INTECH
open science | open minds

OPEN ACCESS
ARTICLE

International Journal of Engineering Business Management

Estimating Productivity of Software Development Using the Total Factor Productivity Approach

Regular Paper

Machek Ondrej[1,*], Hnilica Jiri[2] and Hejda Jan[3]

1 University of Economics, Prague, Department of Business Economics, Faculty of Business Administration
2 University of Economics, Prague, Department of Business Economics, Faculty of Business Administration
3 Czech Technical University in Prague, Department of Biomedical Technology, Faculty of Biomedical Engineering
* Corresponding author E-mail: ondrej.machek@vse.cz

Received 2 July 2012; Accepted 27 August 2012

DOI : 10.5772/52797

Abstract The design, control and optimization of software engineering processes generally require the determination of performance measures such as efficiency or productivity. However, the definition and measurement of productivity is often inaccurate and differs from one method to another. On the other hand, economic theory offers a well-grounded tool of productivity measurement. In this article, we propose a model of process productivity measurement based on the total factor productivity (TFP) approach commonly used in economics. In the first part of the article, we define productivity and its measurement. We also discuss the major data issues which have to be taken into consideration. Consequently, we apply the TFP approach in the domain of software engineering and we propose a TFP model of productivity assessment.

Keywords total factor productivity, software development productivity, productivity of processes

1. Introduction

In economics, a producer is defined as an agent transforming a set of inputs into a set of outputs and the production process is defined by the production function. A similar concept is used in engineering, where a process is defined as a set of interrelated tasks that, together, transform inputs into outputs [1]. The design, control and optimization of engineering processes generally require determination of performance measures such as efficiency or productivity. However, many models of measuring process performance have usually been derived from empirical observations and have found to be inaccurate.

Let's take software development processes as an example. Several past studies, such as [2], [3] or [4] confirmed a fairly large magnitude of relative errors of the estimates of the models. In the meanwhile, the similarity between engineering processes and economic production processes has been noticed by some researchers. Hu, Plant and Hertz [5] proposed a model for software cost

estimation based on the economic production theory which proved to be more accurate than traditional COCOMO [6] or SLIM [7] cost estimation models.

Measuring productivity goes beyond cost estimation since it captures not only costs, but also revenues. Total factor productivity (TFP) is a method of measuring overall productivity of economies, industries or individual firms. However, since productivity is not a property which is limited to aggregated entities, TFP could also be used on a lower level of abstraction, in business or engineering processes which consist of a series of activities.

The aim of this article is to propose the use of total factor productivity methods in the productivity evaluation of software development processes. In the first part of the article, we define productivity and its measurement and we introduce total factor productivity and indexes commonly used in measuring TFP changes. We also discuss the major data issues in the TFP approach. Consequently, we apply this approach on the process of software development. In the final part, we suggest a TFP model of software development productivity analysis.

2. Total factor productivity and its measurement

Traditionally, productivity is defined as a ratio of output over input. The total factor productivity (TFP) approach takes into account all outputs and inputs of the evaluated firm. Therefore, it is necessary to aggregate the set of outputs and outputs so that productivity becomes a scalar value.

In practice, TFP change is measured by productivity indexes or productivity indicators. Indexes have multiplicative form, whereas indicators have additive form. TFP index numbers can be based either on distance function or on price aggregation. Among measures based on distance function, we can cite the Malmquist productivity index [8] and the Hicks-Moorsteen productivity index [9]. The most common TFP measures which are based on price aggregation are the Törnqvist productivity index [10] and the Fisher productivity index [11]. The indexes based on distance functions are rather theoretical and require larger sets of cross-sectional data. By contrast, the cost-based indexes are quite straightforward to calculate and necessitate a minimum of only two observations.

Indexes are a common tool in economics to measure price or quantity changes between two periods, the most known examples being the consumer price index (CPI) or Dow Jones stock index. Since in TFP calculations we deal with the ratio of output and input quantities, we will use quantity indexes.

The calculation is based on observed or estimated quantities and prices of input and output factors. In the following text, let $\mathbf{x} = (x_1,x_2,\ldots,x_M)$ denote the input quantities, let $\mathbf{y} = (y_1,y_2,\ldots,y_N)$ denote the output quantities, $\mathbf{w} = (w_1,w_2,\ldots,w_M)$ is the vector of input prices, $\mathbf{p} = (p_1,p_2,\ldots,p_N)$ is the vector of output prices, M the number of inputs and N the number of outputs.

Let us first introduce the **Fisher index of productivity** [11]. It is a geometric average of the Laspeyres and Paasche indexes.

The **Laspeyres index** weights the quantities with the prices of the basic period. We can specify the Laspeyres output quantity index or input quantity index, respectively as

$$Y_L = \frac{\sum_{n=1}^{N} p_{n,t}\, y_{n,t+1}}{\sum_{n=1}^{N} p_{n,t}\, y_{n,t}}, \text{ resp. } X_L = \frac{\sum_{m=1}^{M} w_{m,t}\, x_{m,t+1}}{\sum_{m=1}^{M} w_{m,t}\, x_{m,t}} \qquad (1)$$

The **Paasche index** weights the quantities with the prices of the current period. The Paasche output (input) quantity index can be specified as

$$Y_P = \frac{\sum_{n=1}^{N} p_{n,t+1}\, y_{n,t+1}}{\sum_{n=1}^{N} p_{n,t+1}\, y_{n,t}}, \text{ resp. } X_P = \frac{\sum_{m=1}^{M} w_{m,t+1}\, x_{m,t+1}}{\sum_{m=1}^{M} w_{m,t+1}\, x_{m,t}} \qquad (2)$$

Since productivity is defined as a ratio of output over input, it is possible to define the Fisher index as the ratio of geometric averages of Laspeyres and Paasche output and input indexes, so

$$\Pi_F = \frac{Y_F}{X_F} = \frac{\sqrt{Y_L Y_P}}{\sqrt{X_L X_P}} \qquad (3)$$

The **Törnqvist index of productivity** [10] is defined as a ratio of output quantity index Y_T and input quantity index X_T. It is a weighted geometric average of quantity relatives. Usually, the two quantity indexes are specified in their logarithmic form as

$$\ln Y_T = \frac{1}{2}\left(\sum_N \left[\frac{p_{n,t}\, y_{n,t}}{\sum_N p_{n,t}\, y_{n,t}} + \frac{p_{n,t+1}\, y_{n,t+1}}{\sum_N p_{n,t+1}\, y_{n,t+1}}\right] \ln \frac{y_{n,t+1}}{y_{n,t}}\right) \qquad (4)$$

$$\ln X_T = \frac{1}{2}\left(\sum_M \left[\frac{w_{m,t}\, x_{m,t}}{\sum_M w_{m,t}\, x_{m,t}} + \frac{w_{m,t+1}\, x_{m,t+1}}{\sum_n w_{m,t+1}\, x_{m,t+1}}\right] \ln \frac{x_{m,t+1}}{x_{m,t}}\right)$$

Finally, the Törnqvist index of productivity can be specified as

$$\Pi_T = \frac{Y_T}{X_T} = e^{\ln Y_T - \ln X_T} \qquad (5)$$

If the prices are not directly observable, it is possible to approximate them using expert estimates or econometric methods such as non-linear regression. Fisher and Törnqvist indexes have several interesting properties, because of which they are classified as exact and superlative indexes (more on this subject in [12]). If the production is represented by a translog function, the Törnqvist index approximates the ideal Malmquist index and similarly, if the production is represented by quadratic function, the Fisher index approximates the ideal Malmquist index. Accidentally, a study [13] found that translog and quadratic cost functions are more suitable for representation of software production processes than the traditionally considered linear or Cobb-Douglas functional forms.

The above defined indexes are suitable for measuring and analysing productivity of a single software development process. When comparing productivity across multiple subjects or processes, the indexes have to satisfy the property of transitivity. However, the Fisher and Törnqvist indexes themselves are not transitive. An example of procedures of the conversion of non-transitive indexes into transitive indexes is the procedure proposed by Caves, Christensen and Diewert (CCD) [8] and is well described in [14]. If I denotes the number of compared processes and I_{ij}^{T} denotes the Törnqvist index for pair i,j, then the transitive indexes can be obtained as

$$I_{ij}^{CCD} = \prod_{k=1}^{I} \left(I_{ik}^{T} \times I_{kj}^{T} \right)^{1/I} \qquad (6)$$

3. Data issues in measuring TFP changes

The measurement of productivity using the TFP approach involves several data issues which substantially affect the results.

3.1 Measurement of Outputs

The definition and measurement of outputs and their prices is one of the challenging tasks. Outputs should represent the complete basket of services and products provided by the transformation process. They should reflect how much is being produced, with what effort, and they should not omit the quality of service. The definition of the quality aspect is particularly challenging. Moreover, one of the main concerns is that if the set of outputs is too large, one may encounter the problem of degrees of freedom and the analysis becomes complicated. In this case, it may be suitable to aggregate the outputs into smaller categories, in which case the so-called Hicks–Leontief conditions for aggregation should be respected ([15], [16]).

The prices of outputs should also be treated carefully and price level changes should not be omitted. If the prices are not directly observable, it is necessary to approximate their weights in the total revenue and derive the prices numerically. These weights are calculated from the share of each output in the total revenue of the process. This approach involves either arbitrary judgments about the relative importance or econometric estimation of cost function (see e.g. [17]). There is an academic debate over which of these approaches performs the best. However, when prices of outputs are not directly observable, some degree of inaccuracy is practically inevitable.

3.2 Measurement of Inputs

Another challenging task is to define and measure accurately the inputs and their prices. Traditionally, the economic theory takes into account at least the following categories of input factors: labour (L), capital (K) and materials (M). Sometimes, within materials, energy (E) and services (S) are also considered, and all these factors together are referred to as KLEMS.

Labour (L) is most often measured by the number of employees or man-hours, which should be corrected, since outsourcing of activities can distort the results. Moreover, it is preferable to distinguish among the people according to their skills, education and experience, since more skilled employees contribute to the productivity growth to a greater extent. Because of these difficulties, labour is sometimes incorporated into operating expenditures (OPEX) which are taken together as an aggregate measure of labour and materials. However, labour costs often represent the major portion of total input costs and therefore, this input should be treated carefully in order to obtain reliable TFP estimates. Wage deflators can be taken into consideration to capture the effects of wages inflation.

However, the most contentious input factor is the measurement of capital (K). The capital input should reflect the total service flow from capital assets used in the process. The assets can be of a tangible or intangible nature such as computers, software in IT, chemical reactors in chemistry production processes, transport equipment, heavy machinery, etc. Of course, the set of assets will vary a lot across industries and even company departments. The capital can be measured directly, in physical units, or indirectly in money value. Both approaches have their advantages and disadvantages; they are discussed in detail in [17].

The productive capital stock can be measured by the perpetual inventory method proposed by OECD [18]. If we denote the productive capital stock by K_t^P, then

$$K_t^P = \sum_{\tau=0}^{T} h_\tau F_\tau \frac{IN_{t-\tau}}{q_{t-\tau,0}} \quad (7)$$

where h_τ is an age-efficiency profile, tracing the loss in productive efficiency as an asset ages, taking values between 1 (when an assets is new) and 0 (when it has lost its entire productive capacity). F_τ is a retirement function that quantifies the share of assets of age τ that are still in service. This function is declining and takes values between 1 (when all assets are in existence) and 0 (when all assets have been retired). IN_τ is the nominal investment expenditure on the asset at time τ, which is deflated by an investment price index $q_{t-\tau,0}$ where subscripts indicate a price index for the asset of age zero (a new asset) in year τ. Following the same manual, the cost of capital (rental price) can be determined using the following formula:

$$\mu_t = q_t(r_t + d_t) - (q_t - q_{t-1}) \quad (8)$$

where q_t is the market price of a new asset, d_t is the depreciation rate and r_t is some measure of the cost of financial capital such as the market rate of interest. However, it is possible to approximate the cost of capital inputs using regression methods following [17].

4. Productivity in engineering applications: software project management

Software development is clearly an example of process transforming a set of inputs into a set of outputs. However, some past studies, such as [19], found this process to be notorious due to cost overruns and time delays. Many scientists have tried to measure productivity of software development projects. In conformity with the traditional definition of productivity, the various methods compare the ratio of outputs and inputs. Generally, however, the methods do not take into account all the outputs and all the inputs which are involved in the analysed process as required by the total factor productivity approach.

Recently, a study [20] summarized the methods of measuring software development team productivity. Most of the authors ([21], [22], [23], [24]) considered only very simplistic measures of outputs and inputs.

For instance, the output is traditionally represented by lines of code (LOC) or function points (FP). But this simple measure does not capture either quality or other possible outputs, such as reusability of the produced code, its complexity, etc. which are not evident at first sight.

Similarly, the set of inputs is often reduced to man-hours of labour. Such a measure is nothing less than a simple labour productivity – a partial productivity measure; furthermore, these man-hours are typically not differentiated according to the knowledge and skills of development team members.

In addition, capital, materials and services are generally omitted from the productivity analysis, but development software – a typical intangible asset which is subject to amortization – represents an important input of the process; the choice of programming language as well as programming techniques clearly affect not only the number of LOC or FP, but also their quality. Usually, depreciation of hardware resources and other materials, such as energy, are considered to be negligible.

We believe that these factors could represent a source of inaccuracy of the measurement; the negligence of quality of outputs, as well as minor inputs, possibly biases the productivity estimates. The total factor productivity approach takes into account all outputs and all inputs.

Some more recent methods described in [20] take into consideration more sophisticated sets of outputs and inputs. Besides mathematical models based on regression techniques (see for example [25]), in their more recent works (i.e., [26], [27] or [28]) the authors have included the quality aspect, as well as tools and other resources.

5. TFP model of software development projects

TFP indexes can easily be estimated using common computing tools such as MS Excel or MATLAB. In this section, we propose a model of software development project productivity measurement based on the TFP approach.

5.1 Output Definitions

Traditionally, the outputs of software development are measured by lines of code (LOC) or function points (FP). These two measures are highly correlated, but additionally, the TFP approach takes into account the complexity (see e.g., [29]).

To incorporate both the quantity and quality aspects, we propose considering code reuse, complexity, functionality and length following [28]. Reuse is measured by the estimated percentage of reused object points; complexity is measured using the big-O notation familiar in computer science, functionality by the function points count and length by taking into consideration the density of comments. Besides these measures of code, we propose incorporating other outputs: data, documentation and training, as proposed by [27]. To sum up, we propose the following set of outputs:

- x_1: Reuse (percentage of reused object points)
- x_2: Complexity (delivered source instructions)
- x_3: Functionality (function points count)

- x_4: Length (LOC corrected with respect to the density of comments)
- x_5: Data (lines of produced useful data)
- x_6: Documentation (lines of produced documentation)
- x_7: Training (number of trained persons)

5.2 Input Definitions

Most of the existing approaches take into account only the labour input. However, it is desirable to differentiate the staff according to their skills. Following [27], we can consider at least four categories of labour - engineering, testing, management and support. To avoid a simple labour productivity calculation, it is necessary to include other input factors. One may consider the productive capital stock (hardware, software and licences) and materials and services, which can be aggregated together, since they represent only a minor portion of software development costs. To summarize, we propose the following set of inputs:

- y_1: Engineering labour (man-hours)
- y_2: Testing labour (man-hours)
- y_3: Management labour (man-hours)
- y_4: Support labour (man-hours)
- y_5: Materials and services (deflated value)
- y_6: Tangible capital stock (deflated value)
- y_7: Intangible capital stock (deflated value)
- y_8: Other capital (deflated value)

5.3 Output and Input Prices Determination

The prices of inputs and outputs are likely not to be directly observable. In this case, it is necessary to estimate them. As mentioned before, the estimation involves either arbitrary judgment or econometric methods. In our model, we follow [17] and use both methodologies.

The prices of inputs can be estimated using the former approach. Labour costs, materials and service value are usually easy to measure, as well as the total revenue of the project. The input weight given to labour is simply the ratio of the labour costs to total revenue of the project, as well as the materials and services ratio. The aggregate weight of the rest of inputs is given by one minus labour share minus materials and services share. This aggregate weight is then divided by the approximate shares of the capital categories in the total value of the assets.

The total project costs are then divided according to the weights to the input categories. By dividing the category costs by the number of appropriate units of input, we determine the vector of costs of inputs **w**.

Once the costs of inputs are calculated, it is possible to estimate the costs of outputs using regression methods. Following [17], we use the Leontief multi-output cost function which takes the form

$$C(y^t, w^t, t) = \sum_{i=1}^{M} w_i^t \left[\sum_{j=1}^{N} \left(a_{ij}\right)^2 y_j^t \left(1 + b_i t\right) \right] \quad (9)$$

where M denotes the number of inputs, N denotes the number of outputs, w_i is an input price, y_j is an output quantity and t is a time trend representing technological change (the number of year). Using the Shephard's lemma known in economics we can derive the input demand equations

$$x_i = \sum_{j=1}^{N} \left(a_{ij}\right)^2 y_j^t \left(1 + b_i t\right) \quad (10)$$

By applying a standard non-linear regression to these equations we obtain the input-output coefficients a_{ij} and the coefficients b_i. Now we can estimate the output weights as

$$h_j^t = \frac{\sum_{i=1}^{M} w_i^t \left[\left(a_{ij}\right)^2 y_j^t \left(1 + b_i t\right) \right]}{\sum_{i=1}^{M} w_i^t \left[\sum_{j=1}^{N} \left(a_{ij}\right)^2 y_j^t \left(1 + b_i t\right) \right]} \quad (11)$$

By dividing the total revenue according to the weights and number of outputs, we determine the costs of outputs **p**.

5.4 Total Factor Productivity Calculations

Once we have defined and measured the outputs **y**, inputs **x,** and their costs **p** and **w**, we can easily calculate the Fisher or Törnqvist productivity indexes as described in the previous sections. These indexes can be used to assess and evaluate productivity changes, and their sources. In the case of multilateral comparisons, i.e., when comparing among multiple projects or entities, it is desirable to convert the indexes into their transitive form, for example, using the method described in [14].

6. Conclusion

Reliable estimation of the productivity of software development process is an important task for software engineering practitioners and academics. The total factor productivity approach is a well-grounded and established methodology frequently used in economics. It is most often used at the aggregate level; however, TFP is not limited to whole economies or industries, but it can also be used in assessing productivity of companies, departments and business processes. In this article, we proposed a model of software development project productivity measurement based on the TFP approach.

In the first part of the article, we defined total factor productivity and its measurement. We suggested the use of the Törnqvist and Fisher quantity indexes to measure

the TFP changes across time periods. When comparing multiple software development processes, the transitive versions of the indexes should be used.

In the following section, we dealt with the major data issues which have to be taken into consideration when measuring inputs, outputs and their prices.

Consequently, we discussed the issues of measuring productivity in engineering applications, in particular in software project management. We found out that most of the currently used methods are in fact simple labour productivity measures and do not include all inputs and outputs that participate in the development process.

In the final part, we suggested a TFP model of software development productivity analysis.

In order to capture all the relevant outputs and the quality, we proposed including the rate of code reuse, the code complexity, functionality and length. Furthermore, we suggested considering other outputs of the software development process, namely data produced, documentation and training of persons.

Concerning the inputs, we proposed to differentiate the labour input according to the function of the development team functions: at least, engineering, testing, management and support labour force should be included in the analysis. Besides labour, we prefer to consider other input factors: materials and services, tangible and intangible assets, and other capital. All these factors together reflect the complete set of classical input factors considered in economic theory.

Finally, we described the process of the determination of input and output prices in case they are not directly observable. Both expert estimates and regression methods can be used.

An accurate estimation of productivity changes could help software engineers identify strengths and weaknesses of development processes, and improve the useful output of the process which, in turn, could be accomplished with less effort. However, empirical evidence is needed, which could be the focus of the future research on this subject.

7. Acknowledgments

This article was written with financial support from the Internal Grant Agency of the University of Economics in Prague, project no. F3/22/2011 "Regulation of energy utilities in Central Europe and the possibilities for improvement."

8. References

[1] TechAmerica (1999) ANSI-EIA-632 Standard: Processes for Engineering a System. TechAmerica.
[2] Mohanty S (1981) Software cost estimation: Present and future. Software: Practice and Experience 11: 103-121.
[3] Kemerer C.E (1987) An empirical validation of software cost estimation models. Communications of the ACM 30: 416-429.
[4] Jorgensen M (1995) Experience with the accuracy of software maintenance task effort prediction models. IEEE Transactions on Software Engineering 21.
[5] Hu Q, Plant R.T, Hertz D.B (1998) Software cost estimation using economic production models. Journal of Management Information System 15: 143-163.
[6] Boehm B.W (1981) Software Engineering Economics. New York: Prentice Hall.
[7] Putnam L.H (1992) Measures for Excellence: Reliable Software on Time, within Budget. New York: Yourdon Press.
[8] Caves D, Christensen L, Diewert W.E (1982) Multilateral comparisons of output, input, and productivity using superlative index numbers. Economic Journal 92: 73-86.
[9] Diewert W.E (1992) Fisher ideal output, input, and productivity indexes revisited. Journal of Productivity Analysis 3: 211-248.
[10] Törnqvist L (1936) The bank of Finland's consumption price index. Bank of Finland Monthly Bulletin 10: 1-8.
[11] Fisher I (1922) The Making of Index Numbers. Boston: Houghton-Mifflin.
[12] Diewert W.E (1976) Exact and superlative index numbers. Journal of Econometrics 4: 115-145.
[13] Hu Q (1997) Evaluating alternative software production functions. IEEE Transactions of Software Engineering 23.
[14] Coelli T.J, Prasada Rao D.S, O'Donnell C. J, Battese G.E (2005) An introduction to Efficiency and Productivity Analysis. New York. Springer.
[15] Leontief W (1936) Composite commodities and the problem of index numbers. Econometrica 4: 39-59.
[16] Hicks J (1939) Value and Capital. Oxford: Clarendon Press.
[17] Lawrence D, Diewert W.E (2006) Regulating electricity networks: The ABC of setting X in New Zealand. In: Coelli T, Lawrence D., editors. Performance Measurement and Regulation of Network Utilities. Northampton: Edward Elgar, pp. 207-241, 2006.
[18] OECD (2001) Measurement of aggregate and industry-level productivity growth. Paris: OECD.
[19] Jenkins A.M, Naumann J.D, Wetherbe J.C (1984) Empirical investigation of systems development practices and results. Information and Management 7.

[20] Sudhakar G.P, Farooq A, Patnaik S (2012) Measuring productivity of software development team. Serbian Journal of Management 7: 65-75.
[21] Tausworthe R.C (1982) Staffing implications of software productivity models. Proc. 7th Annual Software Engineering Workshop, NASA/Goddard, Greenbelt.
[22] Banker R.D, Datar S.M, Kemerer C.F (1991) A model to evaluate variables impacting the productivity of software maintenance projects. Management Science 37.
[23] Potok T.E, Vouk M.A. (1999) A model of correlated team behavior in a software development environment. Proceedings of IEEE Symposium on Application-specific Systems and Software Engineering and Technology, ASSET' 99, Richardson.
[24] Blackburn J.D, Lapre M.A, Van Wassenhove L.N (2002) Brooks' law revisited: Improving software productivity by managing complexity. Vanderbilt University Working paper.
[25] Tockey S (2000) The effect of team size on team productivity and project cost. Lecture notes of software project management, CSSE-515, Seattle University.
[26] Krishnan M.S, Kriebel C.H, Kekre S, Mukhopadhyay T (2000) An empirical analysis of productivity and quality in software products. Management Science 46: 745-759.
[27] Card D.N (2006) The Challenge of Productivity Measurement. Proceeding of Pacific Northwest Software Quality Conference, Portland.
[28] Nwelih E, Amadin I.F (2008) Modeling software reuse in traditional productivity model. Asian Journal of Information Technology 7: 484-488.
[29] Capers J (1994) Function points: A new way of looking at tools. Computer 27: 66-67.