Resource Allocation among Development Phases: An Economic Approach

Peleg Yiftachel, Dan Peled, Irit Hadar and Dan Goldwasser
{peleg@cs, dpeled@econ, hadari@is, dgoldwas@cs}.haifa.ac.il
University of Haifa, Mount Carmel, Haifa 31905, Israel

ABSTRACT
This study presents an economic model for solving the SW manufacturing problem of resource allocation among the different development phases. The suggested model takes into consideration the allocation effects on SW quality, development cost and customers willingness to pay for quality. The model suggests a bi-level definition of SW manufacturer output. The first level considers the quality of the different phases artifacts, (internal quality), while the integration of all these qualities into what is consumed by the customer constitutes the second level (external quality). We illustrate how the model can be applied in an experimental setting, and describe the tools developed for such applications. This illustration lays the basis for future empirical studies evaluating the model's different functions and ability to improve SW development processes.

Categories and Subject Descriptors
D.2.8 [Software Engineering]: Metrics; D.2.9 [Software Engineering]: Management

General Terms

Keywords Software resource allocation, software quality, software development phases, software production function.

1. INTRODUCTION
Disagreements and conflicts about the appropriate effort to be devoted to requirement analysis and design before implementation work should begin, or about the necessity of further SW testing before release, are widespread among SW developers. These conflicts are often attributed to resource constraints and time-to-market considerations, and as such, appear to reflect reasonable economic tradeoffs. It is well recognized that a more thorough work on requirements and design can reduce future development costs, and may result in a higher quality of the developed SW (such as [4], [9], [10]). However, the pressure to have an operational product sooner rather than later can frequently affect the resource allocation among development phases. It is commonly believed that a significant share of development costs can be saved by more careful allocation of work among the development phases. The NIST [11] graphically displays the well-known structure of relative costs of bugs according to the phase in which they are detected, demonstrating the dramatic increase of such costs as defects detection occurs later in the development process. However, we are not aware of any study that examines how resources should be optimally allocated across development phases to avoid these dire costs implications. It is estimated that tens of billions dollars are spent annually in the US alone due to software faults ([11]). There are several sources for these high malfunction costs, such as SW complexity, lack of understanding of customer needs, or inadequate SW development methods. Our approach can shed light on the component of these costs which can be avoided by optimizing resource allocation among development phases.

We address this problem by suggesting a conceptual SW development model that employs the economic description of producer behavior and prescribe how to arrange production to maximize profits. The specific question we address here is how to allocate SW development resources among the different phases in the most profitable way. Since modern SW engineering often uses paradigms of highly iterative nature, (going back-and-forth between phases, such as the agile methodologies), we propose a dynamic version of the conceptual model in which the sequence of the amounts of time spent on each phase is characterized. We demonstrate the validity and advantages of our model by conducting an experimental study of SW development. In that study, based on observing individual SW developers, we control, monitor and record how time is allocated among development phases. We analyze the impact of these allocations on the quality of the resulting SW product.

To our knowledge, the SW development resource allocation problem has never been examined as a profit maximizing problem so fundamental in economic literature. Indeed, Babu and Suresh [2] address the question of resource allocation among different development phases economically. However, their approach considers only the impact of allocated development resources on the finished product, ignoring the iterative nature of the development process itself, and the economic implications of different qualities on the demand for the finished product. From another perspective, the COCOMO-II [4] presents empirical observations on resource allocation among development phases as part of the cost estimation model proposed therein. Our model belongs to the growing literature adopting value-based approaches to software engineering, ([3] and [5], for instance), and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
EDSER ’06, May 27, 2006, Shanghai, China.
Copyright 2006 ACM 1-59593-085-X/06/0005...$5.00.
contributes to it by focusing on the allocation of effort among the development phases.

We consider four main development phases along which SW development work is organized: Requirement, Design, Implementation and Testing. These four phases seem to cover all SW development activities\(^1\). They can be intuitively defined as follows: (1) Requirements – the set of project activities that result in a definition of what the software should do; (2) Design – the set of project activities that result in the definition of how the software should meet its requirements; (3) Implementation – the set of project activities that result in the computer program, which is the physical and executable artifact that is eventually delivered to the customer; (4) Testing – the set of project activities that are aimed at validating implemented artifacts.

Assuming that the developer can clearly distinguish between these phases, and freely choose which phase to work on at any given moment, the problem of profit maximizing by allocating time among development phases is a well-defined optimization problem. The notion of profitability we consider respects clients' needs as they are reflected in the demand relationship facing the developer. The developer can still choose how to organize his/her work, taking into account not just the client's needs, but also how different working time allocations among development phases influence development costs and the quality of the resulting product.

### 2. THE ECONOMIC MODEL

#### 2.1 SW Development as a Production Process

We view the output of a SW development process as a product consisting of a bundle of features that potential users find useful. Each of these features can be provided at different quality levels. The collection of the product features and their quality levels is what we define as the output of the SW development process, and refer to it below as *volume-scaled quality units*. Quality corresponds to the six ISO 9126 [6] quality characteristics and their sub-characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. In fact, our model suggests a bi-level definition manufacturer output. The first level considers the different phase artifacts quality, (internal quality, according to [6]), while the integration of all these qualities into what is ultimately used by the customer constitutes the second level (external quality).

We assume that potential clients of the SW can summarize the features embedded in the product and their quality attributes by an overall product quality index \(Q\), which we view as the output of the SW development process. The number of copies of the SW product sold in the market is given by a *market demand function*, \(D(P,Q)\), where \(P\) is the product price. Higher prices result in lower sales volume for a given quality, and higher quality increases the number of copies sold at any given price, \(D_P(P,Q)<0, D_Q(P,Q)>0\). The precise way by which customers create the quality index \(Q\) involves the relative weights attributed to different features and their qualities. Therefore, uncovering the structure of that index requires statistical analysis of marketing data, (see, for example, [7] for a “cleanroom” approach, or the survey reported in [1]).\(^2\)

The basic assumption of the model is that the quality index \(Q\) associated with any given SW product can be described as a real function of time allocations among the development phases, \(Q = f(R,D,I,T)\), where: \(R, D, I,\) and \(T\) are the time inputs to the following four development phases: Requirements, Design, Implementation and Testing, respectively. For notational convenience we refer to these phases as \(i = 1,2,3,4\), respectively. This production function can also depend on developer-specific characteristics, (both human and organizational aspects), which can be added to the production function when the values of these variables are known, ([4], [12]).

An alternate representation of this same relationship is obtained when we replace the inputs into the four development phases with measures of the quality contribution of each phase to the overall product quality. \(Q = \hat{f}(q_1, q_2, q_3, q_4)\), where \(q_i\) is the quality artifact of phase \(i\) (internal quality). Since the resulting output of each phase depends on the time allocated to that phase, and possibly on the artifact quality of other phases, this last representation of inputs-qualities relationship is equivalent to the former. Denote the time allocated to each phase by \(t_i\) and the output of each phase by \(q_i, i=1,2,3,4\). Output dependence among the four phases can be presented as: \(q_i = h_i(t_i, q_{-i})\), where we assume that both time input into phase \(i\) as well as the output of other phases, \((q_{-i})\), improve the resulting phase \(i\) artifact.

#### 2.2 A Dynamic Economic Model

The function \(\hat{f}\), which maps phase qualities into an index of external quality, can capture either the static aggregative impact of development phases on the final volume-scaled quality units in the product, or the dynamic evolving relationship during the development process. Here, we emphasize that in practice the four phases are intertwined in a rather complex manner, with a typical development often switching back and forth among them.

Suppose there are \(N\) units of time, (or money), which can be used to develop a SW product. Those resource units can be sequentially allocated, one unit after the other, to any of the four phases. The dynamic programming approach allows us to allocate resources among the development phases one at a time, in response to the on-going progress of the project, subject to the remaining available resources. Let \(q \in \mathbb{R}^4\) be a vector representing the current output of the four development phases: think of \(q_i\) as the percentage of the work on development phase \(i\) and its quality that has been accomplished up to the current point in time. Let \(Q = \hat{f}(q)\) be the overall quality index corresponding to the current state of the project development. The evolution of the development state, \(q\), depends on the assignment of each unit of the resource to one of the four phases. We describe this relationship as a “law of motion”, or “SW development process”.

---

\(^1\) While the assumption that it is possible to clearly define the different phases and distinguish between them, as well as the assumption that the mentioned phases cover all development activities is far from trivial, we do not elaborate this point further in this paper. A thorough discussion and empirically based solution of this problem is presented in [12].

\(^2\) Other factors affecting the demand, such as advertisement and promotion, or deployment of the SW product in the client organization, are currently ignored in this model which focuses on the SW development process.
function. This function maps the current development state, \( q \), the assignment of the next resource unit to one of the phases, \( s \in \{1, 2, 3, 4\} \), and possibly other factors which influence the SW development, \( A \), into the new state, \( q' = g(q, s, A) \). Finally, let \( \Pi(Q, n) \) be the maximal profit attainable by selling the SW product with quality \( Q \), having invested \( n \) units of the resource in the project, (with resource unit price \( w \)): \( \Pi(Q, n) = \max\{P(D(P, Q) - w \cdot n)\} \), and let \( V_s(q) \) be the optimal value of the SW project given development state \( q \), and \( n \) remaining units of the development resource.

Given an initial development state, \( q_{0s} \), and \( N \) available units of the resource, the dynamically optimal allocation of the resource to the 4 development phases can be found by solving the following recursive set of problems:

\[
V_0(q) = \Pi(Q(q), N)
\]

\[
V_n(q) = \max\left\{ \Pi(Q(q, N - n)), \left[ \max_{s \in \{1, 2, 3, 4\}} \left\{ V_{n-1}(g(q, s, A)) \right\} \right] \right\},
\]

\( n = 1, 2, \ldots, N \)

with given initial quality \( q_0 \), being typically zero, except when reusing or adapting an existing project.

The first equation reflects the fact that when the development resource is depleted, (0 units of it remain available), and the development state is \( q_0 \) the value of the project is just the highest profit which can be obtained by optimally pricing the product given its quality \( Q(q_0) \). The second equation states that at each stage in the development process, the developer can choose to stop the process and release the project to the market, or continue and optimally assign the next unit of the resource and then decide whether to release the product or continue with the development.

If s/he decides to stop, the profits consist of the revenue to be obtained by selling the project given its current development status net of development costs incurred, \( \Pi(Q(q), N-n) \). If the development continues, the next resource unit is assigned to one of the four phases in order to maximize the next stage's value, taking into account how this assignment affects the subsequent development state. When starting the project, the value of optimally assigning each of the \( N \) units of the resource is \( V_s(q_{0s}) \).

This solution generates an optimal allocation of each resource unit to one of the four phases, \( \{s_N, s_{N-1}, \ldots\} \). Obviously, this process admits highly non-linear development processes, depending on the impact of the assignment of each resource unit on the next development stage.

The dynamic development model presented here can be expanded to include additional factors. In fact, the developer may need to decide on the order in which features are developed, or the possibility of reusing artifacts, in lieu of developing some components, in addition to the decisions modeled here.

3. THE EMPIRICAL STUDY

3.1 Methodology

The objective of the empirical part of this research is to analyze software developing processes in light of the theoretical model, and to obtain some idea the structure the model components. In particular, we study the considerations taken by developers when making resources allocation decisions and try to understand why differ occasionally from the model prescription, and what are the quality implications of these decisions.

The research method we developed to achieve these goals includes both exploratory and measurable aspects. Accordingly, it relied on two different research paradigms: qualitative and quantitative, [8]. Qualitative methods provide insights into the decision making processes of SW developers, while quantitative research methods generate measurable and objective data about variables like the total amount of time invested in each development phase by each developer and the resulting software quality. Importantly, both approaches also serve as validation platforms for future quantitative research based on the theoretical model.

For the purpose of this study we developed two measurement tools: (1) A unique environment, the Econometric, which documented the development process. The Econometric measures the amount of time invested in each development phase throughout the development process, (to be regarded as independent variables). It also supports developer documentation of the considerations justifying the move from one development phase to another. (2) The Quality Checker (QC), is designed to objectively evaluate different quality factors of the system (the dependent variables).

In order to compare different development strategies, we decided not to collect data from the industry, where each product is unique, but to create an artificial environment where several developers work on constructing a solution to the same precisely defined set of customer needs.

The research plan was constructed as a simulation of the development of an information system, executed by individual developers, according to the following principles:

- The software product being developed is a "toy product", that is, a less-complex product simulating some essential functions of the "real" one.
- Software developers are simulated by MIS undergraduate students (in the last year of their studies).
- The external quality of the product is defined in terms of the various quality factors involved and the relative importance of each of them. Software "manufacturers" incentives for optimizing software quality are generated by grading SW application in a manner that is explained upfront, and is included in the students' course grade.

The assumption underlying this experimental design was that we can approximate industrial SW development decision making by providing appropriate incentives to students, despite the vast differences between the classroom and the manufacturing environments. In addition, our simulation approach focused on a single person SW development process, whereas such processes are often undertaken by teams of SW engineers.

3.2 Setting

In the empirical study, we examined four case studies of identical product development by different students. The product was an information system for managing taxi ordering services, to be used by cab companies, called the TOS (Taxi Ordering Server). The system supports taxi ordering services for passengers among given cities under constraints on the number of available cabs.
Our main objective in this empirical study was to explore the developers’ behavior and considerations taken when making resource allocation decisions, in light of the economic model. In addition, we examined the relations between the allocation of development resources as reflected in the development work and the output in terms of volume-scaled quality units.

While developing the SW, the students worked in the Econometric environment, including the reflection tool for documenting their considerations when moving between the development phases. When submitted, each system went through a quality evaluation session which determined the project’s grade. The students were then allowed to re-submit the project in order to improve the initial grade. Students were not aware prior to the first submission of this opportunity to re-submit their projects.

Throughout the development process and following the first submission, personal interviews were conducted with each of the students. The interviews included thorough investigations aimed at clarifying the students’ reflection documents. This allowed us to gain better understanding of the considerations governing decisions to shift among development phases.

### 3.3 TOS External Quality - Definition and Evaluation

In order to make their own decisions on how to develop the software, the developers were given weights on the various ISO 9126-1 quality factors of the application to be developed. According to our theoretical approach, such weights are needed so that the developer can decide, given the underlying SW production function, how to allocate time in the most “profitable” way. Naturally, the weights were given to a rather broad collection of ISO quality factors, whereas the developer still had to use his/her own judgment in order to decide how to weigh sub-factors for which no specific weights had been given. In what follows, we describe the particular concretization of the ISO quality factors in the context of the required TOS application and the weights assigned to these factors in the developer’s grade.

The Quality Measure of the TOS application was defined as an aggregate measure of two key quality factors: Functionality and Maintainability (henceforth F and M, respectively), whereas other factors included in the ISO definition received zero weight. The resulting formula of the TOS quality was as follows:

\[ Q = F^{0.75} \cdot M^{0.25} \]

The Functionality quality factor was divided into two sub-factors: Suitability and Compliance, (henceforth S and C, respectively), while the other ISO defined sub-factors received zero weight each. The Suitability sub-factor was assigned a weight of 96% and the Compliance sub-factor was assigned a weight of 4%. The resulting formula for the Functionality Factor was:

\[ Q = F^{0.96} \cdot S^{0.04} \]

The Company’s profit is calculated on the basis of the payment received from passengers and the operating costs, per unit of distance driven. In light of the general ISO definition of Suitability, we define this quality sub-factor of the TOS application as: \[ S = e^{-\frac{1000}{\text{profit}}} \]. This measure exhibits decreasing marginal contribution of profits to suitability.

Based on the ISO general definition of Compliance, we divided the Compliance quality sub-factor in the TOS application into two sub-factors: C1-- the degree to which the application adheres to related standards in case of correct inputs, and a similar measure of the application’s response to incorrect inputs, C2. We assigned equal weights to each of these components, so that the resulting formula was: \[ C = C_1^{0.5} \cdot C_2^{0.5} \]. Accordingly, the Functionality quality factor is:

\[ F = C^{0.96} \cdot S^{0.04} = \left( C_1^{0.48} \cdot C_2^{0.48} \right)^{0.96 \cdot 0.04} \]

The Maintainability factor was defined in terms of the sub-factor of Changeability only. In the context of the TOS application, Changeability could be evaluated in terms of how long it takes a reasonable SW engineer to make a particular change in the original application. The overall quality measure of the application given to students was:

\[ Q = F^{0.75} \cdot M^{0.25} = \left( C_1^{0.96} \cdot S^{0.04} \right)^{0.75} \cdot M^{0.25} \]

\[ = \left( C_1^{0.36} \cdot C_2^{0.36} \cdot S^{0.03} \right) M^{0.25} \]

In order to assess the Functionality external quality score, an automatic Quality Checker (QC) was developed. The QC calculated the score of each of the Functionality sub-factors, and produced a scalar score according to the formula presented above. The C1 and C2 compliance sub-factors were calculated by the QC on the basis of exhaustive and representative Test-Suits, one for correct inputs, and one for incorrect inputs, respectively, where the application either passed or failed each test. The Suitability score was derived from the C1 Test-Suit, (the correct inputs), according the resulting profits of the taxi company. Profits were calculated by adding all payments received for performed rides, minus the cost of these rides which depend on their length.

The QC yielded a detailed report including a list of all test cases run on the TOS, a "Pass" or "Fail" message for each test case, an indication for the source of the TOS failure, and finally a summary report for Compliance, Suitability and Functionality.

### 3.4 Results

The TOS application provided a platform for validating the use of our model for analyzing the allocation of resources among SW development phases. We used this experimental setting to achieve three goals: (a) verify the ability to use the model for interpreting actual developers’ conduct during the SW development process; (b) obtain some initial information about the nature of the functional forms of general relationships appearing in the model; (c) draw inferences on potential reasons for observed deviation of SW developers conduct from the model’s prescription. This experimental approach was used first to characterize and measure the variables appearing in the model, as reported in [12]. Here we present what has been achieved on each of these goals.

(a) **Model Appropriateness.** Developers’ behavior was generally consistent with the basic assumption underlying the model that development efforts are allocated primarily to achieve the highest

---

3 We use the word “weight” loosely, and mean the elasticity of the score with respect to each quality component.

4 In this study we did not measure the external maintainability factor. However, we evaluated the effects of design (q3) and implementation (q4) on maintainability. For details see [12].
possible quality measures. All students expressed in the reflection document time and again their will to increase (certain aspects of) quality when describing their decisions to move between development phases. This finding is consistent with [1], reporting that the external quality is the most important factor in the evaluation of senior IS managers.

Another premise of our model that was confirmed is that each activity during the development process can be classified into one of the four phases, according to carefully constructed definitions as in [12]. Moreover, we have demonstrated that these activities can be measured by the Econometric application. Obviously, our experiment is consistent with each of the four phases being essential, in the sense that no time spent on any of the phases resulted in zero quality.

On the other hand, we noted that the model should be refined in several ways. For instance, as suggested in Section 2, we can add the following planning decisions to the model: how much time to allocate to different features in the required application, and the order in which they are being developed. We have observed that the latter decisions are determined jointly with the allocation of time among development phases.

(b) Inferences about the model functional forms. We established that external and internal qualities are generally increasing in the time spent on each of the four development phases. The economic model rests on knowing how different combinations of inputs, in the form of hours worked on the different development phases, affect the resulting software. In particular, it is assumed that the SW developer can increase one of the inputs while decreasing the other at some rate without changing the quality of the produced SW. This possible substitution among factors of production allows the developer to choose the least costly combination of inputs to produce the desired SW product.

In our study we observed two kinds of such substitutions. One was the fact that developers chose vastly different input combinations, and yet managed to produce SW products which were more or less of comparable quality. This observation supports the general idea that one can produce a particular SW product by different combinations of inputs, so that there is a non-trivial decision to be obtained here, namely, which input combination yields the highest "return" to the developer.

Our second observation was that personal attributes or traits of the developer, referred to as developer-specific human aspects, can influence the degree of substitutability among the development phases. For instance, we observed that developers with prior programming experience could produce the same quality SW with less time spent on (unit) testing. Likewise, prior experience with certain JAVA tools enabled some developers to exploit these advanced tools during implementation, thus allowing them to get higher quality output from this phase.

Another observation illustrates what can be learned about the functional form relating development phases and quality. Our observations on these aspects are consistent with decreasing marginal product of effort exerted on each of the phases. Besides being an important and very common economic characteristic of production functions in many areas, this property usually assures us that optimal solutions will not be corner solutions, which is a kind of "sanity check" on the model. For instance, expressions such as the following: "I moved to Implementation after achieving a reasonable level of the UML documents relative to the time I had left" appeared many times in the reflection documents. In further investigation we found that students' perception of reasonable level of UML documents included mainly a description of basic functions and their sequence; defining which class performs which function; and "organizing" the functionality according to a logical, sense-making, order. Being aware of the time constraints, it seems that what the students had determined as "reasonable level of design" is the Design construction up to the point where the "marginal product" generated by an additional time unit of Design is lower than the "marginal product" of Implementation.

(c) Development misallocations and their sources. We have found that developers often deviated from the model's prescriptions regarding their resource allocation decision making. As a result, students generally received zero scores on functionality (in early project submissions). We suggest the following as possible reasons for this phenomenon.

Students failed to allocate sufficient time to the Requirements and Testing phases. For example, mistakes in the format of the response message sent by the TOS to an incoming request, a common reason for getting a zero score by the Quality Checker, could have been avoided by reading the requirements more carefully. Likewise, students' projects failed to detect incorrect inputs though they had been explicitly instructed to detect them and respond with an appropriate error messages. A main reason for this failure was that developers failed to include appropriate Testing of incoming orders.

We found that some developers perceive Functionality improvement as stemming almost exclusively from time spent on Implementation, failing to appreciate the contribution of other development phases to this SW quality factor. Thus, allocating time to requirements and design results, in their opinion, only in the increase of the Maintainability factor. Moreover, we found that developers invest time in infrastructural activities mainly when forced to do so. In an interview, a student said: "I considered doing a Design document. Since you didn't say it's a must, and I thought it will take me too much time, I didn't produce it in the end".

Developers tended to over-invest in the Suitability aspect of Functionality at the expense of Compliance, which was defined as a much more important component of Functionality. Suitability was measured by the profits generated by the scheduler. Not only was the weight of Compliance 24 times higher than of Suitability, according to the quality definition given to the developers, the marginal impact of profits on suitability was decreasing.

Following the development process, we performed an investigation into the students' motives while attempting to examine what caused these poor results. Our examination revealed that some students simply did not understand the quality definitions, while others did, but chose to operate differently. For example, in a follow-up interview, a student said: "I was aware of the weights given to different quality factors, but...

---

5 Internal solutions are obtained when the production function is concave, and decreasing marginal product is a necessary condition for concavity.
This study.

Developers tended not to use helpful SW tools with which they were not familiar. It appears that developers ascribe more importance to the evident time costs than to the unknown efficiency gains. Consequently they may allocate too much time to develop new “tools” instead of using available ones, and this often distorts their time allocation between development phases.

Human aspects, such as attention span, concentration ability, memory capacity and diligence, have significant influence on the chosen allocation of time between phases. For example, in many cases students chose their next activity according to how concentrated/tired they felt at that time, or due to a concern they would forget to conduct a certain activity at a later stage.

4. CONCLUSIONS AND FUTURE WORK

This paper presents an economic model for optimal allocation of resources among development phases. An empirical study was conducted in order assess the model usefulness for evaluating and improving SW development processes.

We demonstrate the usefulness of the model as an evaluation tool by carefully constructing an experimental setting that allows us to simulate developer's incentives and goals, while monitoring the work effort on each development phase and the resulting SW quality. We find significant and systematic deviations between the model prescribed allocation and actual developers conduct, and use the model to identify some of the sources of these biases.

We regard this exploratory study as a first step towards a more comprehensive quantitative empirical study, where the nature of the functional forms assumed in our current model can be estimated, (demand for SW quality, SW production function, etc.). However, even our current findings provide support to the idea that it is possible, and useful, to regard SW developers decision making as a rational process of resource allocation aimed to maximize a well defined objective which depends on the costs and resulting quality of the developed application. In particular, we could confirm that such decisions reflect the existence of some substitution among development phases and declining marginal products of effort allocated to different phases, as well as developers' awareness of how the allocation of their efforts affects their own perceived reward.

Future work should expand the empirical basis of this approach, in order to better estimate the model relationships, and developers responses to alternative incentive structures. This will enable us to use the model for prescribing best-practice allocation of developmental resources, depending on the nature of the SW application being developed and the characteristics of the organization and SW engineers engaged in that process. The model can be used to identify where developers misallocation of resources is most critical, and to suggest how to rectify these deviations by better alignment of individual developers' incentives and organizational goals. This extension of the current study requires, among other things, better ability to control for individual developers characteristics than we could exercise in this study.

Another extension of our project involves the industrialization of the methodology and tools for measuring resource allocation, (Econometric), and the resulting internal and external qualities, (Quality Checker). These tools can aid SW manufacturers in finding the most productive way of organizing their work efforts along the SW development phases.

5. ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support and encouragement from the Caesarea Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science, (C.R.I.), at the University of Haifa. We thank Eitan Farchi (IBM, Haifa) for professional guidance, and Regev Porat, Michal Tal, Miri Naim, Nir Gallner, Eran Shemila and Yonatan Netter for their important help.

6. REFERENCES


