1-1-2021

How to Estimate Time Needed for Software Migration

Francisco Zapata
The University of Texas at El Paso, fcozpt@outlook.com

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-21-01
Published in *Applied Mathematical Sciences*, 2021, Vol. 15, No. 1, pp. 9-14.

How to Estimate Time Needed for Software Migration

Francisco Zapata[1], Olga Kosheleva[2] and Vladik Kreinovich[3]
[1]Department of Industrial, Manufacturing,
and Systems Engineering
[2]Department of Teacher Education
[3]Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
fcozpt@outlook.com, olgak@utep.edu, vladik@utep.edu

Abstract

In many practical situations, we need to migrate the existing software package to a new programming language and/or a new operating system. In such a migration, it is important to be able to accurately estimate time needed for this migration: if we underestimate this time, we will lose money and may go bankrupt; if we overestimate this time, other companies who estimate more accuracy will outbid us, and we will lose the contract. The formulas currently used for estimating migration time often lead to underestimation. In this paper, we start with the main ideas behind the existing formulas, and show that a deeper analysis of the situation leads to more accurate estimates. Our empirical study shows that the new, more accurate formulas do not suffer from underestimation.

1 Formulation of the Problem

Need for software migration. In many application areas – especially related to business – algorithms remain largely the same for long period of time. However, programming language change, operating systems change. As a result, periodically, there appears a need to migrate the original software to the new language and/or new operating system; see, e.g., [1] and references therein.

Time estimates are important. Many software packages contain millions lines of code. Migrating such hue packages requires a lot of work. So, a company that agrees to perform this task needs to make a good estimate of the time which is needed for such a migration.

If a company underestimates the amount of work, it will have to spend more time – and thus, more money – on this project than originally planned and which was the basis of the agreed-upon contract. As a result, the company will lose money on this project.

If a company drastically overestimates the amount of work – and thus, asks too much money for this job – it will be overbidden by competitors who have better estimates and who can thus agree to fulfil this task much cheaper.

Comment. From the business viewpoint, underestimation is much more dangerous – it can bankrupt the company. Thus, if an accurate estimation is not possible, it is better to have a slight overestimation that an underestimation.

How this time is estimated now. Deep analysis of a large software package requires a lot of time – and this time is wasted if the competitors get the contract. Thus, usually, when time is estimated on the pre-contract stage, only a simple analysis is performed.

Usually, the only parameters which are used for this estimation is the overall length L of the code and the number of modules M in the package. Specifically, in most cases, the time is simply estimated as $c \cdot M$ for some constant c – the average time needed to migrate a module.

Remaining problem. The main problem with the usual method of estimating migration time is that it often leads to underestimation. We thus need a better way to estimate this time.

2 Analysis of the Problem

Simplest case when we only know the overall length of the code. To come up with a better solution to the above problem, let us start with the simplest situation when all we know is the overall length L of the code – e.g., measured by lines of code. We want to estimate the migration time T based on this length L.

Let $f(L)$ denote the time estimate corresponding to length L. In the first approximation, we can approximate the function $f(L)$ by the first two terms in its Taylor expansion:

$$T = a_0 + a_1 \cdot L. \tag{1}$$

Second approximation: taking into account that the to-be-migrated software package consists of modules. Let us now take into account that the software package consists of modules. The time T_i needed to migrate each module of size L_i can be, similarly to the previous subsection, estimated as

$$T_i = a_0 + a_1 \cdot L_i. \tag{2}$$

The overall time T needed to migrate the whole package depends on these times T_i: $T = f(T_1, \ldots, T_n)$. Similarly to the previous section, in the first

approximation, we can approximate this function by its linear terms:

$$T = b_0 + \sum b_i \cdot T_i. \tag{3}$$

In this approximation, we do not have any specific information about different modules – other than their lengths. Thus, we do not have any reason to assume that the coefficients b_i corresponding to different modules are different. Thus, it is reasonable to consider them all equal to each other: $b_1 = b_2 = \ldots$ Under this arrangements, the formula (3) has a simplified form

$$T = b_0 + b_1 \cdot \sum T_i. \tag{4}$$

Substituting the expression (2) into this expression (4), we conclude that

$$T = b_0 + b_1 \cdot a_0 \cdot M + b_1 \cdot a_1 \cdot \sum_i L_i, \tag{5}$$

where by M, we denoted the number of modules.

Here, the sum $\sum\limits_i L_i$ of the lengths the modules is equal to the overall length L of the package. So, the formula (5) takes the following form

$$T = b_0 + c_0 \cdot M + c_1 \cdot L, \tag{6}$$

where we denoted $c_0 \stackrel{\text{def}}{=} b_1 \cdot a_0$ and $c_1 \stackrel{\text{def}}{=} b_1 \cdot a_1$.

This is what is currently used. The formula (6) is what is currently used to estimate the migration time.

To be more precise, what is usually used is simply an expression proportional to M. The length of each module is limited by some constant L_0, so we have $L \le M \cdot L_0$. Thus, from formula (6), we can conclude that

$$T \le b_0 + c \cdot M, \tag{7}$$

where $c \stackrel{\text{def}}{=} c_0 + c_1 \cdot L_0$. The coefficient b_0 is usually set to 0, since if there are no modules ($M = 0$), there is no need to migrate anything, so the migration time is 0:

$$T \le c \cdot M. \tag{7a}$$

Comment. The best estimate of this type can be obtained if we take into account both the number of modules M and the overall length L of the package. But what if we only use one of these numbers? Usually, people use M.

Why not L? In principle, they can also use L, but this would provide a much cruder estimate. Instead, while there is an upper bound L_0 on the length of the module, there is no meaningful lower bound on this length. There can be very short modules. In other words, the lower bound ℓ_0 is very small. So, if we know L, then the only upper bound on M is $M \le L/\ell_0$. Most modules

are much larger, so using this inequality instead of the actual value M would drastically overestimate the migration time. Because of this, practitioners use M and not L if they need to select only one of these numbers for the desired estimation.

Need for the next approximation: reminder. As we have mentioned earlier, the currently used formulas (6), (7), and (7a) often lead to an underestimation of migration time. It is therefore necessary to consider a more accurate description of the migration process than what is provided by the above second approximation.

Next approximation. In the first approximation, we viewed the package as a whole. In the second approximation, we took into account that the package consists of modules. A natural next approximation is when we take into account that modules are usually not used by themselves: for each task, an executable file is usually formed based on several relevant modules. These executable files are formed, e.g., by makefiles.

The same module may be used in forming several different executable files. This is important to take into account, since it is not enough to migrate all the modules one by one: we also need to make sure that migrated modules used in the same executable file are compatible with each other. So, if the same module is used in forming several different executable files, it takes longer to migrate this module: since we need to make sure that its migrated version is compatible with several different groups of modules.

So now, instead of a simple hierarchical structure of the second approximation, when we simply considered a package consisting of modules, we have a more complex structure: the package consists of executable files, and executable files consist of modules. Let us use the same ideas that were used to derive the currently used formulas (6), (7), and (7a), to transform this structure into the desired formula for estimating migration time.

Similarly to the case of second approximation, for each module i, we have an estimate $T_i = a_0 + a_1 \cdot L_i$, and for each executable file a, we have an estimate

$$t_a = b_0 + b_1 \cdot a_0 \cdot M_a + b_1 \cdot a_1 \cdot \sum_{i \in a} L_i, \tag{8}$$

where M_a is the number of modules used in forming the a-th executable file.

The overall migration time T can be obtained from the estimates $t_1, t_2, \ldots$: $T = F(t_1, t_2, \ldots)$. Similarly to the case of the second approximation, we can safely assume that the dependence $F(t_1, \ldots)$ is linear, and that the coefficients at different values t_a in this linear dependence are the same for all a:

$$T = b_0 + b_1 \cdot \sum_a t_a. \tag{9}$$

Substituting the expression (8) into the formula (9), we conclude that

$$T = b_0 + c_0 \cdot \sum_a M_a + c_1 \cdot \sum_a \sum_{i \in a} L_i. \tag{10}$$

The main difference between this formula and the currently used formulas (6), (7), and (7a) is that if the same module is used in forming several different executable files, it leads to a larger contribution to the second and the third terms. This explains why the usual formulas (6), (7), and (7a) often lead to an underestimation: the currently used formulas do not take into account that some modules are used in forming several executable files.

Towards a simplified version of the formula (10). Similar to the case of the currently used second approximation, we can form a simplified version of the formula (10) if we take into account that there is a limit L_0 on the lengths L_i of the modules. Because of this limit, for each executable file a, we have

$$\sum_{i \in a} L_i \le L_0 \cdot M_a,$$

and thus, the formula (10) leads to the following inequality

$$T \le b_0 + c \cdot \sum_a M_a. \tag{11}$$

Taking into account that if there are no modules $M = 0$, there is no need to migrate anything, we get $b_0 = 0$ and

$$T \le c \cdot \sum_a M_a. \tag{11a}$$

3 Resulting Estimates of Migration Time

Estimates. To estimate the time T needed to migrate a software package, we can use the following formula:

$$T = b_0 + c_0 \cdot \sum_a M_a + c_1 \cdot \sum_a \sum_{i \in a} L_i, \tag{10}$$

where M_a is the number of modules that are used to form the a-th executable file, and L_i is the length of the i-th module. This formula leads to a simplified estimate:

$$T \le b_0 + c \cdot \sum_a M_a \tag{11}$$

and

$$T \le c \cdot \sum_a M_a. \tag{11a}$$

Testing. We have tested these formulas. They indeed lead to more accurate estimation of the migration time that the traditionally used formula (6), (7), and (7a), and – in contrast to the traditionally used formulas – do not lead to underestimation of migration time.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes).

References

[1] G. Blokdyk, *Software Modernization: A Complete Guide*, The Art of Service, 2020.