Survey of ROI Methodologies for Software Process Improvement

Manas Kumar Yogi[1], G.Vijay Kumar [2]

Abstract— **This paper presents the various methodologies adopted to find the Return of Investment (ROI)for any software process improvement. All these methodologies take into account the key elements the ratio of cost by benefit which acts as a robust indicator for supporting critical budgeting decisions of a software organization. We discuss regarding various models for determining the ROI along with the inherent difficulties faced by the team which is determining the ROI.**

Index Terms— **ROI, SPI, Estimation, Cost, Benefit**

I. INTRODUCTION

The return on investment (ROI) of software process improvement (SPI) is an indispensable tool for finding out how effective an organization is at computer programming. The ROI of SPI also extends to the fields of software engineering, information technology, and commercial market software. The ROI of SPI is useful for providers of high-technology products and services. ROI is not limited to a single market sector, but applies equally to the commercial, government, and military sectors. ROI of SPI is useful for ensuring the peak operating efficiency of large, nonprofit organizations.

The ROI of SPI involves determining how much money a new software tool, process, or methodology yields. ROI reveals how much money a software engineering standard yields on the bottom-line corporate balance sheet. It can also reveal how much money a training program, improvement initiative, or new organizational design yields. The ROI of SPI is an invaluable measurement instrument for stakeholders at all levels of a corporation and organization. ROI enables stakeholders to design the most effective strategies to achieve the maximum benefits. The benefits are often expressed in terms of productivity, quality, profits, and peak operating efficiency.

II. IMPORTANCE OF ROI

The ROI of SPI is important because it is a process of calculating the amount of money to be gained from a new process. It can also be used to determine how much money is lost from creating and using a new and improved software process. The ROI of SPI for a new software process can be surprisingly large, sometimes negligible, or sobering negative. The software market is often fraught with dangerous fluctuations, conditions, and confusing indicators. Corporations can avert financial catastrophe and seize their rightful titles as captains of industry by applying the ROI of SPI.

The ROI of SPI is useful for large non profit organizations and government institutions. The ROI of SPI helps them manage their hard-won resources, capitalization, and funding. It does this by helping them directing to a new software process that has negligible effects on peak operating efficiency. However, it is important to note that the ROI of SPI is merely one of many tools to support critical decision-making processes. There are many tools to support decision making when it comes to selecting a new and improved software process. In fact, it may be necessary for a corporation or organization to create a new process that has a negligible or even negative ROI. This is often done because it is socio-politically correct or part of a mandatory government regulation or industrial trade agreement. Sometimes, inefficient processes are tied to other forms of economic incentives, motivations, or gains.

The ROI of SPI is determined by calculating the ratio of benefits to costs for creating a new and improved software process. The ROI of SPI is a simple ratio of all of the benefits to all of the costs for a new and improved software process. The exact formulas for the ROI of SPI involve subtracting the costs from the benefits before stating the ratio of benefits to costs. The formulas are nonetheless very simple and easy to use. The benefits for a new and improved software process are usually increases in product variety, portfolio size, and market share. The benefits also include increases in customer satisfaction, productivity, efficiency, quality, and reliability. Decreases in costs, cycle times, and process complexity are other important benefits.

The costs of a new and improved software process include strategic planning, education, and designing new processes. Additional costs include process and development tools, consultants, training, travel, facilities, lost productivity, and project simulation. Costs also include salaries, actual project effort, sociopolitical resistance, and preparation for appraisals and external audits. We also consider the costs of appraisals and audits, action plans, re audits and re-certification, and software process maintenance. The benefits of some SPI methods are quite large, so we should not be discouraged by the overwhelming costs.

III. KEY PRINCIPLES OF ROI OF SPI

Key principles of ROI of SPI include application of basic ROI formulas, analysis of ROI inputs, and focusing on benefits of SPI. Key principles also include analysis of simple defect models, cost of quality, total life cycle costs, and pervasive defect prevention. ROI formulas contain only two fundamental terms: benefits and costs. Cost is by far the easiest of the terms to grasp, so focus on the benefits. Where does information on benefits come from? Benefit data come from analyzing SPI studies, conducting literature surveys, and collecting cost-benefit data. Information also comes from ROI studies, examining the cost of quality, performing cost modeling, and studying defect models. The benefits of SPI come from two basic sources, increased revenue and profits and decreased costs and cost savings. The benefits of SPI originating from increased revenue and profits are primarily due to increased productivity. That is, increased output or work products per unit time. The benefits of SPI originating from decreased costs and cost savings are due to less maintenance, rework, and testing. This leads to shorter cycle times and faster schedules. In the subsequent sections we present 4 ROI Methodologies for software process Improvement.

IV. SOFTWARE INSPECTION PROCESS ROI METHODOLOGY

The return on investment (ROI) methodology for the Software Inspection Process is a procedure to measure, quantify, and analyze its economic value. The Software Inspection Process is a type of meeting which is held in order to identify defects in software work products. ROI is the amount of money gained, returned, or earned above the resources spent on the Software Inspection Process.

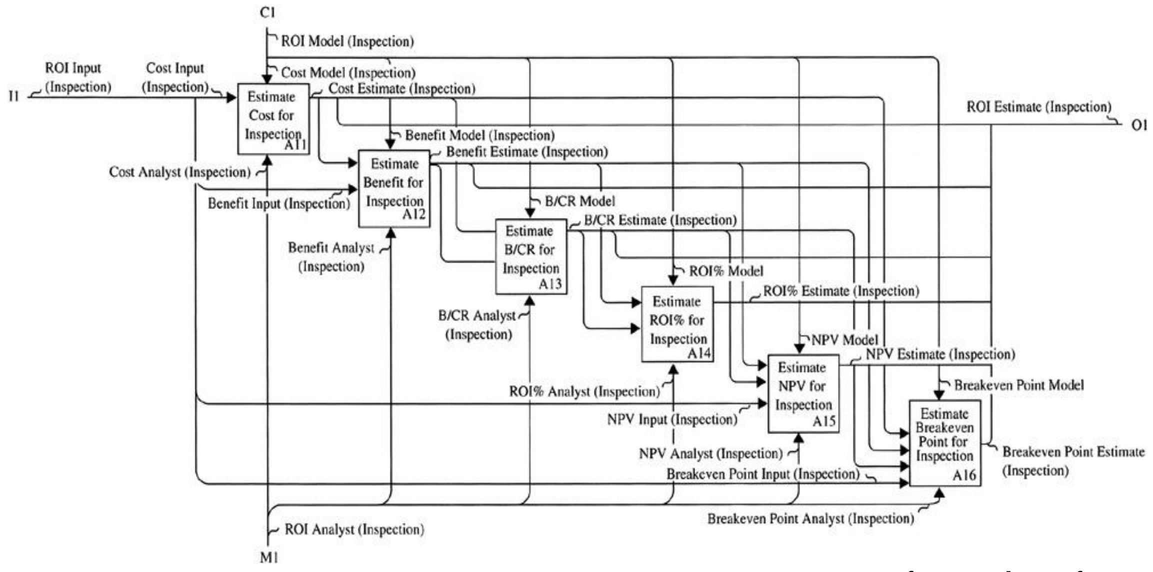

C1
ROI Model (Inspection)
Cost Model (Inspection)
ROI Input (Inspection)
Cost Input (Inspection)
I1
Estimate Cost for Inspection A11
Cost Estimate (Inspection)
ROI Estimate (Inspection)
O1
Benefit Model (Inspection)
Benefit Estimate (Inspection)
Cost Analyst (Inspection)
Estimate Benefit for Inspection A12
Benefit Input (Inspection)
B/CR Model
B/CR Estimate (Inspection)
Benefit Analyst (Inspection)
Estimate B/CR for Inspection A13
ROI% Model
ROI% Estimate (Inspection)
B/CR Analyst (Inspection)
Estimate ROI% for Inspection A14
NPV Model
NPV Estimate (Inspection)
Breakeven Point Model
ROI% Analyst (Inspection)
Estimate NPV for Inspection A15
NPV Input (Inspection)
Estimate Breakeven Point for Inspection A16
Breakeven Point Estimate (Inspection)
NPV Analyst (Inspection)
Breakeven Point Input (Inspection)
Breakeven Point Analyst (Inspection)
ROI Analyst (Inspection)
M1

Fig. 1. ROI Methodology for Software Inspection Process

Its ROI methodology is a six-part process that consists of estimating costs, benefits, benefit/cost ratio (B/CR), return on investment percent (ROI%), net present value (NPV), and breakeven point. The ROI methodology for the Software Inspection Process has unique elements for estimating costs, benefits, and B/CR. Fig 1 see the ROI methodology for the Software Inspection Process.

A. *Inspection Cost Methodology*

The cost methodology for the Software Inspection Process is a procedure to measure, quantify, and analyze the amount of money spent. The Software Inspection Process incurs cost to find defects, but results in lower overall software maintenance costs. Cost is the economic consequence of using the Software Inspection Process to create a new and improved software process. Its cost methodology is a five-part process that consists of estimating training, software, meeting, test, and maintenance costs. The cost methodology for the Software Inspection Process has unique elements for estimating meeting and software maintenance costs. Its cost methodology consists of a complex composite of uniquely interrelated software cost and defect models. Fig 2 shows the cost methodology for the Software Inspection Process.

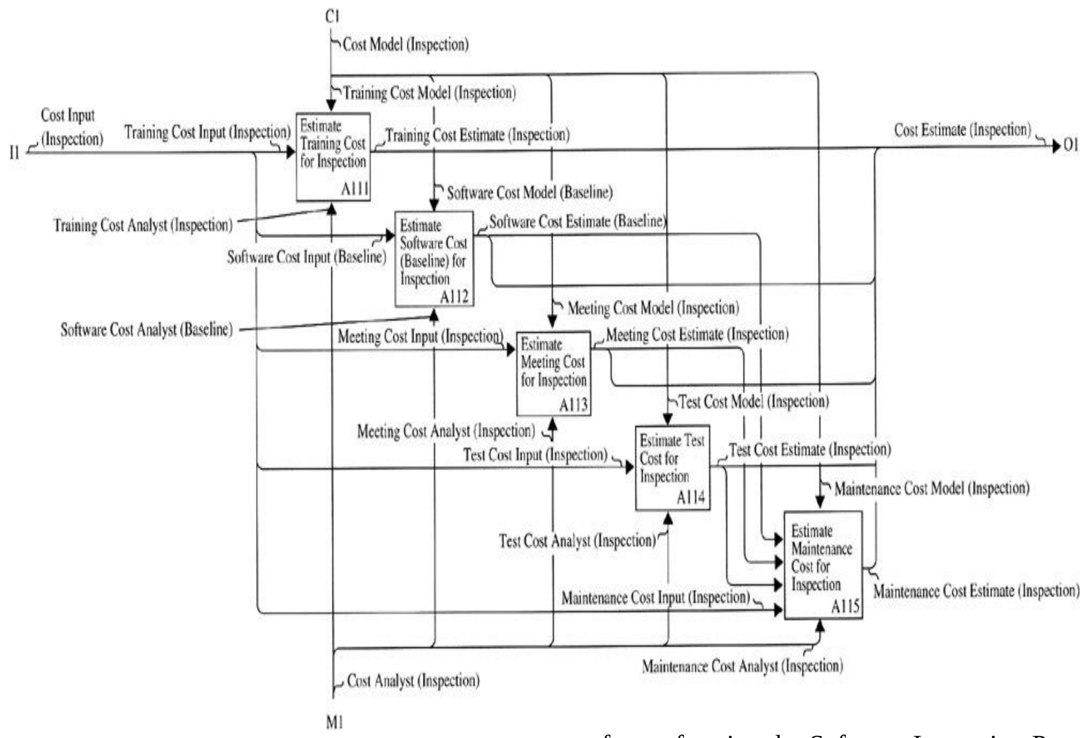

C1
Cost Model (Inspection)
Training Cost Model (Inspection)
Cost Input (Inspection)
I1
Training Cost Input (Inspection)
Estimate Training Cost for Inspection
A111
Training Cost Estimate (Inspection)
Cost Estimate (Inspection)
O1
Software Cost Model (Baseline)
Training Cost Analyst (Inspection)
Estimate Software Cost (Baseline) for Inspection
A112
Software Cost Estimate (Baseline)
Software Cost Input (Baseline)
Meeting Cost Model (Inspection)
Software Cost Analyst (Baseline)
Meeting Cost Input (Inspection)
Estimate Meeting Cost for Inspection
A113
Meeting Cost Estimate (Inspection)
Test Cost Model (Inspection)
Meeting Cost Analyst (Inspection)
Test Cost Input (Inspection)
Estimate Test Cost for Inspection
A114
Test Cost Estimate (Inspection)
Maintenance Cost Model (Inspection)
Test Cost Analyst (Inspection)
Estimate Maintenance Cost for Inspection
A115
Maintenance Cost Input (Inspection)
Maintenance Cost Estimate (Inspection)
Maintenance Cost Analyst (Inspection)
Cost Analyst (Inspection)
M1

Fig. 2. Cost Methodology for Software Inspection Process

Estimate training cost for inspection: The objective of this activity is to estimate all of the training costs for the Software Inspection Process. This sub step includes: multiply training rate, participant, and effort for inspection and add training time and training fee cost for inspection.

Estimate software cost (baseline) for inspection: The objective of this activity is to estimate the costs of software analysis, design, and implementation. This sub steps includes: estimate software cost (Boehm) for inspection, estimate software cost (Walston/Felix) for inspection, estimate software cost (Bailey/Basili) for inspection, estimate software cost (Doty) for inspection, and estimate software cost (average) for inspection. In this case, the outputs of the software cost models by Boehm, Walston/Felix, Bailey/Basili, and Doty are averaged together.

Estimate meeting cost for inspection: The objective of this activity is to estimate the cost for performing the Software Inspection Process. This sub steps includes: estimate meeting cost (BNR) for inspection, estimate meeting cost (Gilb) for inspection, estimate meeting cost (AT&T) for inspection, estimate meeting cost (HP) for inspection, estimate meeting cost (Rico) for inspection, and estimate meeting cost (average) for inspection. The outputs of the BNR, Gilb, AT&T, HP, and Rico Software Inspection Process cost models are averaged together.

Estimate test cost for inspection: The objective of this activity is to estimate the cost of software testing based on defects escaping the Software Inspection Process. This sub steps include: estimate starting defects for inspection, estimate meeting efficiency for inspection, estimate pre-test defects for inspection, estimate test efficiency for inspection, estimate post-test defects for inspection, and estimate test cost (projected) for inspection.

Estimate maintenance cost for inspection: The objective of this activity is to estimate the cost of software maintenance based on defects escaping the Software Inspection Process and testing. This sub steps include: estimate total life cycle cost for inspection and estimate maintenance cost (projected) for inspection.

B. Inspection Benefit Methodology

The benefit methodology for the Software Inspection Process is a procedure to measure, quantify, and analyze the amount of money returned. The Software Inspection Process eliminates defects early, when they are least expensive, resulting in lower maintenance costs. Benefit is the economic value of using the Software Inspection Process to create a new and improved software process. Its benefit methodology is a three-part process that consists of estimating test costs, total life cycle costs of testing, and benefits. Its benefit methodology consists of a variety of defect models used in combination. Fig 3 see the benefit methodology for the Software Inspection Process.

Estimate test cost (baseline) for inspection: The objective of this activity is to estimate the cost of removing the maximum number of software defects using software testing. The sub steps include: estimate post-test defects (baseline) for inspection; and estimate baseline test cost (projected) for inspection.

Estimate total life cycle cost (test) for inspection: The objective of this activity is to estimate software development and maintenance costs associated with using software testing. The sub steps include: estimate total software cost (test) for inspection; estimate total test cost (test) for inspection; and subtract total test from software cost for inspection.

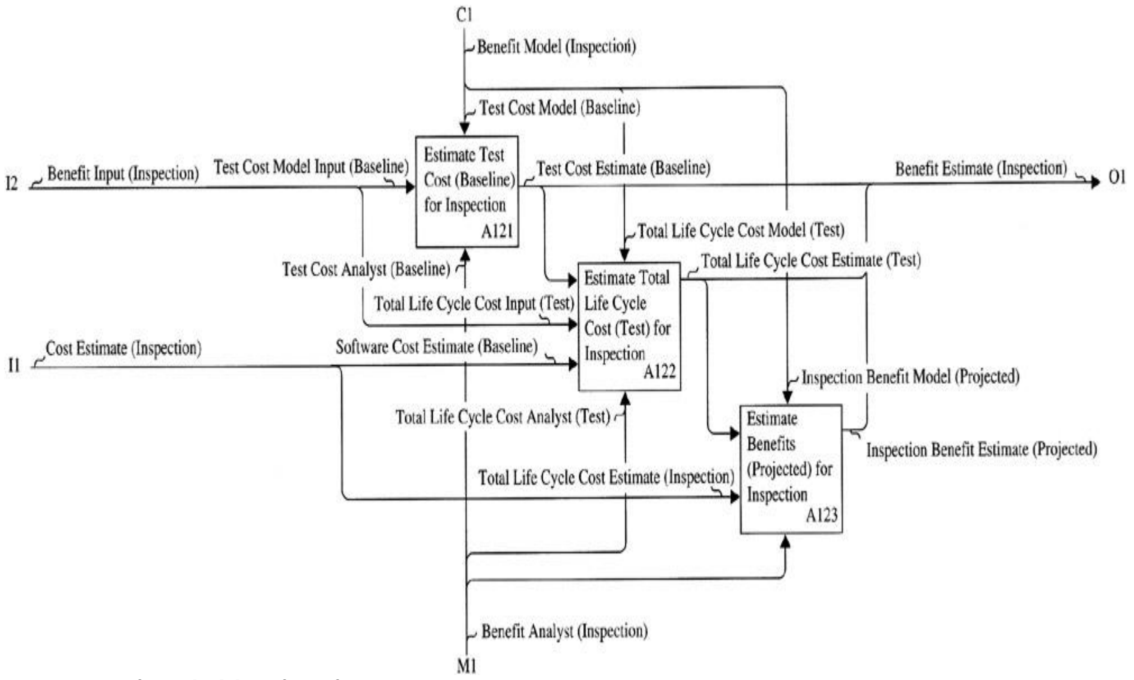

C1
Benefit Model (Inspection)
Test Cost Model (Baseline)
I2
Benefit Input (Inspection)
Test Cost Model Input (Baseline)
Estimate Test Cost (Baseline) for Inspection
A121
Test Cost Estimate (Baseline)
Benefit Estimate (Inspection)
O1
Total Life Cycle Cost Model (Test)
Test Cost Analyst (Baseline)
Total Life Cycle Cost Estimate (Test)
Total Life Cycle Cost Input (Test)
Estimate Total Life Cycle Cost (Test) for Inspection
A122
I1
Cost Estimate (Inspection)
Software Cost Estimate (Baseline)
Inspection Benefit Model (Projected)
Total Life Cycle Cost Analyst (Test)
Estimate Benefits (Projected) for Inspection
A123
Inspection Benefit Estimate (Projected)
Total Life Cycle Cost Estimate (Inspection)
Benefit Analyst (Inspection)
M1

Fig. 3. Benefit Methodology for Software Inspection Process

Estimate benefits (projected) for inspection: The objective of this activity is to compare software development and maintenance costs of testing and the Software Inspection Process. This sub steps include: subtract total life cycle cost of inspection from test for inspection.

C. Inspection B/CR Methodology

The B/CR methodology for the Software Inspection Process is a procedure to measure, quantify, and analyze the ratio of benefits to costs. The ratio of benefits to costs for the Software Inspection Process is high because of reductions in software maintenance costs. B/CR is the economic magnitude of using the Software Inspection Process to create a new and improved software process. Its B/CR methodology is a two-part process that consists of estimating special costs and the benefit to cost ratio. Fig 4 see the B/CR methodology for the Software Inspection Process.

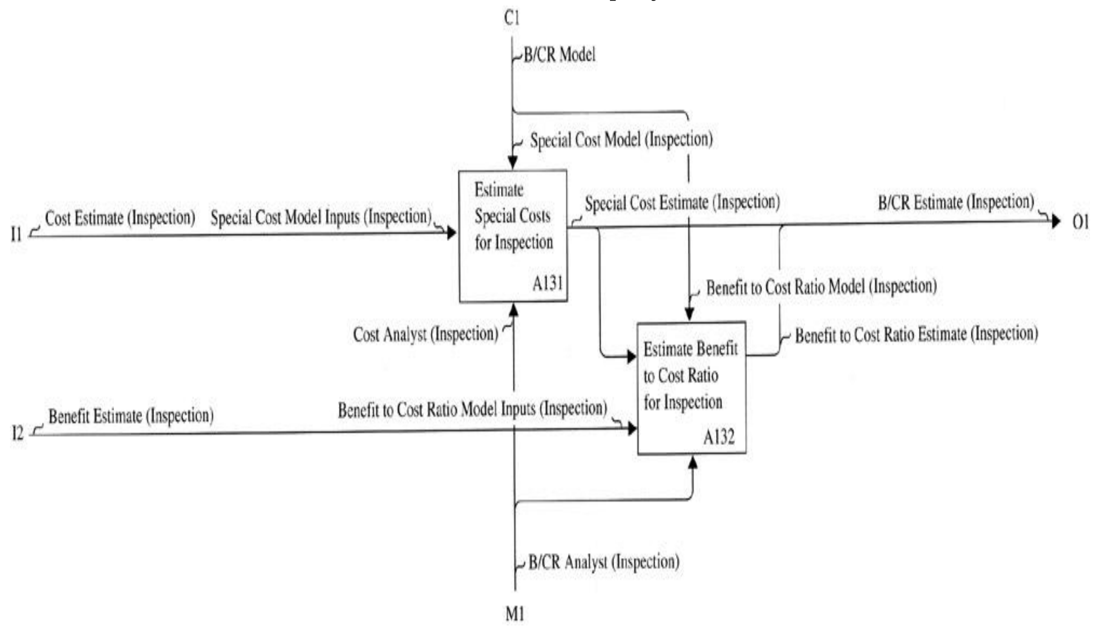

C1
B/CR Model
Special Cost Model (Inspection)
I1
Cost Estimate (Inspection)
Special Cost Model Inputs (Inspection)
Estimate Special Costs for Inspection
A131
Special Cost Estimate (Inspection)
B/CR Estimate (Inspection)
O1
Benefit to Cost Ratio Model (Inspection)
Cost Analyst (Inspection)
Estimate Benefit to Cost Ratio for Inspection
A132
Benefit to Cost Ratio Estimate (Inspection)
I2
Benefit Estimate (Inspection)
Benefit to Cost Ratio Model Inputs (Inspection)
B/CR Analyst (Inspection)
M1

Fig. 4. Benefit/Cost Ratio Methodology for Software Inspection Process

Estimate special costs for inspection: The objective of this activity is to identify and separate the costs associated with the Software Inspection Process from common software development costs. This sub steps includes: add training and meeting costs for inspection.

Estimate benefit to cost ratio for inspection: The objective of this activity is to measure the magnitude of the benefits to the costs for implementing the Software Inspection Process. This sub steps includes: divide benefits by special costs for inspection.

D. Inspection Breakeven Point Methodology

The breakeven point methodology for the Software Inspection Process is a procedure to determine when benefits exceed costs. The benefits for the Software Inspection Process rapidly exceed its cost due to reduced maintenance costs. Breakeven point is the value at which the benefits overtake the costs for using the Software Inspection Process. Its breakeven point methodology is a four-part process that consists of estimating productivity and the cost to productivity difference ratio. Its breakeven point methodology consists of combining costs with productivity difference using the breakeven point formula.

Estimate testing productivity for inspection: The objective of this activity is to determine the software

productivity associated with using the software testing process. This sub steps include: divide software size by total life cycle cost of testing for inspection.

Estimate inspection productivity for inspection: The objective of this activity is to determine the software productivity associated with using the Software Inspection Process. This sub steps include: divide software size by total life cycle cost of inspection for inspection.

Estimate productivity difference for inspection: The objective of this activity is to compare the productivity of software testing to that of the Software Inspection Process. This sub steps include: divide test productivity by inspection productivity for inspection.

Estimate cost to productivity difference ratio for inspection: The objective of this activity is to determine when the Software Inspection Process will begin paying for itself. This sub steps include: divide special cost by one less productivity difference for inspection.

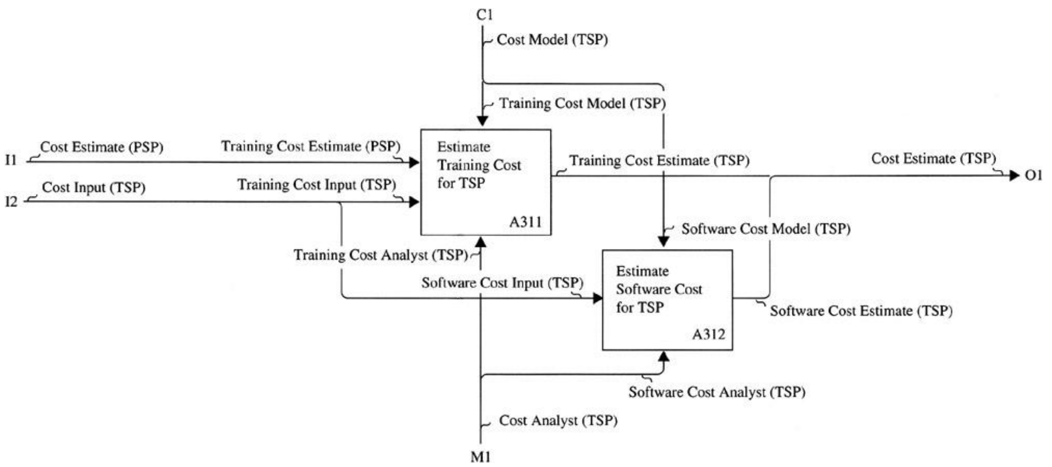
V. TEAM SOFTWARE PROCESS^{SM} ROI METHODOLOGY

The return on investment (ROI) methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze its economic value. The Team Software Process^{SM} is a project, quality, and life cycle management method for large groups of software engineers. ROI is the amount of money gained, returned, or earned above the resources spent on the Team Software Process^{SM}. Its ROI methodology is a six-part process that consists of estimating costs, benefits, benefit/cost ratio (B/CR), ROI%, net present value (NPV), and breakeven point. Fig 5 shows the ROI methodology for the Team Software Process^{SM}.

A. TSP^{SM} Cost Methodology

The cost methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze the amount of money spent. The Team Software Process^{SM} incurs cost for training, but results in one of the lowest overall software maintenance costs. Cost is the economic consequence of using the Team Software Process^{SM} to create a new and improved software process. Its cost

Fig 5. Cost Methodology for Team Software Process^{SM}


C1
Cost Model (TSP)
Training Cost Model (TSP)
I1
Cost Estimate (PSP)
Training Cost Estimate (PSP)
I2
Cost Input (TSP)
Training Cost Input (TSP)
Estimate Training Cost for TSP
A311
Training Cost Estimate (TSP)
Cost Estimate (TSP)
O1
Software Cost Model (TSP)
Training Cost Analyst (TSP)
Software Cost Input (TSP)
Estimate Software Cost for TSP
A312
Software Cost Estimate (TSP)
Software Cost Analyst (TSP)
Cost Analyst (TSP)
M1

methodology is a two-part process that consists of estimating training and software costs. The cost methodology for the Team Software Process^{SM} has unique elements for software development costs.

Estimate training cost for TSP^{SM}: The objective of this activity is to estimate all of the training costs for TSP^{SM}. This sub steps include: estimate training time cost for TSP^{SM}, estimate travel cost for TSP^{SM}, and add training time, travel, and Personal Software Process^{SM} training cost for TSP^{SM}.

Estimate software cost for TSP^{SM}: The objective of this activity is to estimate the costs of software analysis, design, implementation, and test. This sub steps include: divide software size by productivity for TSP^{SM}.

B. TSP^{SM} Benefit Methodology

The benefit methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze the amount of money returned. The Team Software Process^{SM} eliminates defects early and efficiently, when they are least expensive, resulting in no maintenance costs. Benefit is the economic value of using the Team Software Process^{SM} to create a new and improved software process. Its benefit methodology is a four-part process that consists of estimating baseline software costs, test costs, total life cycle costs of testing, and benefits. Its benefit methodology consists of a variety of defect models used in combination. Fig 10 shows the benefit methodology for the Team Software Process^{SM}.

Estimate software cost (baseline) for TSP^{SM}: The objective of this activity is to estimate typical costs of software analysis, design, and implementation. This sub steps include: estimate software cost (Boehm) for TSP^{SM}, estimate software cost (Walston/Felix) for TSP^{SM}, estimate software cost (Bailey/Basili) for TSP^{SM}, estimate software cost (Doty) for TSP^{SM}, and estimate software cost (average) for TSP^{SM}.In this case, the outputs of software cost models by Boehm, Walston/Felix, Bailey/Basili, and Doty are averaged together.

Estimate test cost (baseline) for TSP^{SM}: The objective of this activity is to estimate the cost of removing the maximum number of software defects using software testing. This sub steps include: estimate post-test defects (baseline) for TSP^{SM} and estimate baseline test cost (projected) for TSP^{SM}.

Estimate total life cycle cost (test) for TSP^{SM}: The objective of this activity is to estimate software development and maintenance costs associated with using software testing. This sub steps include: estimate total software cost (test) for TSP^{SM}, estimate total test cost (test) for TSP^{SM}, and subtract total test from software cost for TSP^{SM}.

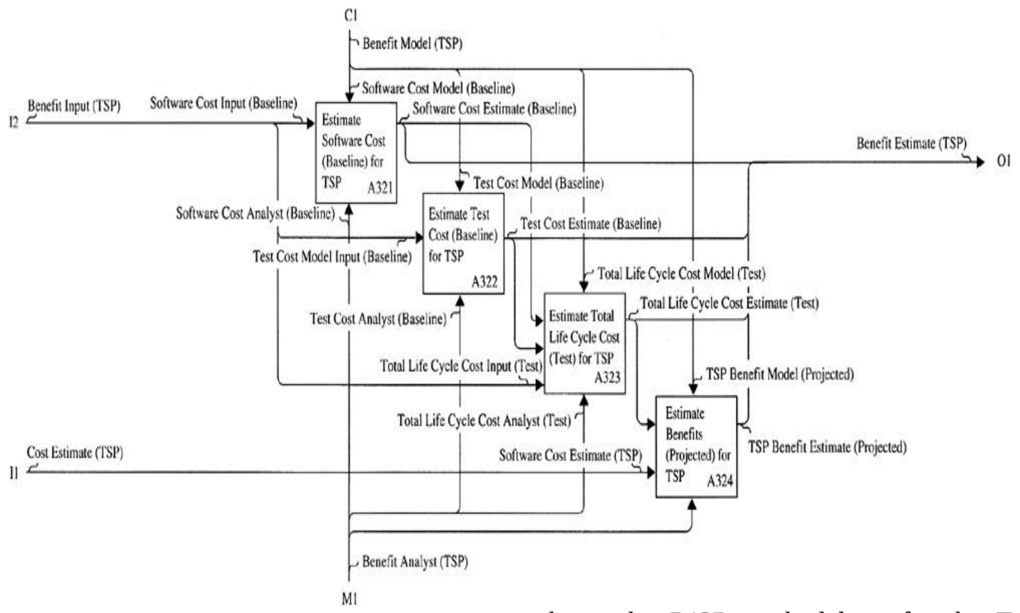

C1
Benefit Model (TSP)
Software Cost Model (Baseline)
Benefit Input (TSP)
I2
Software Cost Input (Baseline)
Estimate Software Cost (Baseline) for TSP
A321
Software Cost Estimate (Baseline)
Benefit Estimate (TSP)
O1
Test Cost Model (Baseline)
Software Cost Analyst (Baseline)
Estimate Test Cost (Baseline) for TSP
A322
Test Cost Estimate (Baseline)
Test Cost Model Input (Baseline)
Total Life Cycle Cost Model (Test)
Total Life Cycle Cost Estimate (Test)
Test Cost Analyst (Baseline)
Estimate Total Life Cycle Cost (Test) for TSP
A323
Total Life Cycle Cost Input (Test)
TSP Benefit Model (Projected)
Total Life Cycle Cost Analyst (Test)
Estimate Benefits (Projected) for TSP
A324
TSP Benefit Estimate (Projected)
Cost Estimate (TSP)
I1
Software Cost Estimate (TSP)
Benefit Analyst (TSP)
M1

Fig. 6. Benefit Methodology for Team Software Process^{SM}

Estimate benefits (projected) for TSP^{SM}: The objective of this activity is to compare development and maintenance costs of testing and TSP^{SM}. This sub steps include: subtract total life cycle cost of TSP^{SM} from test for TSP^{SM}.

C. TSP^{SM} B/CR Methodology

The B/CR methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze the ratio of benefits to costs. The ratio of benefits to costs for the Team Software Process^{SM} is high because maintenance costs are eliminated. B/CR is the economic magnitude of using the Team Software Process^{SM} to create a new and improved software process. Its B/CR methodology is a two-part process that consists of estimating special costs and the B/CR. Fig.7 shows the B/CR methodology for the Team Software Process^{SM}.

Estimate special costs for TSP^{SM}: The objective of this activity is to identify and separate the costs associated with TSP^{SM} from common software development costs. This sub steps include: identify training cost for TSP^{SM}.

Estimate B/CR for TSP^{SM}: The objective of this activity is to measure the magnitude of the benefits to the costs for implementing TSP^{SM}. This sub steps include: divide benefits by special costs for TSP^{SM}.

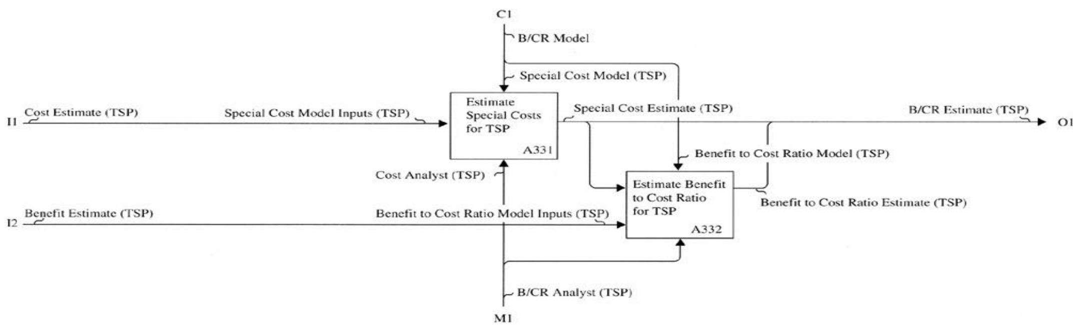

C1
B/CR Model
Special Cost Model (TSP)
I1
Cost Estimate (TSP)
Special Cost Model Inputs (TSP)
Estimate Special Costs for TSP
A331
Special Cost Estimate (TSP)
B/CR Estimate (TSP)
O1
Benefit to Cost Ratio Model (TSP)
Cost Analyst (TSP)
Estimate Benefit to Cost Ratio for TSP
A332
Benefit to Cost Ratio Estimate (TSP)
I2
Benefit Estimate (TSP)
Benefit to Cost Ratio Model Inputs (TSP)
B/CR Analyst (TSP)
M1

Fig. 7. B/CR Methodology for Team Software Process

D. TSP^{SM} ROI% Methodology

The ROI% methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze the money returned. The ratio of net benefits to costs for the Team Software Process^{SM} is high due to vast software maintenance cost savings. ROI% is the money earned from using the Team Software Process^{SM} to create a new and improved software process. Its ROI% methodology is a two part process that consists of estimating the B/CR using net benefits versus gross benefits. Fig 12 shows the ROI% methodology for the Team Software Process^{SM}.

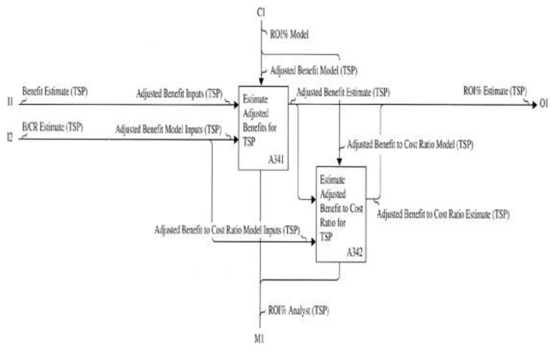
Estimate adjusted benefits for TSP^{SM}: The objective of this activity is to validate the benefits of TSP^{SM} by removing its costs. This sub steps include: subtract special costs from benefits for TSP^{SM}.

Estimate adjusted B/CR for TSP^{SM}: The objective of this activity is to measure the magnitude of the net benefits to the costs for implementing TSP^{SM}. This sub steps include: divide adjusted benefits by special costs for TSP^{SM}.

Fig. 8. ROI% Methodology for Team Software Process^{SM}

E. TSP^{SM} NPV Methodology

The NPV methodology for the Team Software Process^{SM} is a procedure to measure, quantify, and analyze money returned less inflation. The ratio of discounted benefits to costs for the


C1
ROI% Model
Adjusted Benefit Model (TSP)
I1
Benefit Estimate (TSP)
Adjusted Benefit Inputs (TSP)
I2
B/CR Estimate (TSP)
Adjusted Benefit Model Inputs (TSP)
Estimate Adjusted Benefits for TSP
A341
Adjusted Benefit Estimate (TSP)
ROI% Estimate (TSP)
O1
Adjusted Benefit to Cost Ratio Model (TSP)
Estimate Adjusted Benefit to Cost Ratio for TSP
A342
Adjusted Benefit to Cost Ratio Estimate (TSP)
Adjusted Benefit to Cost Ratio Model Inputs (TSP)
ROI% Analyst (TSP)
M1

Team Software Process^{SM} remains high due to reduced maintenance costs. NPV is the discounted money earned from using the Team Software Process^{SM} to create a new and improved software process. Fig 13 shows the NPV methodology for the Team Software Process^{SM}.

Estimate NPV of benefits for TSP^{SM}: The objective of this activity is to discount the gross benefits of TSP^{SM} based on inflation. This sub steps include: divide benefits by devaluation rate for TSP^{SM}.

Estimate adjusted NPV benefits for TSP^{SM}: The objective of this activity is to validate the benefits of TSP^{SM} by removing its costs. This sub steps include: subtract special costs from NPV benefits for TSP^{SM}.

Estimate adjusted NPV B/CR for TSP^{SM}: The objective of this activity is to measure the magnitude of the discounted net benefits to the costs for implementing TSP^{SM}. This sub steps include: divide adjusted NPV benefits by special costs for TSP^{SM}.

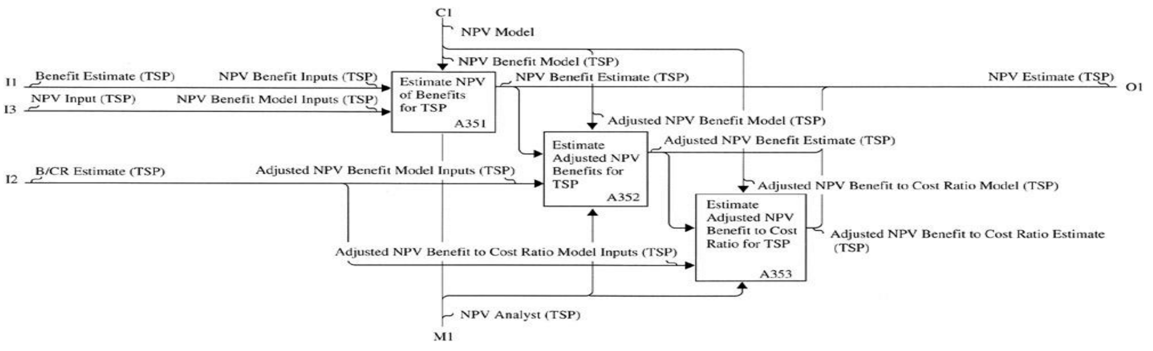

C1
NPV Model
NPV Benefit Model (TSP)
I1
Benefit Estimate (TSP)
NPV Benefit Inputs (TSP)
I3
NPV Input (TSP)
NPV Benefit Model Inputs (TSP)
Estimate NPV of Benefits for TSP
A351
NPV Benefit Estimate (TSP)
NPV Estimate (TSP)
O1
Adjusted NPV Benefit Model (TSP)
I2
B/CR Estimate (TSP)
Adjusted NPV Benefit Model Inputs (TSP)
Estimate Adjusted NPV Benefits for TSP
A352
Adjusted NPV Benefit Estimate (TSP)
Adjusted NPV Benefit to Cost Ratio Model (TSP)
Estimate Adjusted NPV Benefit to Cost Ratio for TSP
A353
Adjusted NPV Benefit to Cost Ratio Estimate (TSP)
Adjusted NPV Benefit to Cost Ratio Model Inputs (TSP)
NPV Analyst (TSP)
M1

Fig. 9. NPV Methodology for Team Software Process^{SM}

F. TSP^{SM} Breakeven Point Methodology

The breakeven point methodology for the Team Software Process^{SM} is a procedure to determine when benefits exceed costs. The benefits for the Team Software Process^{SM} rapidly exceed its cost due to reduced maintenance costs. Breakeven point is the value at which the benefits overtake the costs for using the Team Software Process^{SM}. Its breakeven point methodology is a four-part process that consists of estimating productivity and the cost to productivity difference ratio. Its breakeven point methodology consists of combining costs with productivity difference using the breakeven point formula. Fig 14 shows the breakeven point methodology for the Team Software Process^{SM}.

Estimate testing productivity for TSP^{SM}: The objective of this activity is to determine the software productivity associated with using the software testing process. This sub steps include: divide software size by total life cycle cost of testing for TSP^{SM}.

Estimate TSP^{SM} productivity for TSP^{SM}: The objective of this activity is to determine the software productivity associated with using TSP^{SM}. This sub steps include: divide software size by total life cycle cost of TSP^{SM} for TSP^{SM}.

Estimate productivity difference for TSP^{SM}: The objective of this activity is to compare the productivity of software testing to that of TSP^{SM}. This sub steps include: divide test productivity by TSP^{SM} productivity for TSP^{SM}.

Estimate cost to productivity difference ratio for TSP^{SM}: The objective of this activity is to determine when TSP^{SM} will begin paying for itself. This sub steps include: divide special cost by one less productivity difference for TSP^{SM}.

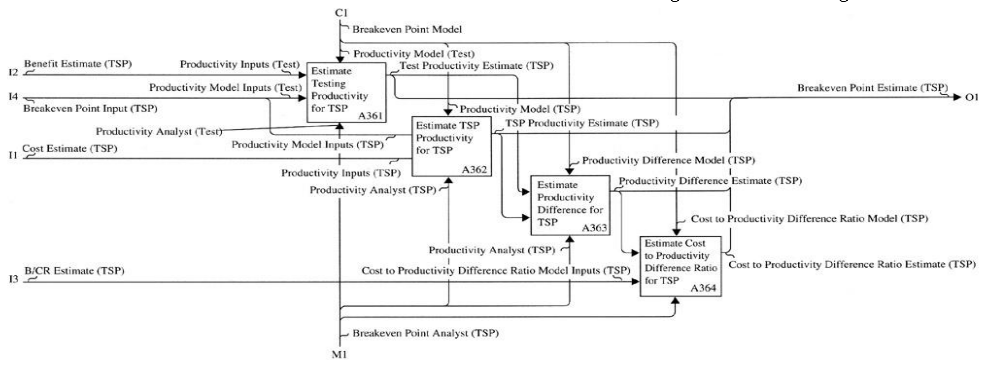

C1
Breakeven Point Model
Productivity Model (Test)
I2
Benefit Estimate (TSP)
Productivity Inputs (Test)
Estimate Testing Productivity for TSP
A361
Test Productivity Estimate (TSP)
I4
Productivity Model Inputs (Test)
Breakeven Point Input (TSP)
Breakeven Point Estimate (TSP)
O1
Productivity Model (TSP)
Productivity Analyst (Test)
Estimate TSP Productivity for TSP
A362
TSP Productivity Estimate (TSP)
I1
Cost Estimate (TSP)
Productivity Model Inputs (TSP)
Productivity Inputs (TSP)
Productivity Analyst (TSP)
Productivity Difference Model (TSP)
Estimate Productivity Difference for TSP
A363
Productivity Difference Estimate (TSP)
Cost to Productivity Difference Ratio Model (TSP)
Productivity Analyst (TSP)
Estimate Cost to Productivity Difference Ratio for TSP
A364
Cost to Productivity Difference Ratio Estimate (TSP)
I3
B/CR Estimate (TSP)
Cost to Productivity Difference Ratio Model Inputs (TSP)
Breakeven Point Analyst (TSP)
M1

Fig. 10. Breakeven Point Methodology for Team Software Process

VI. CHALLENGES IN DETERMINING ROI OF SPI

The Difficulties in determining the ROI of SPI are quite numerous. It is easy to believe that ROI is more than just the application of a few basic equations but unfortunately, many believe that the ROI of SPI and SPI economics are a nonessential part of their continuing SPI journey. Similarly, some people feel that ROI does not apply to them, their situation, or to the field of SPI. A few well-placed individuals think that there is no room for ROI principles in an ad hoc, immature organization. SPI is often perceived as too difficult and immeasurable, but SPI is worthy in spite of its impossibility. Some believe organizations must embark on a multiyear, multimillion-dollar ROI study by itinerant measurement scholars. Some feel that organizations cannot perform an early, top-down ROI analysis of their portfolio of software assets due to immaturity.

REFERENCES

[1] Bockle, G.; Clements, P.; McGregor, John D.; Muthig, Dirk; Schmid, K., "Calculating ROI for software product lines," Software, IEEE , vol.21, no.3, pp.23,31, May-June 2004J

[2] Van Solingen, R., "Measuring the ROI of software process improvement," Software, IEEE , vol.21, no.3, pp.32,38, May-June, 1997.

[3] (2004) The IEEE website. [Online]. Available: http://www.ieee.org/

[4] Boehm, B.; Huang, L.; Jain, A.; Madachy, R., "The ROI of software dependability: The iDAVE model," Software, IEEE , vol.21, no.3, pp.54,61, May-June 2004.

[5] D.F. Rico, ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers, J.Ross Publishing, 2004.

[6] K. El Emam and L. Briand, Cost and Benefits of Software Process Improvement, tech. report ISERN-97-12,Int’l Software Eng. Research Network, 1997.

Manas Kumar Yogi,a Member of IEEE,is currently working as Assistant Professor in Dept. of CSE , Pragati Engineering College,Surampalem,A.P. having more than 6 years of industry and teaching experience.His main area of research interest includes software engineering,Cloud computing,Computer Networks.

G.Vijay Kumar is currently working as Assistant Professor in Dept. of CSE , Pragati Engineering College,Surampalem,A.P. having more than 4 years of industry and teaching experience.His main area of research interest includes Distributed ,Cloud Computing,Computer Networks.