2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Need to Redefine "Value" and Case for a New "Software Valuation" Technique: An Analytical Study

Mohammad Ramzan, Sajid Anwar, Arshad Ali Shahid[+]
National University of Computer and Emerging Sciences (FAST),
A.K. Brohi Road, H11/4 Islamabad, Pakistan,

Abstract. In this era of progress and innovation, every product and service has certain value attached to it. Similar is the case with software products and services. These also assign some value to the stakeholders for whom this product or service has some meaning. Software Engineering (SE) deals with the development of quality software product that fulfills stakeholder's requirements. There are many valuation techniques which are being used to establish value of software products and services. However, irony of the situation is that current practices in software engineering are value neutral in essence. Value Based Software Engineering (VBSE) takes into consideration the value assigned to the software product and to the stakeholders who have interest in that software. However, VBSE as a theory has not redefined the term "Value" for the discipline of software engineering. In this paper, we present an analysis of current definitions of value as well as valuation techniques and subsequently, we have presented a case for the need to redefine the term value and modification of current valuation techniques for software engineering.

Keywords: value, software engineering, software, Process, value based software engineering, requirement engineering

1. Introduction

Software Engineering as a stream of knowledge deals with development of software product that meets certain business and technical quality constraints imposed on it while at the same time meets the requirements of stakeholders. This means that software engineering ensures that certain amount of value is attached to the software (being developed).

It is an interesting fact however, that classical software engineering is "Value Neutral" in essence [1]. This means that software engineering does not consider the value that the product is going to deliver to stakeholders nor does it assign value to the stakeholders who have an interest in development of the software product.

In the past two decades or so, many new theories and changes in the classical SE knowledge have emerged. Emergence of these new concepts has essentially altered the way we perceive SE and software development process. Value Based Software Engineering is one such theory which has been around for more than a decade now. Many known researchers have worked on design and formulation of VBSE. VBSE essentially incorporates the element of value in otherwise value neutral SE practices.

However, it is again interesting to note that VBSE incorporates the concept of value in SE practices without actually first describing what should we mean by value in the domain of software development. Having absolutely no doubt in the fact that VBSE is a major and exciting emerging theory and it has immense potential in redefining the SE principles, we believe that there is a need to define value in the context of software engineering. Only then can we assess the true worth of a software product and stakeholders associated with it.

It is also equally important to have a look at existing valuation techniques. Many valuation techniques have been used both in traditional business as well as software products and services. Some major names among these are Net Present Value, Internal Rate of Return, Sensitivity analysis etc. COCOMO and COCOMO 2 have been designed with the aim of better estimation of software. There is an urgent need to investigate the application of all of these approaches to determine their suitability and present a case for further improvements if necessary.

We have also to be mindful of the fact that many of the concerns that we express in our work are present in the mind of software developers, project managers and other stakeholders when they valuate the software product. What we have tried to emphasize upon is the need for a formal new definition of value and anew valuation technique so that all of these concerns can be accommodated formally.

[+] *E-mail address*: muhammad.ramzan@nu.edu.pk , sajid.anwar@nu.edu.pk , arshad.a.shahid@nu.edu.pk

In this paper, we have put forward a case to describe the reason and significance of redefining the concept of value for software engineering. At the same time an analysis of current valuation techniques with the aim of insight and further improvement has been made. The paper is structured as follows; after giving a brief introduction in section 1, a brief literature review is given where we describe certain prevalent definitions of value and their significance along with a study of valuation techniques. In section 3, we have discussed COCOMO and COCOMO II from the perspective of software valuation. Section 4 presents a critique of these definitions and techniques while analyzing their strengths and shortcomings. In section 5, we have described certain preconditions that any new definition of value should cater for in order to become a suitable definition of value for SE. At the same time, we have shown how these modifications can be useful towards better establishing the value of a product based on the stakeholder requirements. Conclusion and future work has been presented in section 6.

2. Literature Review

It is important to understand how value has been interpreted in the literature. This section describes in detail certain definitions of value and valuation techniques being applied in modern day software development.

2.1. What is Value?

The ultimate aim of any industrial knowledge (and same is true for software engineering) is to create certain products or services which add value to the existing worth of stakeholders, for whom that service, product or process is designed. If the element of value is excluded, the creation of these products, services or processes is rendered meaningless.

When talking of SE practices, the creators of Value Based Software Engineering theory conclude that "Our aim is to bring such value considerations to the foreground so that software engineering decisions at all levels can be optimized to meet explicit objectives of the involved stakeholders" [1].

What is important in this definition is the identification of the "Value" considerations for the purpose of maximization of objectives of all stakeholders. So it seems appropriate that we take a look at what literature offers us in terms of definition of value.

- According to Merriam-Webster online dictionary, value can be; a fair return or equivalent in goods, services, or money for something exchanged
- the monetary worth of something
- relative worth, utility, or importance
- a numerical quantity that is assigned or is determined by calculation or measurement [2]

Dictionary of Canadian Economics defines value as: *"The quantity of one product or service that will be given or accepted in exchange for another"* [3]. This definition caters for the most classic form of value assigned to any product or service. Value therefore becomes a matter of mere economic significance according to this definition. According to this definition, value relates to the "quantity" that any product or service can be exchanged for. This definition though explains value quite well for almost all classical products or services, it is unable to define the value of software products or services that well.

Another definition of value comes from Oxford Companion to Law and it states that "*...value may consist of spiritual or aesthetic qualities or in utility in use, or in the amount of money or other goods which could be obtained in exchange for the thing in question...*"[4]. This definition though quite comprehensive in its own right still leaves many questions unanswered. Lastly, Dictionary of Sociology defines value as a *"...generalized principle of behavior to which the members of a group feel a strong commitment and which provides a standard for judging specific acts and goals"* [5]. This definition generalizes the concept of value to such an extent that it is difficult to apply it particularly on software engineering.

What all of these definitions fail to cater for can be summed up as:

- These definitions rely too much on money as the relative unit to describe value
- The value in all of these definitions is established based on market forces.
- There is no room for accommodating the value of process and its maturity that goes into development of product or services.
- Human resource and its quality that goes into development of these products or services has no significant role in establishing value
- The value of product is established based on its business value in classical sense which is not applicable to the modern products or services like software.

2.2. "Value " in Software Engineering

The significance of value to software engineering is quite manifest. As Stefan Biffl et al have aptly highlighted in their work that the ultimate aim of software engineering is to add value to the existing state of affairs through creation

of products, services and processes [1]. The authors have also mentioned the overall negative impact that this whole process can cause if the value consideration remains implicit. We shall briefly describe in chronological order, the application and evolution of concept of value in software engineering.

Traditionally value has been used to describe cost models in software engineering. The first major work to address the concept of value beyond cost models was Boehm's software engineering economics [6]. Consequently, in 1986, spiral model was introduced by Boehm after establishing relationship between value and software process. McTaggart's work [7], titled as "The Value Imperative" resulted in a new way of thinking which was subsequently named as value-based management movement. A result of this movement was an IEEE software essay titled "When the Pursuit of Quality Destroys Value" by Favaro [8] in 1996. In this essay, Favaro argued that pursuit of quality should not be the sole aim as in many cases, this pursuit can destroy the value of the product. Later, in another article, the adjective "value-based" was used by Favaro et al. in the software development context addressing the economics of software reuse. [9].

WinWin model was another model of software engineering by Boehm et al. This model was proposed in 1998 and it basically dealt with the concept of requirement negotiations [10].The formal agenda of Value Based Software Engineering was proposed by Boehm et al in 2003. This agenda captured the expanding scope in the domain of value based management approaches as well as agile development methods [11].

2.3. Economic Valuation Techniques

Traditionally, software economics has relied heavily of estimating the cost. For example, COCOMO II calculates the overall effort of a software project through using this equation.

Effort = (Personnel)(Environment)(Quality)(SizeProcess) [12]

According to Patrick McKenna, this equation captures certain key factors and delivers an estimate of the effort required for software project [13]. These equations have been mainstay of many researchers like those working with IBM's Rational. However, these researches and applications have only covered the cost side of this whole software economics [13]. The value side of this whole process has been covered with the help of VBSE by Barry and several researchers who have worked on it.

Many valuation techniques can be found in literature. Though originating from economics, these are equally applicable in the field of software engineering. Many of these techniques have been widely used for valuation purpose of software projects quite effectively. One major valuation technique is return on investment (ROI). Value can be directly calculated using ROI technique which determines time to payback [13].

Net Present Value (NPV) is another major valuation technique which measures the profitability and hence value of the product according to certain statistical techniques. Internal Rate of Return (IRR) is used in capital budgeting and has been a favorite technique for valuating software projects for quite some time now. Sensitivity analysis is actually performed to show what kind of uncertainties can have serious affect on the value of software product. Monte Carlo simulation method for valuation of software product uses random numbers and probabilistic approach to generate multiple possible scenarios and establishing value of the product according to probabilities of these scenarios. Final result of this simulation is the product's value which can be quite impressive provided sufficient data and a realistic model is used to run the simulation [13].

3. COCOMO and COCOM II, Means for Software Valuation

COnstructive COst Model (COCOMO and COCOMO II) have been the leading estimation techniques for software products and services. In this capacity, these can be used for valuation of the software as well. In 1981, COCOMO was first published by Barry W. Boehm in his Book Software engineering economics as a model for estimating effort, cost, and schedule for software projects. The work was based on the study of 63 projects at TRW Aerospace. At that time, Barry Boehm was Director of Software Research and Technology at the organization. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. COCOMO model is often referred as COCOMO 81.

In 1997 COCOMO II was developed and it appeared for the first time in published format in 2001 in the book Software Cost Estimation with COCOMO II. As a successor to COCOMO 81, COCOMO II is better suited for estimating modern software development projects. More importantly, it caters to the software which are developed with process models other than the waterfall model and new programming languages. It established its utility in the era of desktop development, code reusability and the use of off-the-shelf software components.

Here, we are more interested in understanding how COCOMO II works and how it can be used for valuation purposes. COCOMO II allows one to estimate the cost, effort, and schedule when planning a new software development activity. There are three major components to this model where each model caters to different level of abstraction and detail. These sub models are called the Applications Composition, Early Design, and Post-architecture models. COCOMO II was designed to meet following objectives:

- Investment and financial decision making based on software development effort

- Project budgeting and schedule
- Tradeoff negotiation
- Risk management
- Level of reusability and legacy software inventory decision
- Setting mixed investment strategies
- Process improvement strategy

COCOMO II offers many advantages towards estimation when it comes to effort or cost. The main advantage is the status of COCOMO II as an industry standard. Its easy availability and highly understood process for estimation make it ideal for the specific purposes. The presence of tool support is another highlight of this model. With extensive research going on in this specific model all over the world, it is well positioned to be the standard estimation approach for a long time to come. At the same time, one major drawback of COCOMO model is its non-suitable nature for small scale development effort.

However, when we talk of valuation process, we are not simply discussing the cost or effort estimation. Additionally this estimation is essentially aimed towards development of the product. Our actual aim in valuation should be at reaching to a model where these estimates can be used for establishing a realistic value for the software. Also some new models which discuss the valuation of the product not just from its development perspective but also from other perspectives such as utility, applicability, requirements fulfillment etc. Without the presence of such mechanisms, we shall always be tempted to value the product based upon the total cost which has gone into its development (whether for effort or time). Many other factors which should play their role into valuation will still remain missing.

So summing all of this discussion regarding COCOMO II, we believe COCOMO II in its present form can be a useful mechanism for valuation provided certain other parameter (essentially non-cost factors) are incorporated into it.

4. Critique of Contemporary Definitions and Valuation Techniques

In this critique, we present an in depth analysis of current definitions as well as valuation techniques so that in the later section, we can present our observations in the form of guidelines which should be accommodated in any new scheme of things.

4.1. Analysis of Current definitions of Value

As can be seen from table 1, there are many strengths of current definitions of value. However, there are many drawbacks associated with it as well. In this section, we will highlight some of those drawbacks which have a significant bearing on software engineering in particular.

Definition is too Abstract: As we have seen in all definitions of value described in the sections, the description is very abstract in its nature. The effect of this abstraction is a possible absence of many other diverse properties of value. For example, when we describe value of any entity in modern world, we usually want to describe its business value. However, most software products don't have a business value that strictly meets the above description of business value. For software, the real business value is not a direct monetary return that it can fetch.

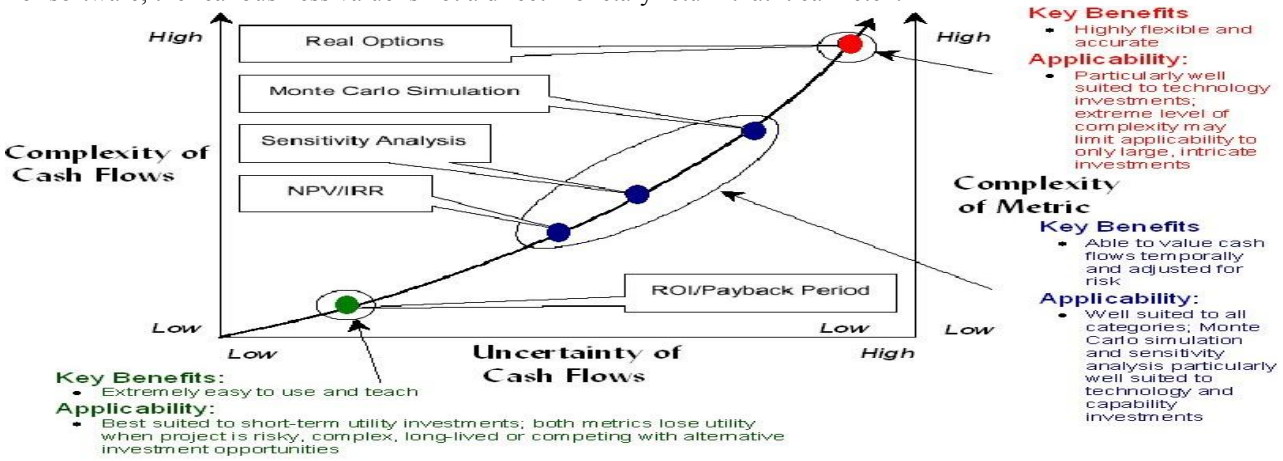

Fig.1: An analysis of Value Measurement Techniques [18]

- **Definition of Value is relative to Money:** The way value is currently defined, it strongly resembles many other definitions of physical sciences like mass, energy, pressure etc. The common aspect of all of these definitions is

that all of these (including value) are relative. All these definitions require some other definition of unit to describe them. For example, definition of mass can't be fully understood unless the definition of its units i.e. kilograms or pounds is used with it. Similarly, current definition of value cannot be fully comprehended until the definition of its associated unless is not understood. In current scenario, the associated unit for value is often money. We usually assign value to any product or service in terms of how much money, this particular product or service can fetch. However, when we talk of s software product or service. It need not be the one to have a lot of money associated with it to make it valuable. In fact, it has been the case that many most valuable software products and services are absolutely devoid of any money associated with them. So attaching a unit like money to describe the value of software is not practical or even justified to describe it value.

- **Market Forces are not determined according to current scenario:** Current definitions of value imply that market forces always have a significant role in determining the value of any product or service. However, there is increasingly large evidence that the effort that goes in the development of product or service has its own value. We can cite the example of the development process that is used for development of software. Many a times, the value of ultimate product heavily relies on the maturity of the process that goes into development of the product. CMM and CMMI certifications actually evaluate the quality and ultimate value of software product based on maturity of the process. This aspect has also not fully catered for in the current definitions of value.

- **Human and Time Resource are not properly incorporated:** Goods have been evaluated based on the human resources involved. However, most of the times, this has played quite negligible role in the overall valuation process. Similarly, the time that a product or service takes in development has also not played any significant role in this process. We know that many of modern day products (including software) have human resource as their major raw material. Current definitions have failed to put so much significance on this specific aspect. Also we need to establish how time plays a role in establishing value for a product or service

- **Concept of Value Prioritization is lacking:** Current definitions of value fail to grasp with the fact that value of the same product can be different for different stakeholders. Thus it is not possible to assign one single value to a product. This is particularly true in the case of software engineering where various stakeholders perceive same product or service with varying degree of significance.

	Strengths	Limitations
1	Establishes the concept of Return	Fails to determine value of many products e.g. software
2	Established and applied practices with practical applications	Lack of all stakeholder's participation
3	Notion of utility and desirability	Too abstract in nature
4	Covers many aspects of daily life	No concept of value prioritization
5		Customer satisfaction not incorporated
6		Value of various qualities achieved through the process is not addressed

Table 1. Strengths and Limitations of current definitions of "Value"

4.2 Analysis of Contemporary Valuation Techniques

As, we have shown in section 2, there are many valuation techniques out of which we have selected five techniques which have been used for valuation of software projects and products. These include NPV, IRR, ROI, Sensitivity Analysis and Monte Carlo Simulation. An apt comparison of these and few more valuation techniques was presented by Nancy Burchfield in a diagram in her work [14] where she has highlighted the applicability and advantages of these techniques. This comparison is shown in figure 1.

In this section we analyze each of these techniques and present an analysis for applicability of these approaches for a better valuation process.

ROI/Payback Period: According to Lutz Prechelt [15], **"**In the dynamic view, ROI describes the periodically recurring profits (returns) from fixed financial capital (investment). In the static view, ROI describes the one-time income or saving (return) realized as a consequence of a one-time expenditure (investment)"

Return over investment is a process of directly calculating the value using following equation;

$$ROI = \frac{\text{Cost of Project}}{\text{Annual Cash Inflows}}$$

According to this technique, the best valued product is one with the lowest payback time. In other words, higher, the cash inflows, smaller the payback time and thus better the value of product. When we look more closely into this technique, we see it plagued with several problems. Some of these can be:

- From purely economic perspective, it ignores the future trends of money depreciation or otherwise. Thus the value calculated is unrealistic in the first place.
- The technique does not accommodate the concept of risks for a software product. Thus the results achieved are heavily discounted from this perspective.

As we have seen in the previous discussion regarding value, this technique too establishes value of a software product purely on the basis of monetary returns. No other important considerations are entertained.

Net Present Value (NPV): In this technique, present value of cash inflows is subtracted by present value of cash outflows. Main improvement of this technique over ROI is that it takes the future movement of currency into consideration. However, just like ROI, the accuracy of this technique is also dependent on reliable availability of cash inflows in future. According to investopedia [16], NPV can be calculated as:

$$NPV = \sum_{t=1}^{T} \frac{C_t}{(1+r)^t} - C_o$$

Where C = cash inflows or outflows and r = discount rate over a period of time t.

Net Present Value offers a direct advantage over ROI in the sense that inflatory factor is taken into consideration. One major problem faced by net present value as discussed by Joan Pasqual et al. [17] is the non "monotonic" nature of NPV function in most of the cases. This makes it difficult to interpret the results properly. Similarly, at the same time, the glaring factor remains that just like ROI; NPV also considers value to be only monetary in nature.

Internal Rate of Return (IRR): IRR is a mechanism for analysis of a major investment with relation to the time value of money. It basically calculates the interest rate equivalent of the dollar amount of return of the investment. If the knowledge about interest rate is known, we can compare it to IRR rates on other investments.

When talking specifically about the software products or services, it is quite difficult to calculate the amount of return over certain period of time. The requirements for software products change much more frequently than many other products. Consequently, the level of anticipation for return is quite low. In the absence of certain careful mechanism where experts try to figure out the future trends of that particular domain, relying merely on IRR can be fatally misleading.

Sensitivity Analysis: Whenever we perform analysis of the system, we are interested in knowing how "sensitive" the proposed system is to the change in value of various parameters being used for analysis. This is a very fascinating technique for establishing real value of a product such as software which shows very dynamic behavior in its lifespan.

According to Lucia Breierova and Mark Choudhari, sensitivity analysis is quite useful when building confidence in model by studying and applying certain variations or "uncertainties" present in various parameters of the model [18]. However, sensitivity analysis poses its own limitations. One major limitation is that it is almost impossible to check all parameters for all possible changes so that a really impressive analysis could be performed.

Monte Carlo Simulation: In monte carlo simulation, there is a heavy reliance on the usage of probabilistic approaches and random numbers to find solutions for our problems.

The PMI (Project Management Institute) defines Monte Carlo Simulation as: "A technique that performs a project simulation many times to calculate a distribution of likely results."

In the words of McKenna "The Monte Carlo Simulation typically uses random number generators to generate multiple scenarios of a model by repeatedly sampling values from the probability distributions for the various input variables" [13]. This approach also shows the possible variance in value of the product due to risks involved which gives more credence to the technique. However, its purely economic sense makes it difficult to establish an all round value of the product.

5. Modification guidelines for new definition of Value

Before proposing the modifications in the existing definitions of value to make them more suitable for software engineering and its emerging concepts, it is essential to point out those specific aspects which need to be incorporated in the classical definitions. These specific aspects can work as guidelines towards new definition. In our opinion, following aspects need to be considered in this regard.

- Abstraction in the definition needs to be reduced so that the definition specifically caters for those aspects which are relevant.
- The relative nature of value with currency or money (as in classical considerations) needs to be replaced with utility and throughput.
- Market forces that affect the value of any service or product need to be redefined.
- The value of a product or service should also cater for the maturity of effort that goes into its development.
- Quality of human resource that goes into development of products or services needs to be used to establish its value.
- Establishing a mechanism to assign different value to similar for different stakeholders.

1	Abstraction must be reduced
2	Relativity with "money" should be replaced with either "utility" or "throughput"
3	Incorporation of worth of "maturity" of the process
4	Incorporation of worth of "quality" of human resource
5	Value prioritization mechanism
6	User satisfaction

Table 2. Summary guidelines for new definition of "Value"

- Establishing the fact that degree of accommodation of stakeholder's requirements remains the ultimate mechanism for establishing the value of a product.
- New definition of value should be a mechanism for determining the stakeholder's satisfaction with respect to the product or service delivered and process that goes into development of that product or service.
- We believe that it is essential for meaningful evolution of new software engineering paradigms like VBSE to redefine the concept of value. This process should accommodate all the concerns that have been mentioned above.

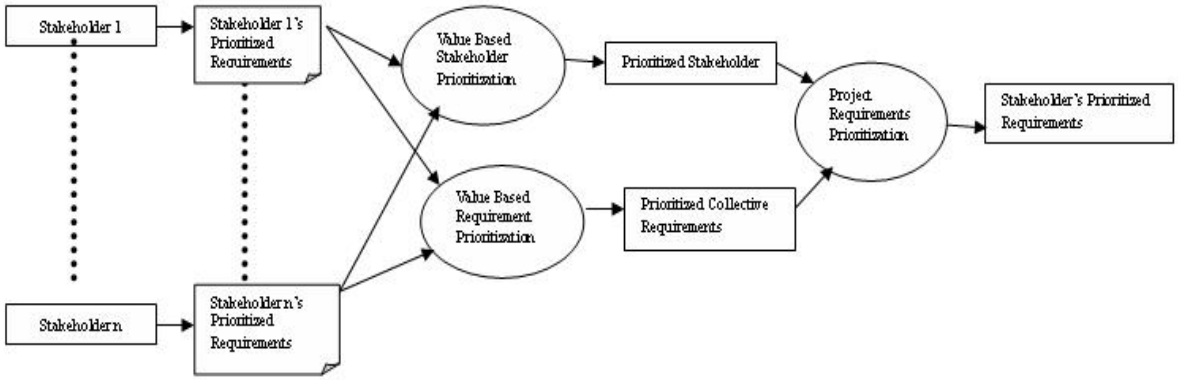

Fig. 2: Value of the product based of requirement Prioritization

At the same time, we believe that a new concept of stakeholder's prioritized requirements should be introduced to establish a more realistic value of the software product or service. We have tried to demonstrate this in figure 2. This concept is also difficult to implement with current definitions of value.

Summing up all this together, if we examine the pure definition of value which states "The value of goods is the product of their quantity multiplied by their price." we can easily come to the conclusion that this definition doesn't cater for the modern day software development and engineering.

6. Conclusion and Future Work

Emergence of new paradigms like value based software engineering is an exciting evolution in the knowledge of software engineering. However this emergence is not without its challenges. One particular challenge in this regard has been the definition of "value" in the context of value based software engineering. Current definitions of value are

unable to precisely describe the worth of software products and services. We have presented the case for the need of emergence of some new and meaningful definition of value. We have also described the guidelines which should be kept in mind while proposing any modification in the definition of value to better represent the modern day products and services.

Work is already in progress to establish a meaningful definition of value. We are also in the process of developing anew software valuation schema which accommodates all of these concerns and give a better realistic value of the software in the early requirement engineering stage of the software development. A critical evaluation and appraisal is also needed to further establish the worth of this new definition.

7. Acknowledgments

The authors, Mr. Muhammad Ramzan and Mr. Sajid Anwar would like to acknowledge the Higher Education Commission (HEC), Govt. of Pakistan and NU-FAST for providing funding and required resources to complete this work. It would have been impossible to complete this effort without their continuous support.

8. References

[1] Stefan Biffl et al.: Value Based Software Engineering (Springer, 2006)
[2] "value." Merriam-Webster Online Dictionary. 2008. Merriam-Webster Online. 23 October 2008, http://www.merriam-webster.com/dictionary/value
[3] “Value”.Dictionary of Canadian Economics. 2006
[4] “Value”. New Oxford Companion to Law, ed. Walker, David M. (Oxford: Clarendon Press. 1980)
[5] “Value”. “Dictionary of Sociology”, 2005
[6] Boehm, B. W.: Software Engineering Economics (Prentice-Hall, 1981)
[7] McTaggart, J.: The Value Imperative (The Free Press, 1994)
[8] Favaro, J.: When the Pursuit of Quality Destroys Value. IEEE Software (May 1996)
[9] Favaro, J., Favaro, K. R., Favaro, P. F.: Value-based Reuse Investment, Annals of Software Engineering, 5 (1998)
[10] Boehm, B. W., Egyed, A., Kwan, J., Port, D., Shaw, A., Madachy, R.: Using the WinWin Spiral Model: A Case Study. IEEE Computer, (July 1998)
[11] Boehm, B. W.: Value-Based Software Engineering. Software Engineering Notes, 28(2):2003
[12] Boehm, B and Sullivan, K. J. , "Software Economics: A Roadmap," in The Future of Software Engineering, 22nd International Conference on Software Engineering, June 2000.
[13] Patrick McKenna, “Assessing the economic value of software projects”, IBM Rational Edge, November 2005 http://www.ibm.com/developerworks/rational/library/nov05/mckenna/
[14] Nancy Burchfield,"IT Options Analysis" , IBM Global Services, ISMC Orlando, October 2002, ,.
[15] Lutz Prechelt, “Optimizing Return-On-Investment (ROI) for Empirical Software Engineering Studies”, in Empirical Software Engineering Issues. Critical Assessment and Future Directions, PP 54-57, Springer Berlin, June 2007
[16] from Investopedia, at http://www.investopedia.com/terms/n/npv.asp
[17] Joan Pasqual Rocabert & José Antonio Tarrío & María José Pérez, "Anomalies in net present value calculations. A solution", Hacienda Pública Española, IEF, vol. 173(2), pages 47-60, June 2005.
[18] Lucia Breierova & Mark Choudhari, “An Introduction to Sensitivity Analysis (D-4526)”, System Dynamics in Education Project, System Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, September 6, 1996.