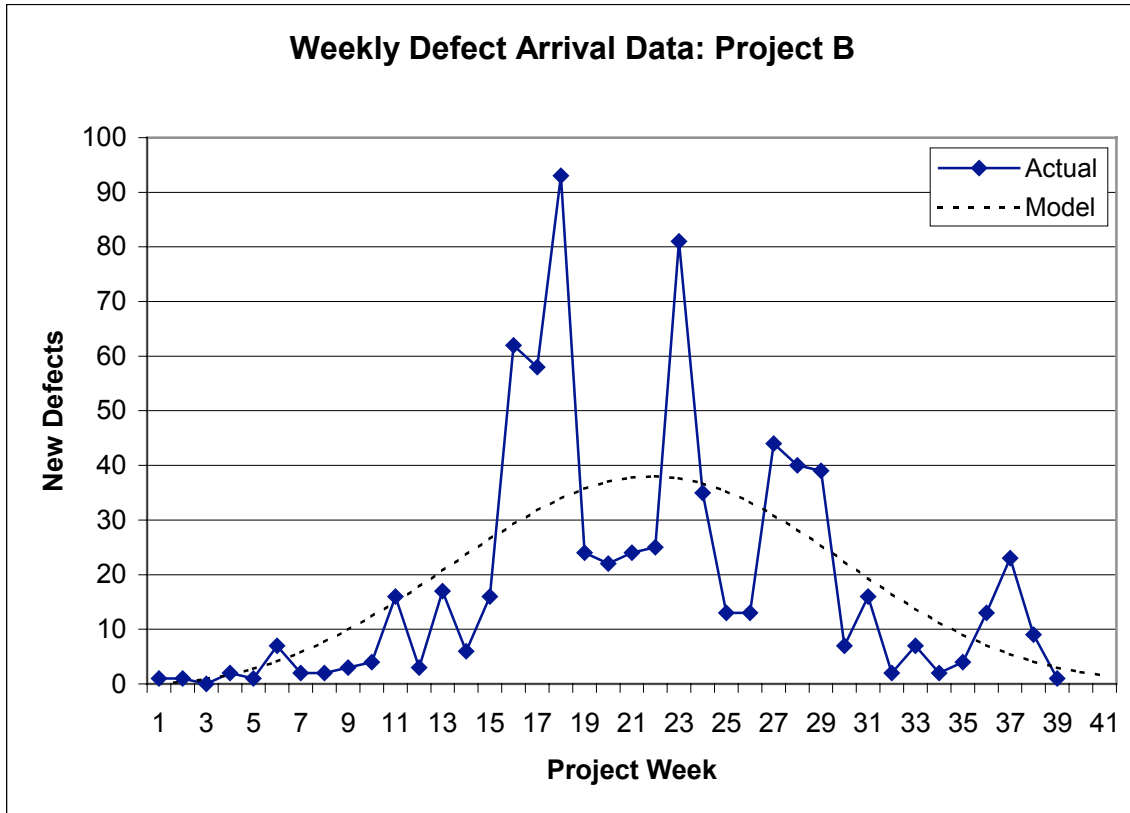


Figure 8

The cumulative defect arrival pattern for Project B is shown in Figure 9. While this view of the data has some value, it is also a good illustration of the way that cumulative statistics can hide local trends. The significant spike in defect arrival between weeks 36 and 38 looks no different than other periodic differences caused by the build rhythm, but Figure 7 shows that it is very different. Had this been a trend towards a much higher defect density (through increased defect injection, for example), the problem may not have been apparent for several more weeks using only the cumulative view of the data.

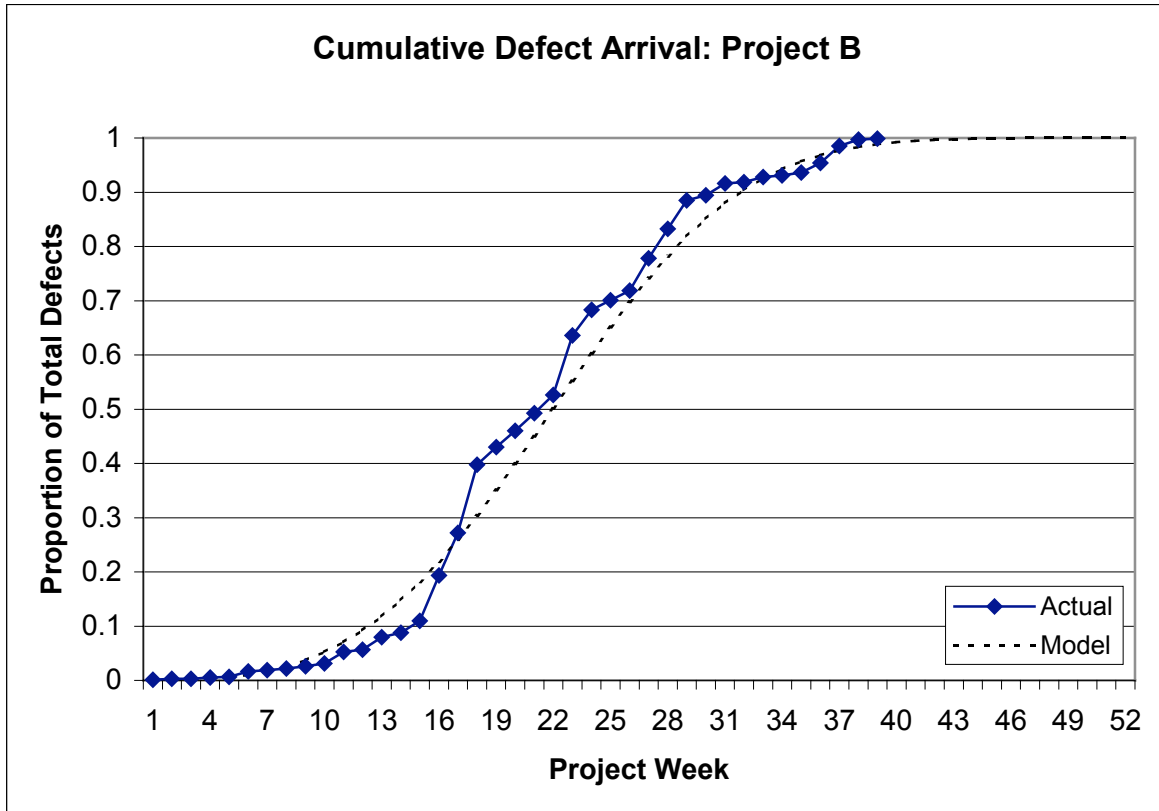


Figure 9

Project C

Project C was a traditional two-tier client/server application written by a team of 12 developers. The system was based on a core system that already existed, but there were extensive additions. There were between 4 and seven testers on the project at various times. Testing consisted of a mix of manual and automated testing, but was predominantly manual. Testing was based on extensive test plans and specifications.

The system was built through several small to moderate integration cycles, with one larger cycle near the end of the project.

The Weibull distribution was fit initially to an estimate of 2500 total defects (2000 assumed to be high or medium priority) obtained from commercial software. After examining the defect arrival trends about three-fourths of the way into the project, it became apparent that the total defect count was closer to 3000 (2500 assumed to be high or medium priority). This later estimate of 2500 defects is used in the figures that follow. The linear fit is shown in Figure 10. Three failure modes are present. The first mode containing weeks 1-3 represents mainly human factors and usability testing of the design using screen shots and design walkthroughs. Failures in this mode were approximately random (a β of approximately 1). The second mode indicates infant mortality as found in the other two case studies (a β of about .61). Once blocking was removed and functional testing began in week 10, the third failure mode appears with β approximately 2.6.

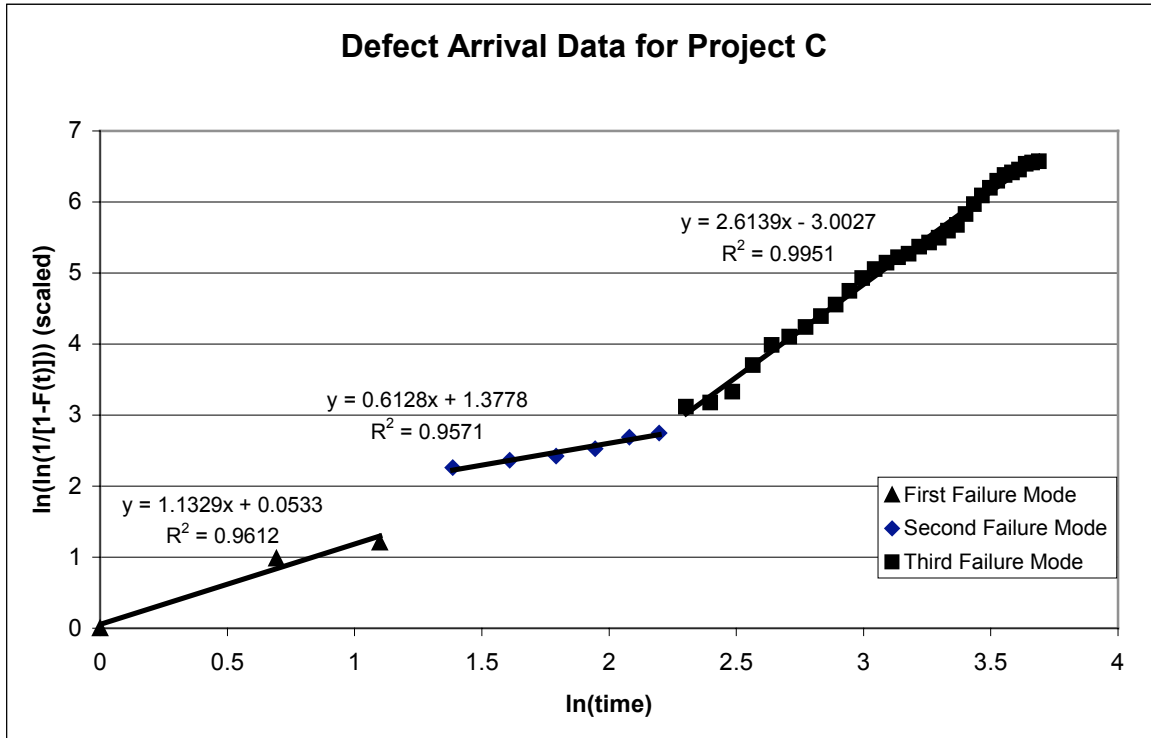


Figure 10

The weekly arrival pattern is shown in Figure 11. The pattern shows the characteristic build rhythm of an iterative project, and the blocking prior to week 10 is evident. The large integration that occurred in week 30 is followed by the largest spike in defect arrival (as might be expected). While this spike caused some concern during the weeks it arrived, examination of the failure mode data as shown in Figure 10 and the defect database gave some feeling that this was not a new failure mode, but more of a singular event. This turned out to be the case, as can be seen by defect arrival levels following week 35.

On this project, defect arrival modeling data was used as strong evidence to adjust the initial schedule estimate. It also was used to help decide when to stop functional testing of the current version of the system and to begin the next version with increased functionality. Based on having found about 90% of the estimated defects, functional testing was curtailed in week 40 and the new round of development was begun.

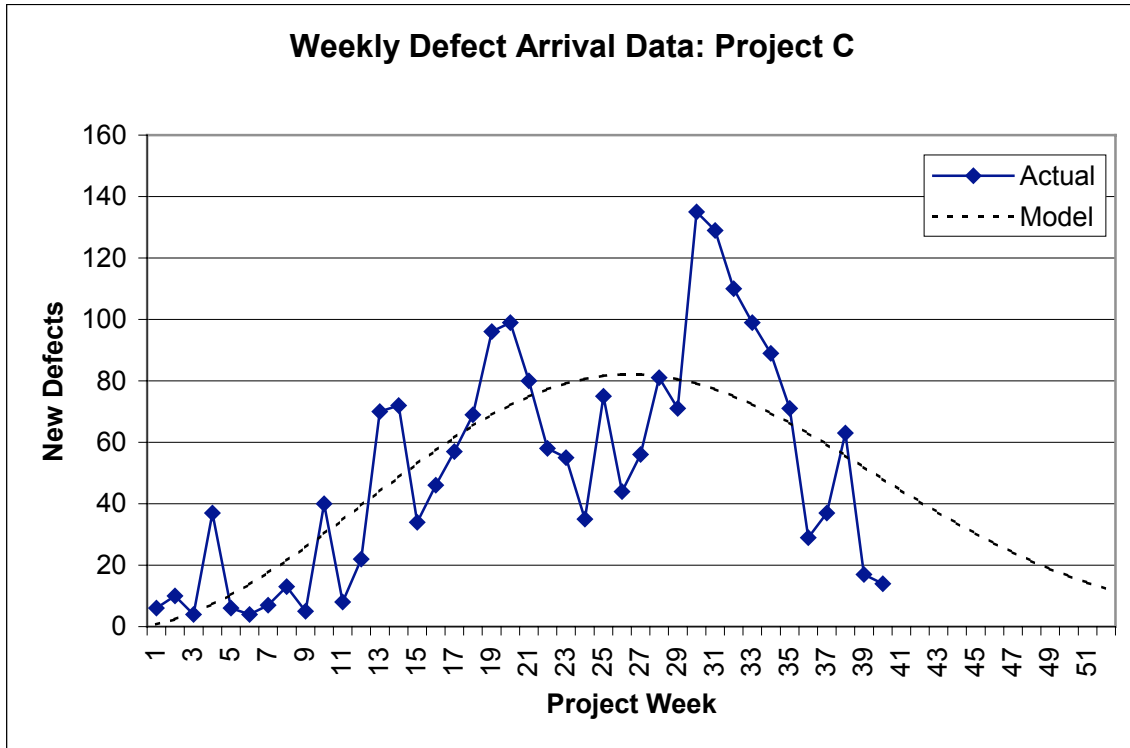


Figure 11

Discussion

When fitting Weibull models to defect arrival data there are some things to consider based on these results:

Fits appear to be better when the more serious (non-cosmetic, high/medium priority, etc.) defects are used. Fits using all defects are not as good.

All three case studies had a period of 'infant mortality' at the start of testing. Once blocking and integration defects were removed, the main failure mode appeared and remained relatively constant thereafter. The length of the infant mortality failure mode appears to be proportional to the size of the system (consistent with intuition).

The value of β for a failure mode is influenced by where on the axes the transformed data are plotted, since both axes are measured in a log scale rather than a linear scale. When early (i.e., infant mortality) failure modes are included in the Weibull plot the value of β for the main failure mode is larger than when it is omitted. This matters if two projects are to be compared for testing effectiveness by comparing their respective β values. The best approach may be to plot the values obtained without the early failure modes included (see Figure 12). In this case, the first two projects each used manual testing and a single tester, and their dominant failure modes have very similar slopes. Project C used several testers and a mix of manual and automated testing, and the dominant failure mode shows a correspondingly steeper slope.

Estimates of β appear to be reliable enough for basic decision support soon after the appearance of the main failure mode. While the successive estimates to β vary by week in the case study data, they are consistent enough that the basic conclusions drawn from the data do not change.

The particular project lifecycle used for a project will influence the defect arrival pattern. Iterative lifecycles will have different patterns than those using continuous integration or a modified waterfall. In [Putnam92], the distribution is used to model defects located in all phases of development through many types of testing and static defect removal activities. In this paper, the data come from system testing only. In either case, the model fits the data well enough to make project management and engineering support decisions.

This technique should work well on all types of applications, relying only on a good source of defect data. However, it has not been widely used so future work may uncover situations where it works too poorly to be useful.

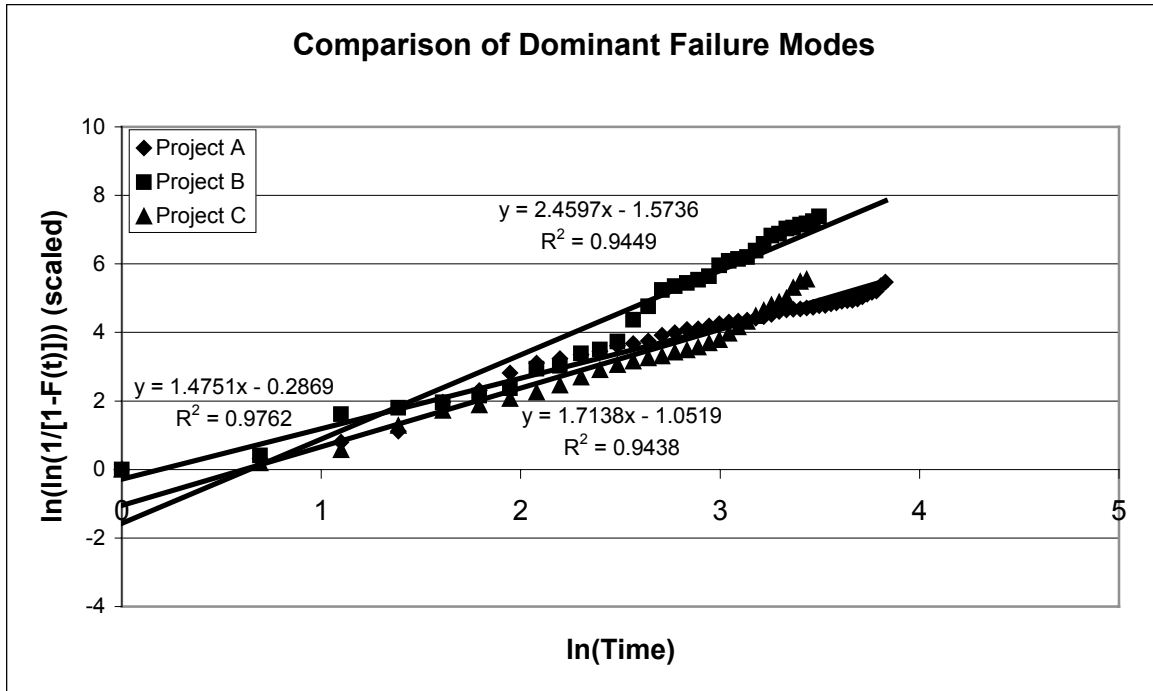


Figure 12

Opportunities for Further Work

There are three principle areas for further work in this area. The first is an understanding of the relationship between factors such as the number of testers, type of testing (manual, automated, mixed), etc. and the resulting shape parameter β . This information can be used in a technique known as Weibayes (coined from a contraction of *Weibull* and *Baysian*) to make accurate inferences very early in the testing process when only a few defects have arrived [Abernethy98]. Many such libraries for hardware failure exist, but no one has investigated possible parallels in the software world.

Second, techniques that remove the need to estimate the total number of defects in the software might be explored, as this is in most cases a difficult, error-prone task. Specialized Weibull analysis techniques for interval data might be adapted and applied to software defect arrival given coarse data like that used in this paper. If more precise failure time data are available, the use of median rank plotting positions may be able to remove the need to know the total number of defects.

Last, a better understanding of how results vary for different types of data would be helpful in guiding the choice of what data to collect. The number of failures in calendar time is easy to collect and gives useful results. Other papers could outline comparative fits obtained using this type of data and execution time domain time between failures (generally regarded as the best for these purposes).

Summary

Because of its flexibility and power, the Weibull distribution can be used to create useful models of software defect arrival even in a calendar time domain. The Weibull distribution is fit to defect arrival data via a six-step process that is easily automated in a PC spreadsheet. The resulting models are capable of locating different failure modes caused by changes in the test environment or application under test. Projections based on the models can help drive schedule and release decisions, and might allow a test manager to measure the affects of adding testers, moving to automated testing, and similar strategies on the defect arrival rate. It may be possible to establish libraries for the shape parameter β so that given data on the number of testers, the type of testing, etc., accurate projections of the amount of time required to test the application can be made very early in the project. Better precision within failure time data and or more advanced Weibull analysis techniques may remove the need to estimate the total number of defects in the software.

Acknowledgements

The author acknowledges the kind support of several colleagues who provided data and background for the case studies contained in this paper. Thanks are also due to the reviewers that helped improve the presentation and content.

References

- | | |
|-------------|---|
| Abernethy98 | Abernethy, Dr. Robert B., <i>The New Weibull Handbook</i> , 3 rd ed., Self-published 1998 |
| Jones97 | Jones, Capers, <i>Software Quality: Analysis and Guidelines for Success</i> , Thompson Computer Press 1997 |
| Kan95 | Kan, Stephen H., <i>Metrics and Models in Software Quality Engineering</i> , Addison Wesley 1995 |
| Lyu95 | Lyu, Michael (ed.), <i>Handbook of Software Reliability Engineering</i> , McGraw-Hill/IEEE Computer Press 1995 |
| Musa87 | Musa, John, <i>et al.</i> , <i>Software Reliability: Measurement, Prediction, Application</i> , McGraw-Hill 1987 |
| Putnam92 | Putnam, Lawrence, and Myers, Ware, <i>Measures for Excellence – Reliable Software On Time, Within Budget</i> , Yourdon Press 1992 |

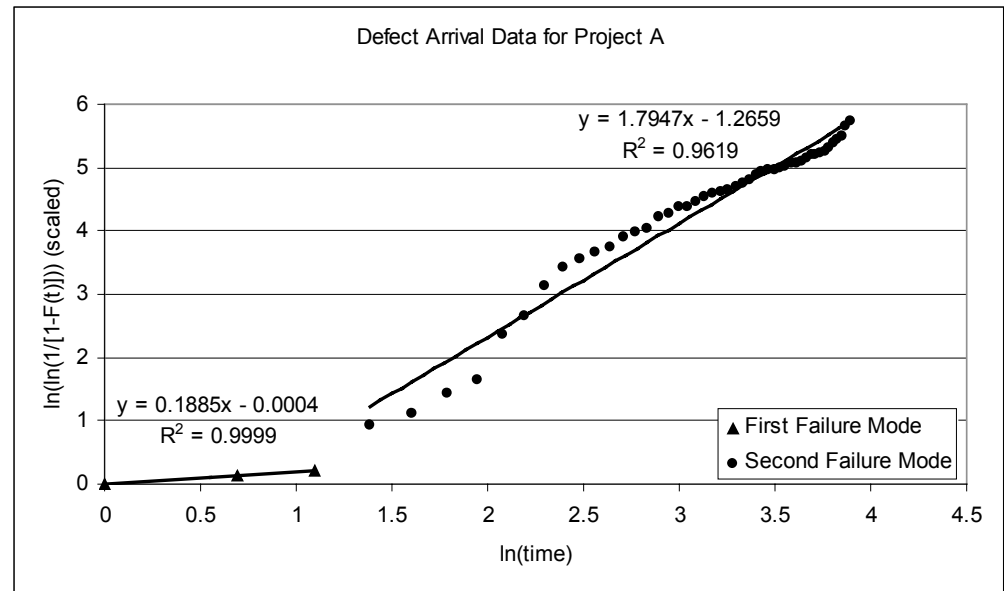
* Third-party brands and names are the property of their respective owners.



When Will We be Done Testing?

Software Defect Arrival Modeling With the Weibull Distribution

Erik Simmons, Intel Corporation





Contents

- The Weibull Distribution
- Software Defect Arrival Modeling – the Basics
- Fitting the Weibull Distribution to Defect Arrival Data
- Case Studies
- Sources for More Information



The Weibull Distribution

Discovered in 1937 by Waloddi Weibull

Used since the 1950s to model diverse things:

- Hardware failures
- Radar clutter
- Warranty & support costs
- Spare parts levels
- And many more...

The two-parameter Weibull:

$$F(t) = 1 - e^{-(t/\eta)^\beta}$$



The Weibull Distribution

The parameters of the Weibull distribution:

β - the Shape parameter

η - the Characteristic Life

$\beta < 1$ indicates 'infant mortality, where the longer the system runs the less likely failure becomes

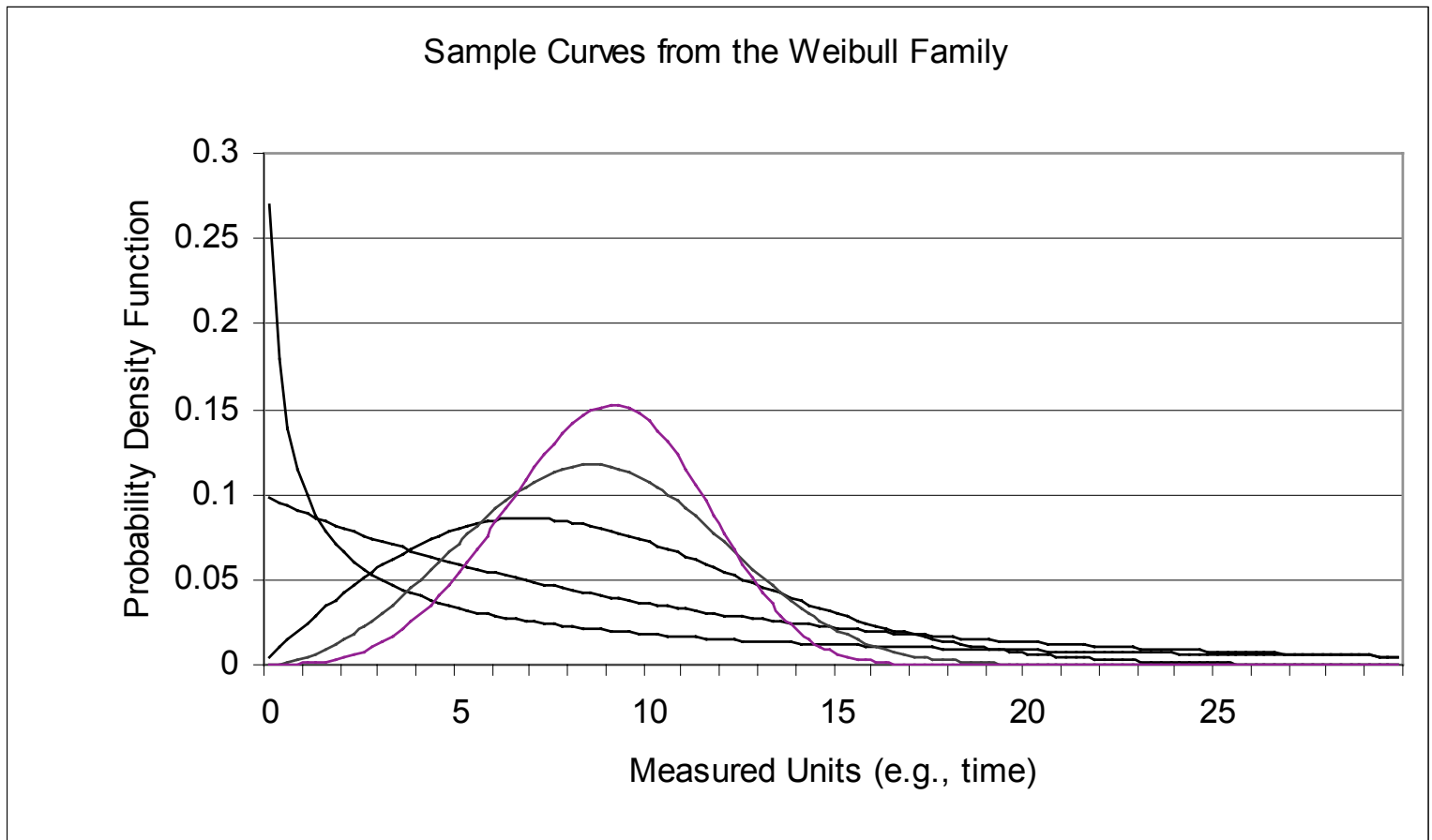
When $\beta = 1$, the Weibull reduces to the Exponential distribution, implying random failure

$\beta > 1$ indicates wear out, where the longer the system runs the more likely failure becomes

η is the point at which 63.2% of the failures have occurred



The Weibull Distribution





Software Defect Arrival Modeling

Failure model classifications [Lyu95]:

- Failures per time period
- Time between failures

Time domains [Musa87]:

- Execution time
- Calendar time

The execution time domain is generally recognized as superior to calendar time, but can be harder to measure

This paper uses the number of failures per calendar week



Weibull Model Assumptions

The assumptions of the Weibull model are (after [Lyu95]):

- The rate of defect detection is proportional to the current defect content of the software
- The rate of defect detection remains constant over the intervals between defect arrivals
- Defects are corrected instantaneously, without introducing additional defects
- Testing occurs in a way that is similar to the way the software will be operated



Weibull Model Assumptions

- All defects are equally likely to be encountered
- All defects are independent
- There is a fixed, finite number of defects in the software at the start of testing
- The time to arrival of a defect follows a Weibull distribution
- The number of defects detected in a testing interval is independent of the number detected in other testing intervals for any finite collection of intervals

Luckily, the Weibull is robust to most violations...



Fitting the Weibull Distribution

The steps to fit the two-parameter Weibull distribution to defect arrival data are:

- Obtain an estimate of the number of defects in the software
- Calculate the cumulative proportion of total defects arriving each period
- Transform the data to obtain a linear form
- Fit a least-squares line to the data
- If the fit is acceptable, use the line to obtain estimates of β and η .
- Plot the Weibull distribution versus the actual data



Sample Data

Week	New Defects
1	5
2	10
3	27
4	82
5	94
6	61
7	77
8	111



Estimate the Total Number of Defects

An estimate can be derived in several ways:

- Historical data
- Commercial software
- LOC → Function Points → Defects
- Etc...

If the estimate is off significantly, you will see it in the plots as time goes on

The estimate can be revised during the process

Sample data estimate: 1200 defects



Calculate the Cumulative Proportions

Week t	New Defects	Cumulative Proportion $F(t)$
1	5	0.004
2	10	0.013
3	27	0.035
4	82	0.103
5	94	0.182
6	61	0.233
7	77	0.297
8	111	0.389



Transform the Data to a Linear Form

The two-parameter Weibull can be re-expressed in a linear form:

$$\ln\left(\ln\left(\frac{1}{1-F(t)}\right)\right) = \beta \ln(t) - \beta \ln(\eta)$$

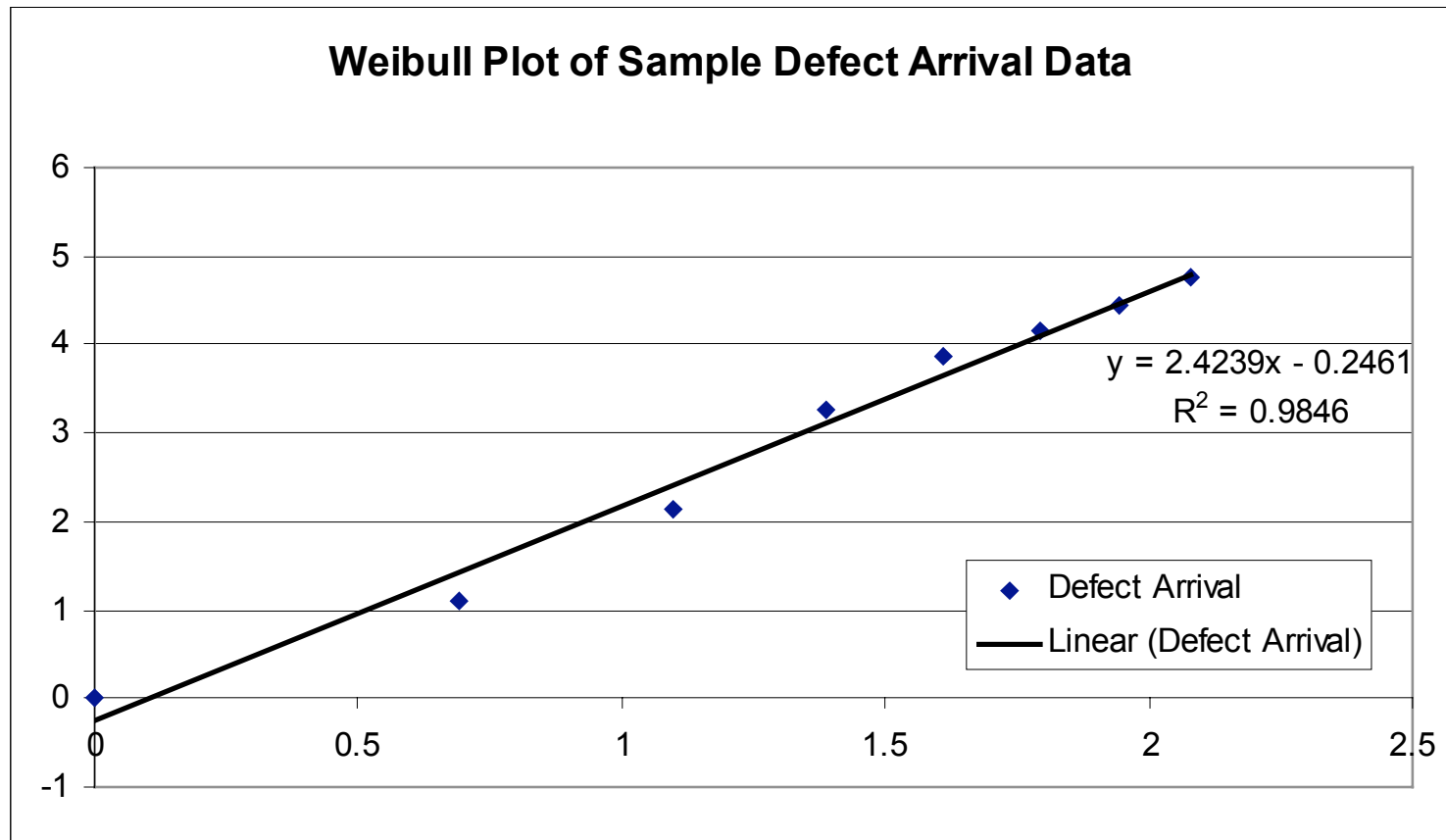
$$Y = mX + B$$



Transform the Data to a Linear Form

$\ln(\text{Week})$	$\ln(\ln(1/(1-F(t))))$	Rescaled
0	-5.47855	0
0.693147	-4.37574	1.102808
1.098612	-3.33465	2.143905
1.386294	-2.21576	3.262797
.1609438	-1.60701	3.871539
1.791759	-1.32947	4.149070
1.94591	-1.04434	4.434213
2.079442	-0.70739	4.771166

Fit a Least Squares Line to the Data





If the Fit is Acceptable, Estimate β and η

The R^2 for the line is .9846, so about 98% of the variation in the data is explained by the line

β can be read from the regression equation as the slope:
2.4329

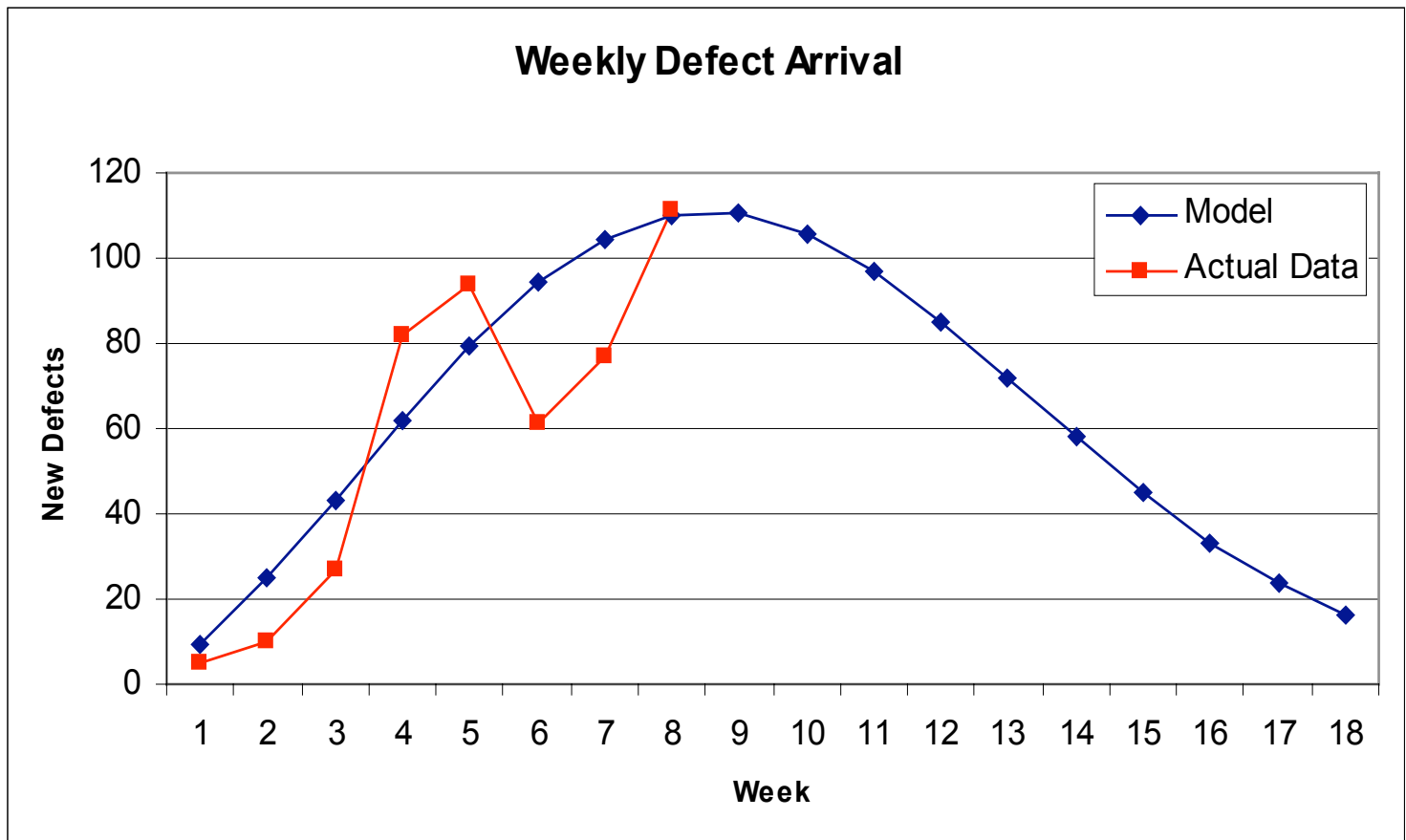
η can be calculated by solving the following equation (remember that we rescaled the data):

$$\ln\left(\ln\left(\frac{1}{1-.632}\right)\right) + 5.47855 = 2.4239\ln(\eta) - .2461$$

Solving, we get $\eta =$ **10.6** weeks

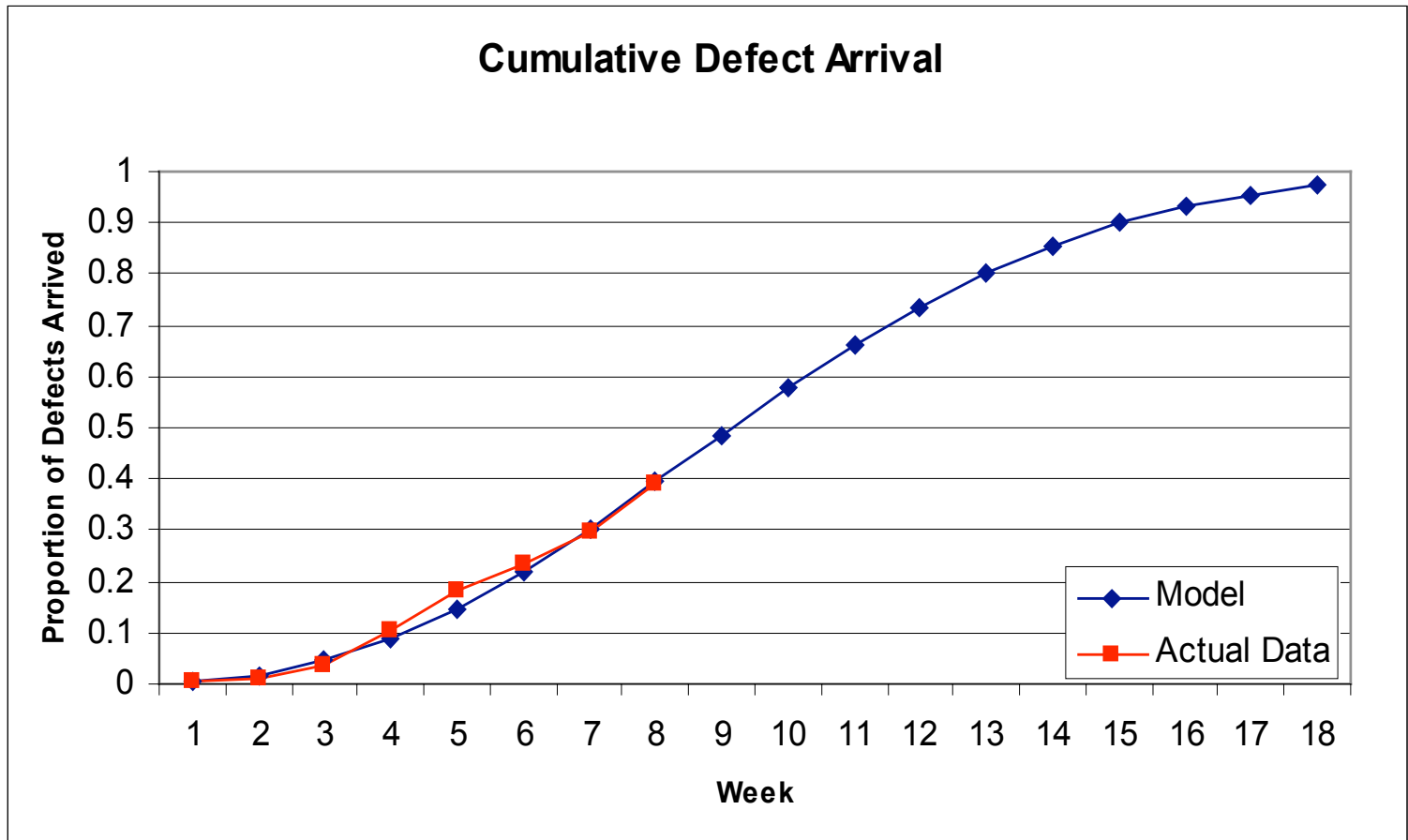


Plot the Model vs. the Actual Data





Plot the Model vs. the Actual Data





Case Studies



Project A

Two-tiered client/server application

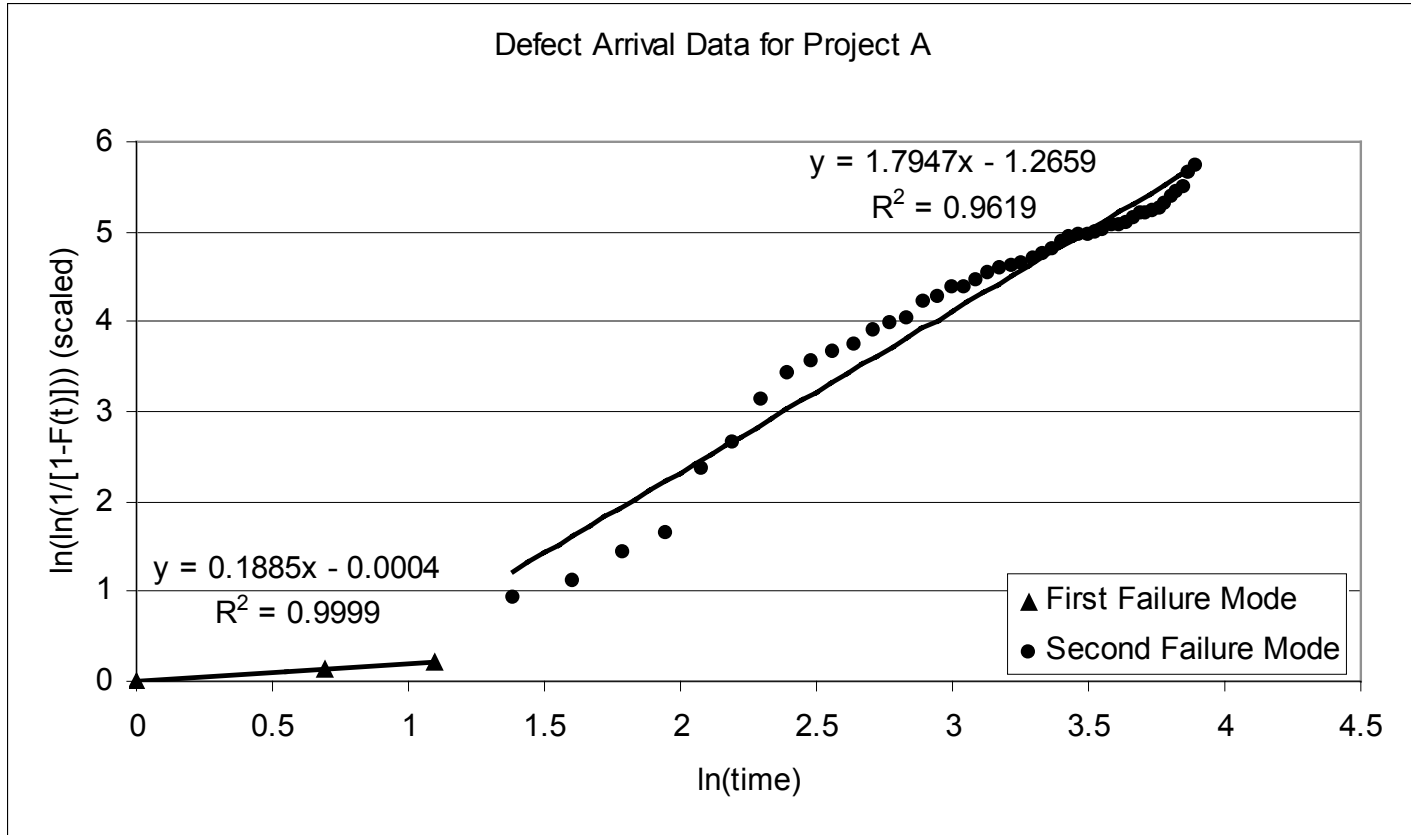
4 developers, 1 tester

Manual testing according to a written plan

Weibull fit using 1008 located defects

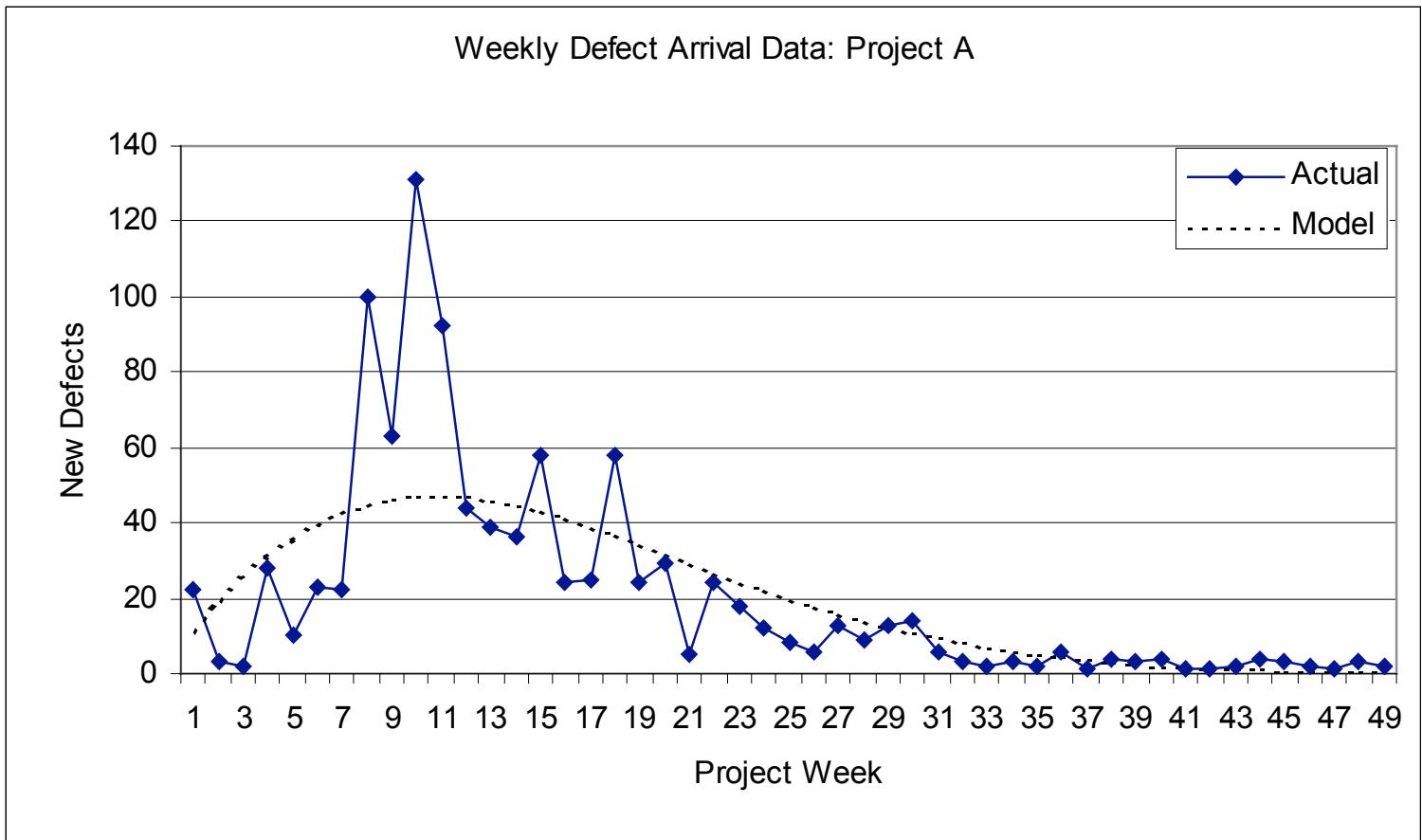


Project A: Weibull Plot





Project A: Weekly Arrival





Project B

Web-enabled three-tiered client/server application

8 developers, 1-2.5 testers

Manual testing according to a extensive, inspected plan

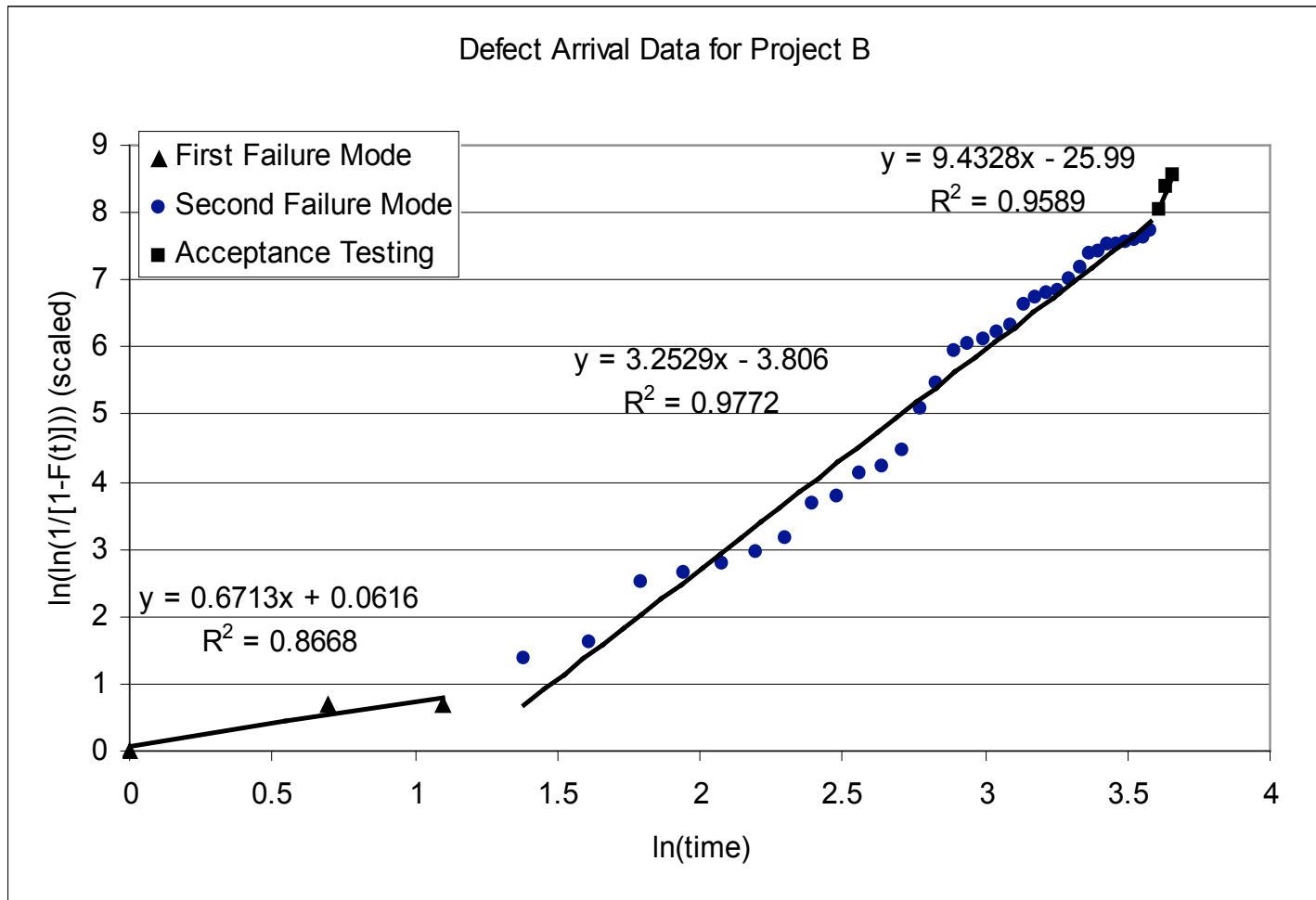
First Weibull fit to 1000 estimated defects

Total defect estimate trimmed to 800 near the end

Final fit to 736 defects

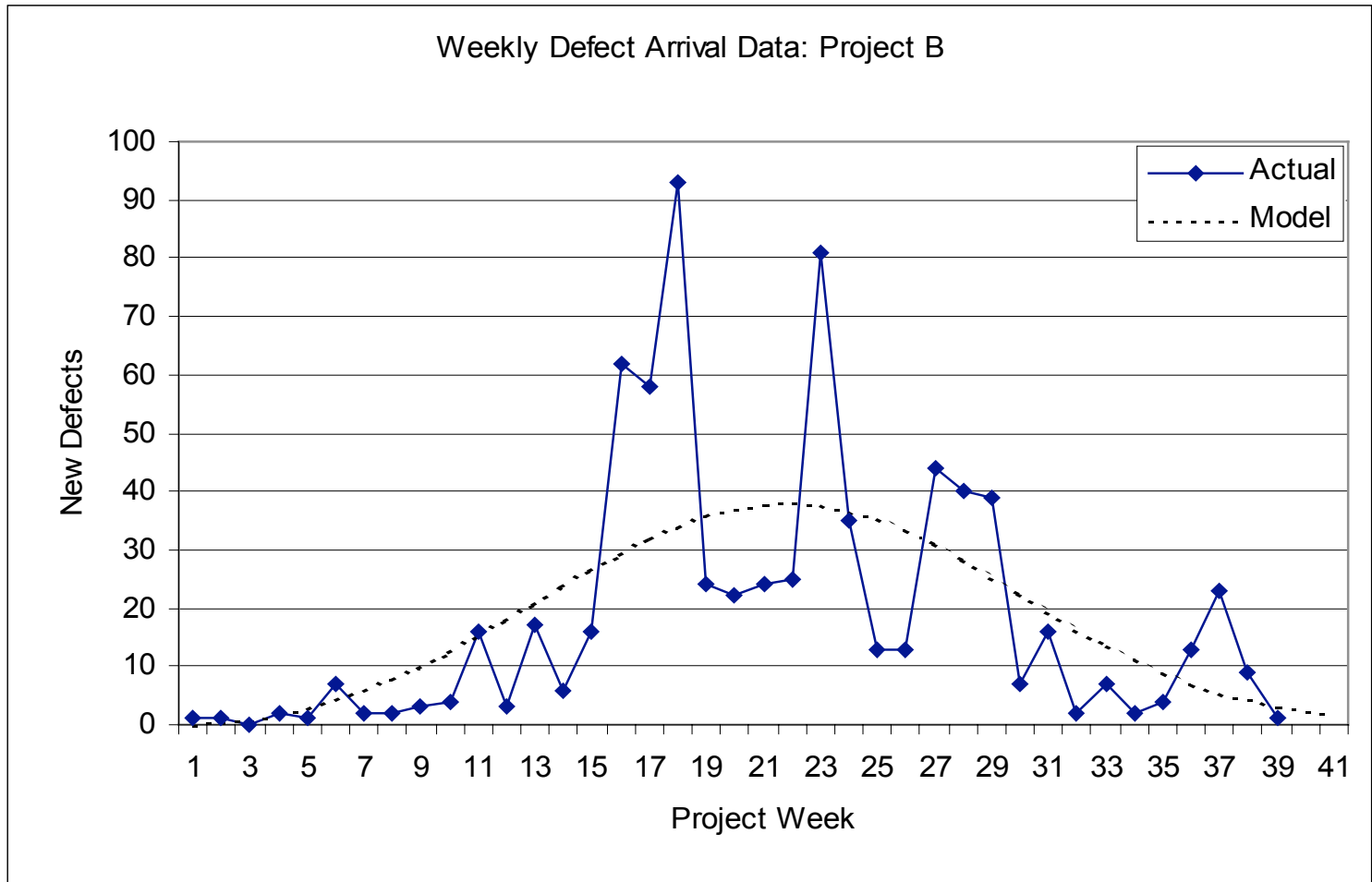


Project B: Weibull Plot



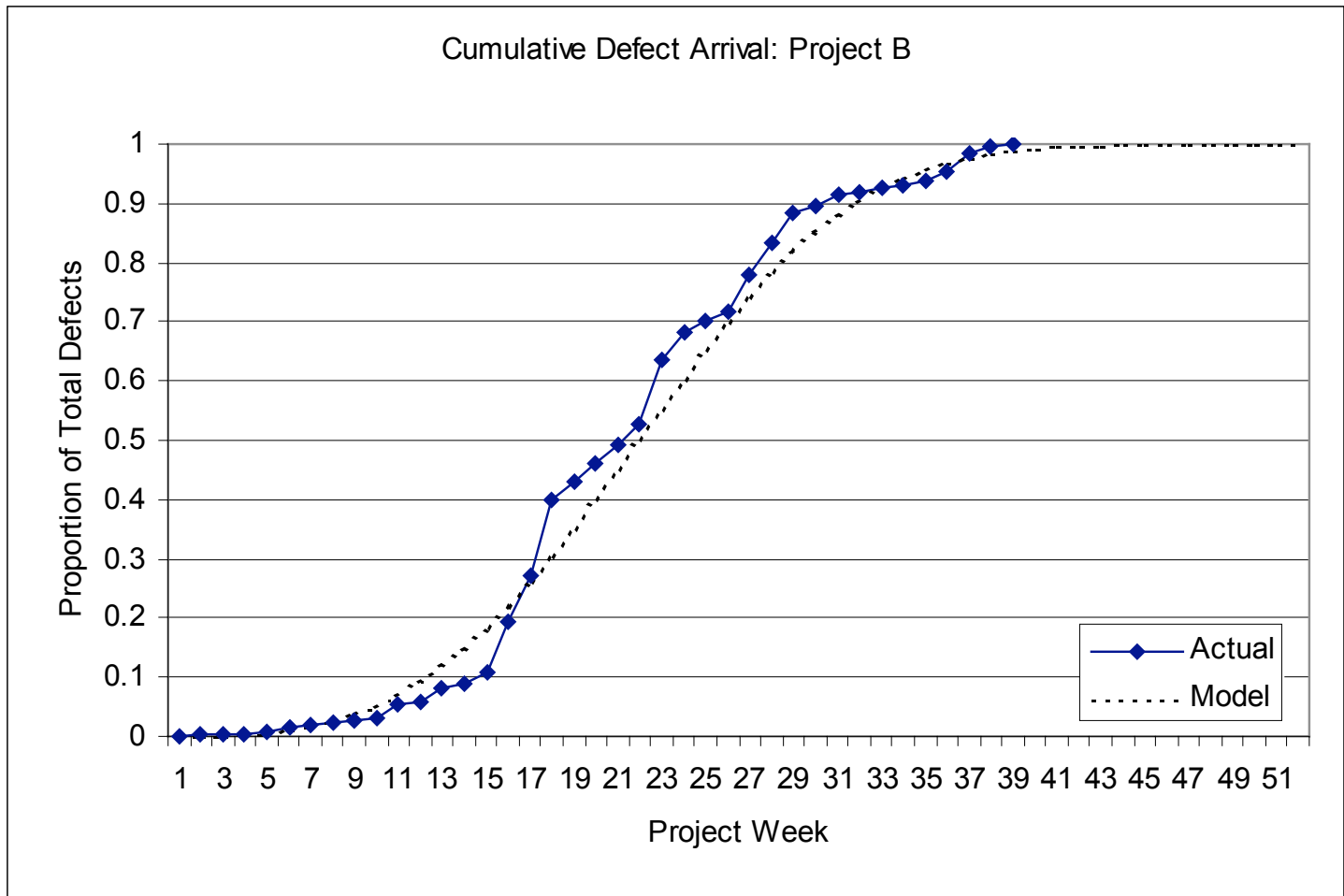


Project B: Weekly Arrival





Project B: Cumulative Arrival





Project C

Traditional two-tiered client/server application

12 developers, 4-7 testers

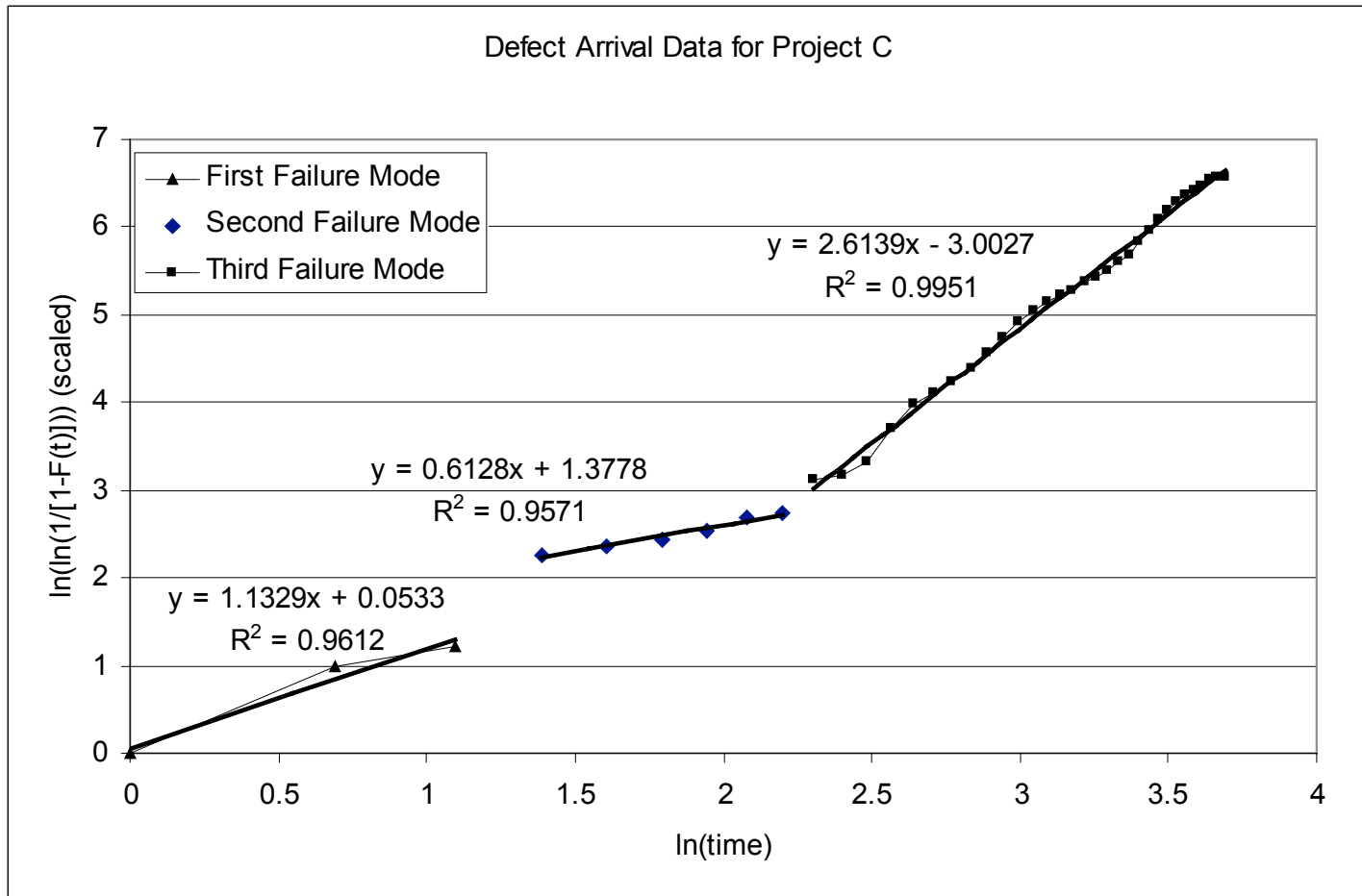
Mix of mostly manual and some automated testing according to a extensive plans and specifications

First Weibull fit to 2000 estimated defects

Total defect estimate raised to 2500 during testing

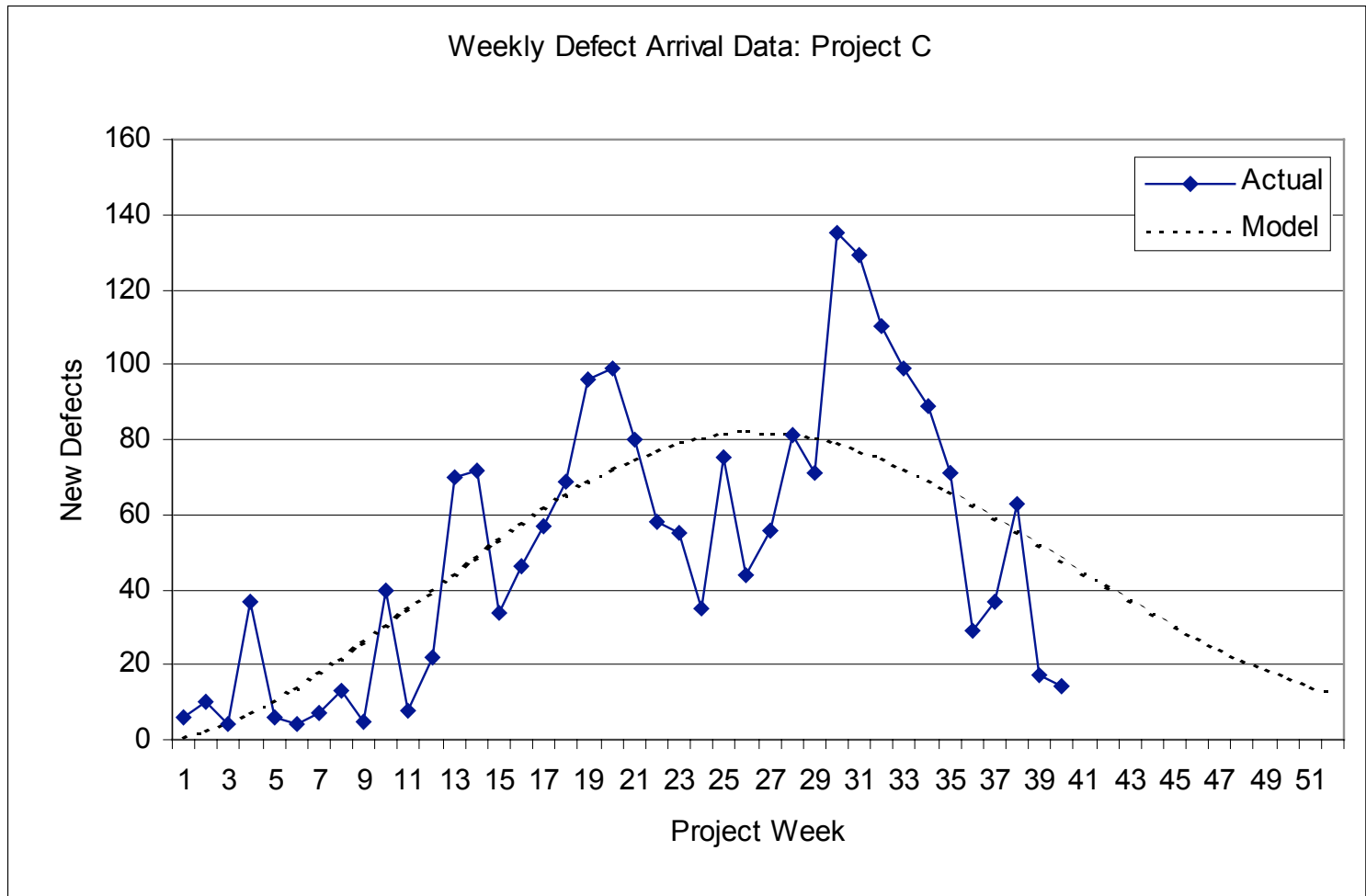


Project C: Weibull Plot





Project C: Weekly Arrival

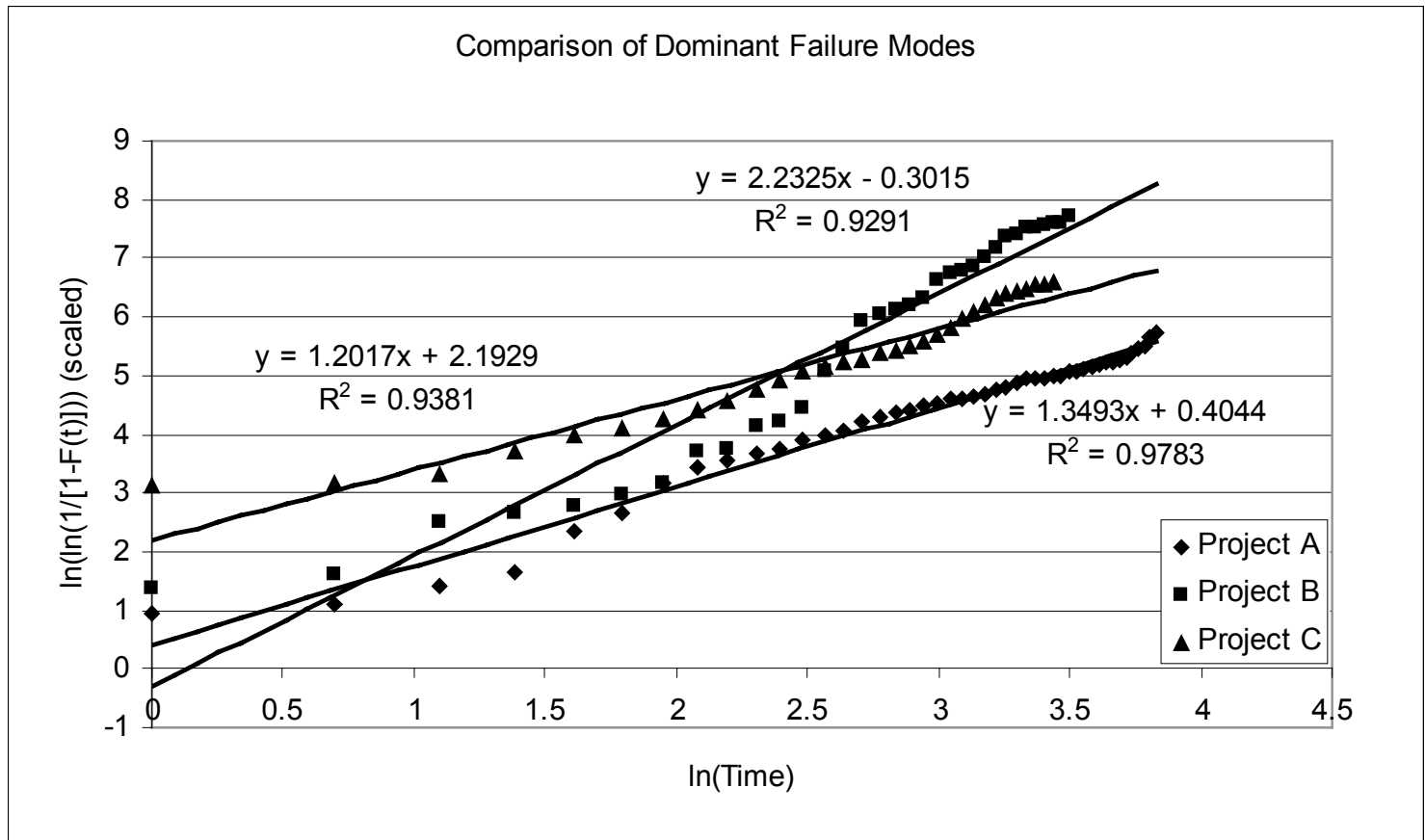




Discussion

- Fits appear to be better when non-cosmetic defects are used
- ‘Infant mortality’ failure is common at the beginning of testing efforts
- Use case when comparing β parameters between projects because of nonlinear axes
- Estimates appear to be stable enough to be useful only a few weeks after the main failure mode appears
- The project’s lifecycle will influence defect arrival patterns

Comparison of Main Failure Modes





Possible Future Work

There are three main areas for more work:

- What factors influence β ?
- Can the need for an estimate of total defects be removed?
- Are results significantly better with execution time domain and/or time between failures data?



References

- Abernethy98 Abernethy, Dr. Robert B., *The New Weibull Handbook, 3rd ed.*, Self-published 1998
- Jones97 Jones, Capers, *Software Quality – Analysis and Guidelines for Success*, Thompson Computer Press 1997
- Kan95 Kan, Stephen H., *Metrics and Models in Software Quality Engineering*, Addison Wesley 1995
- Lyu95 Lyu, Michael (ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill/IEEE Computer Press 1995
- Musa87 Musa, John, *et al.*, *Software Reliability – Measurement, Prediction, Application*, McGraw-Hill 1987
- Putnam92 Putnam, Lawrence, and Myers, Ware, *Measures for Excellence – Reliable Software On Time, Within Budget*, Yourdon Press 1992