Stochastic Modelling: Software Reliability

Y. Eshel, A.V. Giang, N. Toon

Course: XB_0027
January 2021

Abstract

This report estimates the time before a statistics package developed by MathWorks is ready for release. For this purpose statistical tools such as Maximum Likelihood Estimators and multinomial random variables are used.

This reports consists of three parts: (1) developing the model, (2) estimating the model parameters and (3) calculating the time before the software ready for release.

Contents

1 Introduction

This paper aims to develop a statistical model to estimate the testing time needed before a piece of software is ready for release. MathWorks is a software company that develops Matlab and Simulink used for numerical calculations and simulations by many scientists and engineers. Having developed a new statistics package named StatWorks, the manager at MathWorks are interested in the time needed for testing before the software is ready for release. Since the number of bugs in StatWorks is unknown, it is impossible to determine whether the software is truly free of bugs. Having said that, MathWorks needs to release StatWorks knowing that it is sufficiently reliable. As an example, this paper requires that the probability of the software running without failure for 4 CPU hours is 95%, however its parameters can be easily changed to fit other requirements.

In Section 2.1 a simplified version of the model is developed, using the assumption that there are only 2 bugs in the software to develop the intuition for the model. In Section 2.2 an unknown number of bugs u is introduced and some useful equations are derived. Section 3 uses the failure times from Mathworks in Appendix A and the method of Maximum Likelihood Estimators (MLE) to find estimates for the parameters of the model. Lastly, Section 4 introduces some assumptions about the development teams in MathWorks to estimate the time before StatWorks is sufficiently reliable and ready for release.

2 Constructing the model

The aim of a software package is to produce the desired output for a given input. We say that a certain bug caused a failure if a specific input does not produce the correct output. To test the software, the test team at MathWorks

uses randomly generated inputs and checks if the program provides the correct input, otherwise it records the failure time and writes a report for the repair team. This section and the next measures the testing time in CPU seconds, which are the time the processor spends running the tests, neglecting any time spent loading external libraries, verifying outputs or preparing inputs. The model presented assumes that

- There is a finite number u of bugs in the software
- Each bug in the software is independent in equally likely to cause a failure during testing
- Whenever a failure occurs the corresponding bugs is removed without introducing any additional bugs in the process.

2.1 A simple model

Suppose we know for certain that the software contains exactly two bugs, b_1 and b_2. Let B_i denote the time after which bug b_i causes a failure. Then the time it takes to spot the first bug is $T_1 = \min(B_1, B_2)$ and for the second bug $T_2 = \max(B_1, B_2)$. Since each bug is independent and equally likely to cause a failure, B_1 and B_2 will be identically distributed. Let $F(t)$ be their distribution and $f(t)$ their density. Using the Inclusion Exclusion Principle we find that

$$\begin{aligned} 1 - F_{T_1}(t) &= \mathbb{P}(\min(B_1, B_2) > t) \\ &= \mathbb{P}(B_1 > t, B_2 > t) \\ &= 1 - \mathbb{P}(B_1 \leq t) - \mathbb{P}(B_2 \leq t) + \mathbb{P}(B_1 \leq t, B_2 \leq t) \\ &= 1 - 2F(t) + F(t)^2 \end{aligned}$$

where $\mathbb{P}(B_1 \leq t, B_2 \leq t) = F(t)^2$ follows from the independence of B_1 and B_2. Hence the distribution of T_1 is $F_{T_1}(t) = F(t)(2 - F(t))$ and its density is $f_{T_1}(t) = \frac{d}{dt} F_{T_1}(t) = 2f(t)(1 - F(t))$. Similarly the distribution of T_2 is given by

$$\begin{aligned} F_{T_2}(t) &= \mathbb{P}(\max(B_1, B_2) \leq t) \\ &= \mathbb{P}(B_1 \leq t, B_2 \leq t) \\ &= F(t)^2 \end{aligned}$$

and its density is $f_{T_2}(t) = 2F(t)f(t)$.

The above densities are useful for determining the likelihood of the number of failures that occur before some given time t. For example, let M be the number of failures before or at $t = 10$. Then the probability no failure occurring before $t = 10$ is given by

$$\begin{aligned} \mathbb{P}(M = 0) &= \mathbb{P}(T_1 > 10) \\ &= 1 - F_{T_1}(10) \\ &= (1 - F(10))^2. \end{aligned}$$

Similarly, the probability of both failure occurring before $t = 10$ is

$$\begin{aligned}\mathbb{P}(M = 2) &= \mathbb{P}(T_2 \leq 10) \\ &= F_{T_2}(10) \\ &= F(10)^2\end{aligned}$$

and only one failure before $t = 10$ is

$$\begin{aligned}\mathbb{P}(M = 1) &= 1 - \mathbb{P}(M = 0) - \mathbb{P}(M = 2) \\ &= 1 - (1 - F(10))^2 - F(10)^2 \\ &= 2F(10)(1 - F(10)).\end{aligned}$$

The expectation and variance of M are

$$\begin{aligned}\mathbb{E}(M) &= 2F(10)(1 - F(10)) + 2F(10)^2 \\ &= 2F(10)\end{aligned}$$

$$\begin{aligned}\mathrm{Var}(M) &= \mathbb{E}(M^2) - \mathbb{E}(M)^2 \\ &= \mathbb{P}(M = 1) + 4\mathbb{P}(M = 2) - 4F^2(10) \\ &= 2F(10) - 2F(10)^2 + 4F(10)^2 - 4F(10)^2 \\ &= 2F(10)(1 - F(10)).\end{aligned}$$

Everything above suggests that M is a binomial distribution. It is also intuitively evident, since each failure (or trial, in terms of binomial distribution) can either be before or after $t = 10$ (success or failure). Therefore $M \sim \mathrm{Bin}(2, F(10))$.

Let $M(t)$ be the number of bugs found up to some time t. In a similar manner to the abovementioned reason, for a given t, $M(t)$ is binomially distributed with parameters 2 and $F(t)$. One can in fact expand on the reasoning above and instead of dividing the testing time in two, namely $(0, t]$ and (t, ∞), divide it into any number of intervals. We then get a multinomial variable, since each failure can occur in a single interval with a fixed probability. For example, the number of bugs in some interval $(t_1, t_2]$ is given by $M(t_2) - M(t_1)$ with probability $F(t_2) - F(t_1)$. Note that $F(0) = M(0)$, $\lim_{t_2 \to \infty} F(t_2) = 1$ and $\lim_{t_2 \to \infty} M(t_2) = 2$ (the total number of bugs). The fact that a partition of the timeline gives rise to a multinomial random variable is repeatedly used in the rest of this paper.

One of the properties multinomial random variables is that their components are binomially distributed

Lemma 2.1. *If $\mathbf{X} = (X_1, \ldots, X_k)$ is multinomially distributed with parameters n and probability vector $(p_1, \ldots, p_k)$ then each X_i is binomially distributed with parameters n and p_i.*

Proof. Let $\alpha = (x_1, x_2, \ldots, x_k)$ and $\gamma = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$ be *multi-indices*. The marginal density of X_i is given by

$$p_{X_i}(x_i) = \sum_{|\gamma|=n-x_i} \binom{n}{\alpha} p^\alpha \tag{1}$$

Where $p^\alpha = p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}$. From the multinomial theorem we know that

$$\begin{aligned}\sum_{|\gamma|=n-x_i} \binom{n-x_i}{\gamma} p^\gamma &= (p_1 + \cdots + p_{i-1} + p_{i+1} + \cdots + p_k)^{n-x_i} \\ &= (1-p_i)^{n-x_i}\end{aligned}$$

where we used $\sum_{i=1}^k p_i = 1$ and $p^\gamma = p_1^{x_1} \cdots p_{i-1}^{x_{i-1}} p_{i+1}^{x_{i+1}} \cdots p_k^{x_k}$. Factoring $\binom{n}{x_i} p_i^{x_i}$ from (1) and applying the result above we get

$$\begin{aligned}p_{X_i}(x_i) &= \binom{n}{x_i} p_i^{x_i} \sum_{|\gamma|=n-x_i} \binom{n-x_i}{\gamma} p^\gamma \\ &= \binom{n}{x_i} p_i^{x_i} (1-p_i)^{n-x_i}\end{aligned}$$

and so X_i is binomially distributed. □

To illustrate the usefulness of Lemma 2.1, let $t_2 > t_1$. Then $(0, t_1], (t_1, t_2], (t_2, \infty)$ is a partition of the testing time and

$$\mathbf{X} = (X_1, X_2, X_3) = (M(t_1), M(t_2) - M(t_1), 2 - M(t_2))$$

is multinomially distributed with parameters 2 and probability vector

$$(p_1, p_2, p_3) = (F(t_1), F(t_2) - F(t_1), 1 - F(t_2)).$$

Thus

$$\begin{aligned}\mathbb{P}(X_1 = m_1) &= \binom{2}{m_1} p_1^{m_1} (1-p_1)^{2-m_1} \\ &= \binom{2}{m_1} F(t_1)^{m_1} \big(1 - F(t_1)\big)^{2-m_1}\end{aligned}$$

and

$$\begin{aligned}\mathbb{P}(X_2 = m_2 - m_1) &= \binom{2}{m_2 - m_1} p_2^{m_2 - m_1} (1-p_2)^{2-m_2+m_1} \\ &= \binom{2}{m_2 - m_1} \big(F(t_2) - F(t_1)\big)^{m_2 - m_1} \big(1 + F(t_1) - F(t_2)\big)^{2-m_2+m_1}.\end{aligned}$$

Then the probability of a failure occurring in $(t_1, t_2]$ given that m_1 failures occurred in $(0, t_1]$ is

$$\begin{aligned}
\mathbb{P}(X_2 = m_2 - m_1 \mid X_1 = m_1) &= \frac{\mathbb{P}(X_2 = m_2 - m_1, X_1 = m_1)}{\mathbb{P}(X_1 = m_1)} \\
&= \frac{\binom{2}{m_1, m_2 - m_1, 2 - m_2} p_1^{m_1} p_2^{m_2 - m_1} p_3^{2 - m_2}}{\binom{2}{m_1} p_1^{m_1} (1 - p_1)^{2 - m_1}} \\
&= \binom{2 - m_1}{m_2 - m_1} \frac{p_2^{m_2 - m_1} p_3^{2 - m_2}}{(p_2 + p_3)^{2 - m_1}} \\
&= \binom{2 - m_1}{m_2 - m_1} \left(\frac{p_2}{p_2 + p_3}\right)^{m_2 - m_1} \left(\frac{p_3}{p_2 + p_3}\right)^{2 - m_2}.
\end{aligned} \tag{2}$$

Hence the number of failures in the future given that some bugs were repaired follows a binomial distribution.

There is an important relationship between $M(t)$ and T_i which together with the result above allows us to find the probability that a certain number of failures will occur before a given time. If $M(t) \geq i$ then at least i failures have occurred before time t. Since T_i denotes the time at which the ith failures occurs it follows that $T_i \leq t$. Hence

$$M(t) \geq i \quad \iff \quad T_i \leq t.$$

For example, the probability of two failures before t is given by

$$\begin{aligned}
\mathbb{P}(T_2 \leq t) &= \mathbb{P}(M(t) \geq 2) \\
&= \mathbb{P}(M(t) = 2) \\
&= \binom{2}{2} F(t)^2 (1 - F(t))^0 \\
&= F(t)^2,
\end{aligned}$$

while the probability of only one failure is

$$\begin{aligned}
\mathbb{P}(T_1 \leq t) &= \mathbb{P}(M(t) \geq 1) \\
&= \mathbb{P}(M(t) = 1) + \mathbb{P}(M(t) = 2) \\
&= F(t)\big(1 - F(t)\big) + F(t)^2.
\end{aligned}$$

As another example, we can determine the probability of exactly one failure occurring in $(t_1, t_2]$. The second failure can occur either before t_1 or after t_2, and so

$$\begin{aligned}
\mathbb{P}(X_2 = 1) &= \mathbb{P}(X_1 = 0, X_2 = 1, X_3 = 1) + \mathbb{P}(X_1 = 1, X_2 = 1, X_3 = 0) \\
&= \binom{2}{0, 1, 1} p_2 p_3 + \binom{2}{1, 1, 0} p_1 p_2 \\
&= 2p_2(p_3 + p_1) \\
&= 2\big(F(t_2) - F(t_1)\big)\big(1 + F(t_1) - F(t_2)\big)
\end{aligned}$$

Having familiarised ourselves with the assumptions of the model and its different properties, the next section introduces an unknown number of bugs to the model. This has the effect of adding an unknown number of trials to the multinomial distribution described above.

2.2 Adding some bugs

We now change our model to allow for an unknown number of bugs u. One approach to estimate u is to use *bug seeding*. This is done by introducing M artificial bugs which are assumed to be equally likely to cause a failure as the original bugs (this is rather difficult in practice). When testing, out of N failures, M' are caused by the newly introduced bugs while u' are caused by the original ones. Then $\frac{u'}{M'} \approx \frac{u}{M}$ and so $\frac{Mu'}{M'}$ is the approximate number of bugs in the software. However, in practice it is difficult to create bugs that are as difficult to detect as the original bugs. In Section 3 a different method that lacks this drawback is used.

An important measure for the reliability of the software is *failure intensity* which measures the probability of a failure occuring in the near future. Formally, the failure intensity $\lambda(t)$ is given by

$$\begin{aligned}\lambda(t) &= \lim_{\Delta t \to 0^+} \frac{\mathbb{P}(\text{at least one failure in } (t, t+\Delta t))}{\Delta t} \\ &= \lim_{\Delta t \to 0^+} \frac{\mathbb{P}(M(t+\Delta t) - M(t) > 0)}{\Delta t}\end{aligned} \tag{3}$$

Using the partition $(0, t], (t, t+\Delta t], (t+\Delta t, \infty)$ we get the multinomial variable

$$(M(t), M(t+\Delta t) - M(t), u - M(t+\Delta t))$$

with parameters u and probability vector

$$(F(t), F(t+\Delta t) - F(t), 1 - F(t+\Delta t)).$$

By Lemma 2.1 it follows that $M(t+\Delta t) - M(t) \sim \text{Bin}(u, F(t+\Delta t) - F(t))$ and so

$$\begin{aligned}\mathbb{P}\big(M(t+\Delta t) - M(t) > 0\big) &= 1 - \mathbb{P}\big(M(t+\Delta t) - M(t) = 0\big) \\ &= 1 - \big(1 + F(t) - F(t+\Delta t)\big)^u.\end{aligned}$$

Substituting the result above to Equation (3) and applying l'Hopital's rule

$$\begin{aligned}\lambda(t) &= \lim_{\Delta t \to 0^+} \frac{\mathbb{P}(M(t+\Delta t) - M(t) > 0)}{\Delta t} \\ &= \lim_{\Delta t \to 0^+} \frac{1 - (1 + F(t) - F(t+\Delta t))^u}{\Delta t} \\ &= \lim_{\Delta t \to 0^+} u(1 + F(t) - F(t+\Delta t))^{u-1} f(t+\Delta t) \\ &= uf(t).\end{aligned}$$

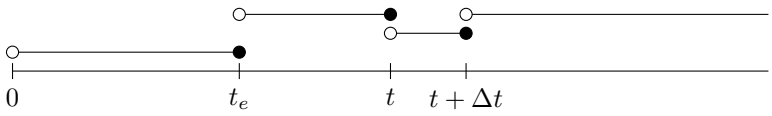

Figure 1: A partition of the timeline

The failure intensity can also be used to find the expected number of failures up to some t. Since $M(t)$ is binomially distributed its expectation is $uF(t)$ which is equivalent to

$$uF(t) = \int_0^t uf(s)\,ds = \int_0^t \lambda(s)\,ds,$$

where we used the fact that $F(0) = 0$.

After testing for t_e CPU seconds, it useful to know the reliability of the software at that point. If we know that m_e bugs were found and fixed at time t_e. Then the conditional probability

$$\mathbb{P}(\text{no failure in } (t, t+\Delta t] \mid M(t_e) = m_e).$$

measures the likelihood of the program to run without failure for some duration Δt. To find this probability we divide our timeline slightly differently this time: $I_1 = (0, t_e], I_2 = (t, t+\Delta t]$ and $I_3 = (t_e, t] \cup (t+\Delta t, \infty)$. An illustration of this partition is found in Figure 1. Thus $M(t_e)$ is the number of failures in I_1, $M(t+\Delta t) - M(t)$ is the number of failures in I_2 and $u - M(t+\Delta t) + M(t) - M(t_e)$ is the number of bugs in I_3. The probability of a failure occurring in each interval can be similarly found. Then

$$\mathbf{X} = (X_1, X_2, X_3) = \big(M(t_e), M(t+\Delta t) - M(t), u - M(t+\Delta t) + M(t) - M(t_e)\big)$$

with parameters u and probability vector

$$(p_1, p_2, p_3) = \big(F(t_e), F(t+\Delta t) - F(t), 1 - F(t+\Delta t) + F(t) - F(t_e)\big)$$

is a multinomial random variable. So

$$\begin{aligned}
\mathbb{P}(\text{no failure in } (t, t+\Delta t] \mid M(t_e) = m_e) &= \mathbb{P}(X_2 = 0 \mid X_1 = m_e) \\
&= \frac{\mathbb{P}(X_1 = m_e, X_2 = 0, X_3 = u - m_e)}{\mathbb{P}(X_1 = m_e)} \\
&= \frac{\binom{u}{m_e, 0, u-m_e} p_1^{m_e} p_2^0 p_3^{u-m_e}}{\binom{u}{m_e} p_1^{m_e} (1-p_1)^{u-m_e}} \qquad (4) \\
&= \left(\frac{p_3}{1-p_1}\right)^{u-m_e} \\
&= \left(1 - \frac{F(t+\Delta t) - F(t)}{1 - F(t_e)}\right)^{u-m_e}.
\end{aligned}$$

is the probability of the software running without failure in the near future.

Let the conditional failure intensity at time $t \geq t_e$ be defined as

$$\lambda(t \mid M(t_e) = m_e) = \lim_{\Delta t \to 0^+} \frac{1}{\Delta t} \mathbb{P}\big(\text{at least one failure in } (t, t+\Delta t] \mid M(t_e) = m_e\big).$$

Recall that

$$\begin{aligned}\mathbb{P}(&\text{at least one failure in } (t, t+\Delta t] \mid M(t_e) = m_e) \\ &= 1 - \mathbb{P}(\text{no failure in } (t, t+\Delta t] \mid M(t_e) = m_e),\end{aligned}$$

and so

$$\begin{aligned}&\lambda(t \mid M(t_e) = m_e) \\ &= \lim_{\Delta t \to 0^+} \frac{\mathbb{P}(M(t+\Delta t) - M(t) > 0)}{\Delta t} \\ &= \lim_{\Delta t \to 0^+} \frac{1 - \left(1 - \frac{F(t+\Delta t) - F(t)}{1 - F(t_e)}\right)^{u - m_e}}{\Delta t} && \text{(By (4))} \\ &= \lim_{\Delta t \to 0^+} (u - m_e) \left(1 - \frac{F(t+\Delta t) - F(t)}{1 - F(t_e)}\right)^{u - m_e - 1} \left(\frac{f(t+\Delta t)}{1 - F(t_e)}\right) && \text{(By l'Hopital)} \\ &= (u - m_e) \frac{f(t)}{1 - F(t_e)}\end{aligned}$$

This quantity is useful since it allows one to find the expected number of failures in the near future given that m_e failures occurred. Let $t = t_e$. Then the partition becomes $(0, t_e], (t_e, t_e + \Delta t], (t_e + \Delta t, \infty)$. Note that

$$\begin{aligned}\mathbb{P}(X_2 = k \mid X_1 = m_e) &= \frac{\mathbb{P}(X_1 = m_e, X_2 = k, X_3 = u - m_e - k)}{\mathbb{P}(X_1 = m_e)} \\ &= \frac{\frac{u!}{m_e! k! (u - m_e - k)!} p_1^{m_e} p_2^k p_3^{u - m_e - k}}{\frac{u!}{m_e!(u - m_e)!} p_1^{m_e} (1 - p_1)^{u - m_e}} \\ &= \binom{u - m_e}{k} \left(\frac{p_2}{p_2 + p_3}\right)^{u - m_e} \left(\frac{p_3}{p_2 + p_3}\right)^{u - m_e - k}\end{aligned}$$

is binomially distributed with parameters $u - m_e$ and $\frac{p_2}{p_2 + p_3}$. Therefore the

conditional expectation is

$$\begin{aligned}\tilde{\mu}(t) &= \mathbb{E}(X_2 \mid X_1 = m_e) \\ &= (u - m_e)\frac{p_2}{p_2 + p_3} && \text{(Expectation of binomial ditribution)} \\ &= (u - m_e)\frac{F(t_e + \Delta t) - F(t_e)}{1 - F(t_e)} \\ &= \int_0^{\Delta t} (u - m_e)\frac{f(t_e + t)}{1 - F(t_e)}\, dt \\ &= \int_0^{\Delta t} \lambda(t_e + t \mid X_1 = m_e)\, dt.\end{aligned} \tag{5}$$

Suppose the management at MathWorks requires that the software will run for 4 CPU hours (14,400 CPU seconds) without failure with probability of 95%. If, for example, at time t_e, $m_e = 90$ out of $u = 100$ bugs were found and fix, how reliable is the software? To find the probability we need to decide on a distribution function. Since the bugs are independent and equally likely to cause a failure, the failures times are independent and occur a constant average rate. Thus letting $F(t) = 1 - e^{-\beta t}$ is the natural choice. Letting $\beta = 10^{-6}$ and plugging all these values to Equation (4) we get

$$\begin{aligned}\mathbb{P}(\text{no failure in } (t_e, t_e + \Delta t] \mid M(t_e) = m_e) &= \left(1 - \frac{F(t_e + \Delta t) - F(t_e)}{1 - F(t_e)}\right)^{u - m_e} \\ &= e^{-0.144} \\ &\approx 0.866,\end{aligned}$$

and so more testing is needed. But how much? Let $\tilde{t}$ be the additional testing time needed for the software to meet the requirements. We know that the probability of no failure in $(t, t + \Delta t]$ with $t = t_e + \tilde{t}$ is given by

$$\left(1 - \frac{F(t_e + \tilde{t} + \Delta t) - F(t_e + \tilde{t})}{1 - F(t_e)}\right)^{u - m_e} = \left(1 - \frac{e^{-\beta(t_e + \tilde{t})} - e^{-\beta(t_e + \tilde{t} + \Delta t)}}{e^{-\beta t_e}}\right)^{u - m_e}.$$

Setting the above equation to 0.95 and solving for $\tilde{t}$ we obtain

$$\tilde{t} = -\frac{\log \frac{1 - 0.95^{\frac{1}{u - m_e}}}{1 - e^{-\beta \Delta t}}}{\beta}. \tag{6}$$

Hence the abovementioned example would require additional $\tilde{t} = 1.02763 \times 10^6$ CPU seconds of testing or about 12 days for the software to meet the requirements.

3 Estimating the parameters

The problem of software reliability lies precisely in the fact that the number of bugs is unknown. For if it was, testing would just continue until all the bugs

are fixed and the software could be bug free and 100% reliable. Therefore this section uses the method of Maximum Likelihood Estimators (MLE) to estimate the value of the parameters in our model, namely u and β. To that end, we start with finding an expression for the probability of observing m_e failures up to some time t_e.

It is given that m_e bugs were discovered in the testing period $(0, t_e]$, let the recorded time of their detection be $t_1, t_2, .., t_e$. The number of bugs m_e can be seen as the greatest integer such that $t_e \leq t_{m_e}$ and $t_{m_e+1} > t_e$. Thus, the probability of observing m_e w.r.t t_i is therefore

$$\mathbb{P}(T_1 = t_1, T_2 = t_2, ..., T_{m_e} = t_{m_e}, T_{m_e+1} > t_e). \tag{7}$$

with T_i be the random variable of the time the i^{th} bug was detected. Because of the continuous nature of T_i, we will assume that the time of bug detection T_i is in an interval $(t_i - \Delta/2, t_i + \Delta/2)$ for small and positive Δt, instead of a single point in time. Thus (7) becomes

$$\begin{aligned}\mathbb{P}(T_1 \in (t_1 - \Delta t/2, t_1 + \Delta t/2), T_2 \in (t_2 - \Delta t/2, t_2 + \Delta t/2), ..., \\ T_{m_e} \in (t_i - \Delta t/2, t_i + \Delta t/2), T_{m_e+1} > t_e).\end{aligned}$$

Using the assumption that during each interval $(t_i - \Delta t/2, t_i + \Delta t/2)$ only one bug occurs, we can transform the above probability into

$$\mathbb{P}(M_1 = 1, M_2 = 1, ..., M_{m_e} = 1, u - M(t_e) = u - m_e)$$

with M_i denoting the number of bugs found in the interval $(t_i - \Delta/2, t_i + \Delta/2)$ and $M(t_i)$ the number of bugs detected up to and including t_i. Evidently, this is the multinomial distribution with probability vector

$$\begin{aligned}(F(t_1 + \Delta t/2) - F(t_1 - \Delta t/2), F(t_2 + \Delta t/2) - F(t_2 - \Delta t/2), ..., \\ F(t_{m_e} + \Delta t/2) - F(t_{m_e} - \Delta t/2))\end{aligned}$$

which can be shortened to

$$(F_1, F_2, .., F_{m_e}, 1 - F(t_e)).$$

It can therefore be easily deduced, using the fact that the time of detection is recorded and the distribution of $(M_1 = 1, M_2 = 1, ..., M_{m_e} = 1, u - M(t_e) =$

$u - m_e$) is multinomial that

$$\begin{aligned}\mathbb{P}(T_1 \in (t_1 - \frac{1}{2}\Delta t, t_1 + \frac{1}{2}\Delta t], T_2 \in (t_2 - \frac{1}{2}\Delta t, t_2 + \frac{1}{2}\Delta t], ..., \\ T_{m_e} \in (t_{m_e} - \frac{1}{2}\Delta t, t_{m_e} + \frac{1}{2}\Delta t], T_{m_e+1} > t_e) \\ = \mathbb{P}(M_1 = 1, M_2 = 1, ..., M_{m_e} = 1, u - M(t_e) = u - m_e) \\ = \binom{u}{1, \ldots, 1, u - m_e} F_1^1 F_2^1 ... F_{m_e}^1 [1 - F(t_e))]^{u - m_e}\end{aligned} \tag{8}$$

Note that this is a rather cumbersome expression and thus, it would be beneficial to simplify and approximate it. Recall that

$$F_i = F(t_1 + \Delta t/2) - F(t_1 - \Delta t/2) = \mathbb{P}(T_i \in (t_1 - \Delta t/2, t_1 + \Delta t/2)) = \int_{t_1 - \Delta t/2}^{t_1 + \Delta t/2} f(s)ds,$$

so $F_i \approx \Delta t f(t_i)$ and (8) now become

$$\begin{aligned}&\frac{u!}{(u - m_e)!} f(t_1)\Delta t f(t_2)\Delta t ... f(t_e)\Delta t \big(1 - F(t_e)\big)^{u - m_e} \\ &= \prod_{i=1}^{m_e} \big((u - i + 1) f(t_i)\Delta t\big) \cdot \big(1 - F(t_e)\big)^{u - m_e}.\end{aligned} \tag{9}$$

As in the previous section, it is assumed that the detection time per bug is exponentially distributed with $F(t) = F_\beta(t) = 1 - e^{-\beta t}$ and consequently, $f(t) = \frac{d}{dt} F_\beta(t) = \beta e^{-\beta t}$. Equipped with a density function and (potentially) known data, one would naturally want to estimate the function's parameters. Then Equation 9 viewed as a function of u and β is likelihood function. For simplicity, we can divide (9) by Δt^{m_e} since under differentiation, said term disappears. Thus

$$\begin{aligned}\mathcal{L}(u, \beta \mid m_e) &= \prod_{i=1}^{m_e} \big((u - i + 1) f_\beta(t_i)\big) \cdot \big(1 - F_\beta(t_e)\big)^{u - m_e} \\ &= \prod_{i=1}^{m_e} \big((u - i + 1)\beta e^{\beta t_i}\big) \cdot \big(e^{-\beta t_e}\big)^{u - m_e}.\end{aligned}$$

Then the log-likelihood is

$$\begin{aligned}\ell(u, \beta) &= \log \mathcal{L}(u, \beta) \\ &= \sum_{i=1}^{m_e} \log(u - i + 1) + m_e \log(\beta) - \beta \sum_{i=1}^{m_e} t_i - \beta t_e (u - m_e).\end{aligned}$$

Under differentiation w.r.t β, we have

$$\frac{d}{d\beta} \ln L(u, \beta) = \frac{m_e}{\beta} - \sum_{i=1}^{m_e} t_i - t_e(u - m_e),$$

setting it to 0 to obtain an estimator for β

$$\hat{\beta} = \frac{m_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u} - m_e)}.$$

Similarly, one can easily find the estimator for u as the solution of

$$\begin{aligned} \sum_{i=1}^{m_e} \frac{1}{\hat{u} - i + 1} &= \hat{\beta} t_e \\ &= \frac{m_e t_e}{\sum_{i=1}^{m_e} t_i + t_e(\hat{u} - m_e)} \text{ (Substituting in the estimator } \hat{\beta}) \end{aligned} \tag{10}$$

Equation (10) is solved numerically using Mathematica in Appendix B.

To test the theories derived thus far, the code given in Appendix B was ran on the table of data in Appendix A which resulted in the following estimators:

$$\hat{u} = 141.007$$
$$\hat{\beta} = 0.0000355835.$$

Using the above values and Equation (6) we can then find the maximum likelihood estimator for the remaining time before the software meets the reliability requirements set by the management. By simply substituting the appropriate values, we find that

$$\tilde{t} = 103,161 \text{ CPU seconds}$$

or about 29 CPU hours.

4 How long before release?

It is more useful to know how much time is left until the release in 'people hours' instead of CPU hours. In reality, every CPU hour of testing may take a while longer, since it requires the testing team to prepare the data, generate a random input and verify the output. Suppose that every CPU hour requires 2.5 people hours and that the testing team consists of 2 people. Hence, every CPU hour costs the company 1.25 people hours. Furthermore, suppose that for every failure an additional 1.5 hours are required to verify it and write a report for the repair team. So it takes 0.75 hours per failure per person. So the expected time spent reporting and verifying failures for a CPU hour of testing is 0.75 times the expected number of failures in an hour. We have shown in Section 2.2 that given $M(t_e) = m_e$ the expected number of bugs in $[t_e, \tau)$ is given by

$$\tilde{\mu}(\tau) = \int_0^\tau (u - m_e)\lambda(t_e + t \mid M(t_e) = m_e)\, dt = \int_0^\tau (u - m_e) \frac{f(t_e + t)}{1 - F(t_e)}\, dt.$$

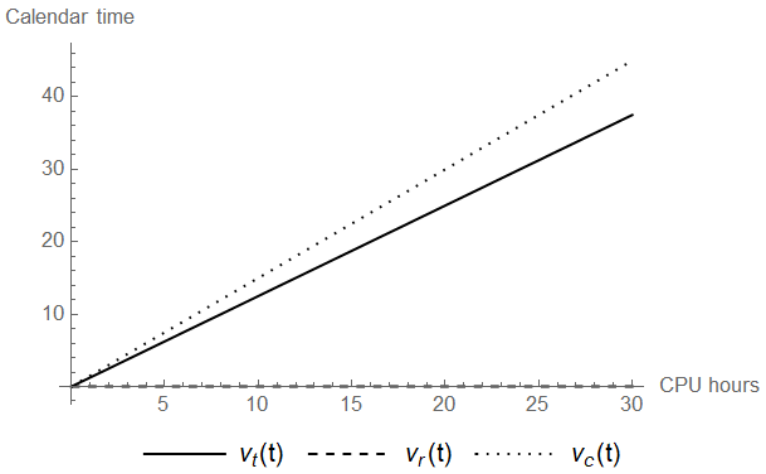

Figure 2: Calendar time vs CPU time for the three teams

Using $F(t) = 1 - e^{-\beta t}$ and $f(t) = \beta e^{-\beta t}$ we get that

$$\tilde{\mu}(\tau) = \int_0^{\tau} (u - m_e) \frac{\beta e^{-\beta(t_e+t)}}{e^{-\beta t_e}} \, dt = (1 - e^{-\beta\tau})(u - m_e)$$

and we can substitute the estimators found in Section 3 (note that β was stated in CPU seconds, but we can simply divide by 3600 to convert it to hours). Then

$$\nu_t(\tau) = 1.25\tau + 0.75\tilde{\mu}(\tau)$$

is the number of people hours it takes the test team for τ CPU hours.

Moving to the repair team, they are only needed whenever there's a failure. Assuming that the repair team takes 6 people hours per bug (for one person) and consists of 2 people we get that

$$\nu_r(\tau) = 3\tilde{\mu}(\tau)$$

Lastly we consider the computers needed. We assume that for every CPU hour the computer runs for 6 hours (running external libraries and background processes). Additionally we assume that every failure costs an additional 0.5 hours. Assuming that 4 computers are available full time we get

$$\nu_c(\tau) = 1.5\tau + 0.125\tilde{\mu}(\tau)$$

At this point we stop for a moment to check that our model thus far makes intuitive sense. According to our estimates the testing team at MathWorks found over 95% of the bugs in the software. Therefore the repair team has little work left to do and most of the testing time is spent on generating inputs and

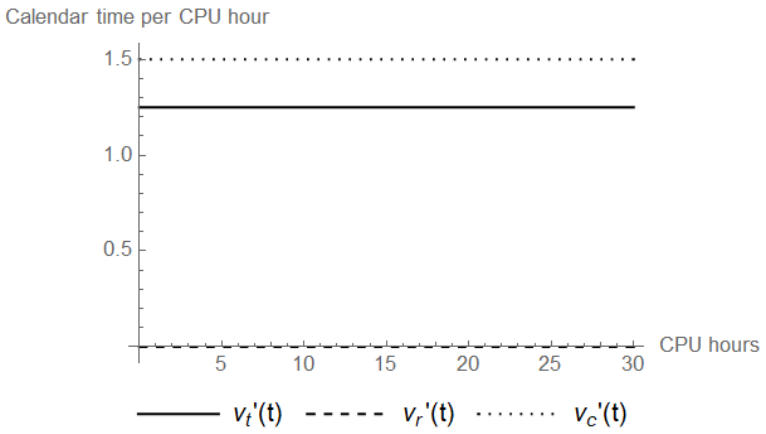

Figure 3: Calendar time per CPU time for the three teams

verifying outputs. Thus the computers running the software will consume the most time, as seen from Figure 2.

The total time that passes during the interval $(0, \tau]$ (where τ is in CPU hours) depends on the load on each team. We assume that the teams are called upon simultaneously which is approximately true. At any given moment, any of the three groups might be the limiting factor. When testing starts, many bugs are discovered rapidly and the testing needs to stop often to give the repair team sufficient time to fix the bugs. Hence they will be the limiting factor. But as the failures become more sparse, the repair team has less to do and ultimately the computer resources (how fast can they test different outputs) become the limiting factor. In any given interval, the team that is the limiting factor is the one that has the highest increase in people hours compared to CPU hours. In other words, the graph with the steepest slope. Hence the amount of people hours elapsed after τ CPU hours is given by

$$\nu(\tau) = \int_0^\tau \max\{v_t'(t), v_r'(t), v_c'(t)\}\, dt$$

where

$$\begin{aligned}
\nu_t'(t) &= -0.75\beta e^{\beta(-t)}\left(e^{\beta t} - 1\right)(u - m_e) + 0.75\beta\,(u - m_e) + 1.25, \\
\nu_r'(t) &= 3\beta\,(u - m_e) - 3\beta e^{\beta(-t)}\left(e^{\beta t} - 1\right)(u - m_e) \\
&\text{and} \\
\nu_c'(t) &= -0.125\beta e^{\beta(-t)}\left(e^{\beta t} - 1\right)(u - m_e) + 0.125\beta\,(u - m_e) + 1.5.
\end{aligned}$$

We know from Section 3 that additional $\tilde{t} = 103214$ CPU seconds of testing are required. Using $\hat{\mu} = 141$, $\hat{\beta} = 0.0000355775$ CPU seconds we find that

$$\nu(\tilde{t}/3600) = 43.0059 \text{ hours}$$

are needed before the software will be ready for release. So supposing it is 1st of February, 2020 at 09:00AM, the software will be ready for release on Monday 8th of February at 11:00AM.

5 Conclusion

Starting with unrealistically low number of bugs, the initial model was generalized to accommodate for a finite, arbitrary u bugs by discretizing testing time into intervals and the number of failures per interval was found to be multinomially distributed. Equipped with said information, it is necessary for businesses to derive (un)conditional reliability measures for the software by the means of calculating the expected number of failures up to a time t given that m_e bugs had already occurred or not. The theory was then applied to the data in Appendix A to find the Maximum Likelihood Estimators for the model's parameters. These results were applied to the situation in MathWorks to determine the release date of StatWorks.

However, the model is not without faults. From the beginning it was assumed that bugs are independent (one bug does not cause another) and any bug can be fixed without the introduction of new bugs. This is rarely true in practice because one bug can cause another and to fix such a bug, an unidentified number of other bugs must be fixed first which can be time-consuming. Furthermore, while fix a given bug it is very well possible that the repair teams introduced a new bug (or bugs) instead. The possibility that the detection time per bug follows a distribution other than exponential was neglected, a fact which may have restricted our options and findings.

A Table of failure times

3	2676	7843	16185	35338	53443
33	3098	7922	16229	36799	54433
146	3278	8738	16358	37642	55381
227	3288	10089	17168	37654	56463
342	4434	10237	17458	37915	56485
351	5034	10258	17758	39715	56560
353	5049	10491	18287	40580	57042
444	5085	10625	18568	42015	62551
556	5089	10982	18728	42045	62651
571	5090	11175	19556	42188	62661
709	5097	11411	20567	42296	63732
759	5324	11442	21012	42306	64103
836	5389	11811	21308	45406	64893
860	5565	12549	23063	46653	71043
968	5623	12559	24127	47596	74364
1056	6080	12791	25910	48296	75409
1726	6380	13121	26770	49171	76057
1846	6477	13486	27753	49416	81542
1872	6740	14708	28460	50145	82702
1986	7192	15251	28493	52042	84566
2311	7447	15261	29361	52489	88682
2366	7644	15277	30085	52875	—
2608	7837	15806	32408	53321	—

,

B Mathematica code

$\mathbf{Clear}[t, u, i, \beta, m, \mathrm{tt}, \mu, v, \mathrm{vt}, \mathrm{vr}, \mathrm{vc}]$

$\mathbf{Subscript}[t, e]=.$

$\mathbf{Subscript}[m, e]=.$

$F[\mathrm{t\_}] = 1 - \mathbf{Exp}[-\beta t]$

$f[\mathrm{t\_}] = F'[t]$

$\mu[\mathrm{t\_}] = \mathbf{Integrate}\left[(u - m_e)\, f[\mathrm{te} + s]/(1 - F[\mathrm{te}]), \{s, 0, t\}\right]$

$\mathrm{vt}[\mathrm{t\_}] = 1.25t + 0.75\mu[t]$

$\mathrm{vr}[\mathrm{t\_}] = 3\mu[t]$

$\mathrm{vc}[\mathrm{t\_}] = 1.5t + 0.125\mu[t]$

$\mathrm{dvt}[\mathrm{t\_}] = \mathrm{vt}'[t]$

$\mathrm{dvr}[\mathrm{t\_}] = \mathrm{vr}'[t]$

$\mathrm{dvc}[\mathrm{t\_}] = \mathrm{vc}'[t]$

$t =$

Drop[

Flatten[

SemanticImport[“failure-times.csv”,Automatic, “Columns”]], −2]

$\mathrm{te} = 91208$

$m_e = \mathrm{Length}[t]$

$\mathrm{dt} = 14400$

$\mathrm{sol} = \mathrm{NSolve}[\mathrm{Sum}\,[1/(u - i + 1), \{i, 1, m_e\}] == m_e \mathrm{te}\,/(\mathrm{Sum}\,[t[[i]], \{i, 1, m_e\}] + \mathrm{te}\,(u - m_e))$

$\&\&u \geq m_e, u, \mathrm{Reals}]$

$u = u/.\mathrm{sol}[[1]]$

$\beta = m_e/\,(\mathrm{Sum}\,[t[[i]], \{i, 1, m_e\}] + \mathrm{te}\,(u - m_e))$

$\mathrm{tt} = -\,(\mathrm{Log}\,[\,(1 - 0.95\hat{}\,(1\,/(u - m_e)))/\,(1 - \mathrm{Exp}[-\beta \mathrm{dt}])])/\,\beta$

Clear[t]

$\beta = \beta/3600$

$d[\mathrm{t\_}] = \mathrm{Max}[\mathrm{vt}'[t], \mathrm{vr}'[t], \mathrm{vc}'[t]]$

$v[\mathrm{t\_}] = \mathrm{Integrate}[d[s], \{s, 0, t\}]$

$v[\mathrm{tt}/3600]$