Simulator & Simulation Models for Cost-Benefit Analysis of Software Reuse Policies

P.K. Suri[1] and Neeraj Garg[2]

1 *Professor, Department of Computer Science & Applications, Kurukshetra University, Kurukshetra (Haryana) India,*
2 *Asst. Professor & Head, Department of Masters in Computer Science & Applications , Maharaja Agresen Institute of Management & Technology , Jagadhri(Haryana), India*

Summary
In this paper we a simulator is developed for the cost benefit analysis of the software before it is launched. Two cases are discussed . In one case the analysis is done on the basis of Bayes' Posterior Probability theorem during the development stage of the software. The decision are made whether to go with the implementation of the reuse policy or not based on the conditions of the economic environment and the word by the consultant.
In another case we have used the Decision under uncertainty principle of Laplace, Maximin(Minimax), Savage, Hurwicz criterions to reach to a certain decision that how many reusable functionalities can be added to a software so that the development costs are minimum and the profits are maximum.
The different cases are simulated on the simulator which takes into consideration the various factors which effect the decision and the environmental conditions. The inputs are either generated randomly by a random number generator or can be input by user. For calculating the costs the simulator makes use of an economic model.

Key Words

Software reusability, cost benefit analysis, software economic model, Bayes, posterior probability, decision under uncertainty , Laplace, Maximin(Minimax), Savage, Hurwicz criterions.

Introduction

Reuse is the application of existing solutions to new problems. Reuse can reduce the time spent in creating solutions by avoiding duplicated efforts. Frakes, notes that "using reusable software generally results in higher overall productivity" [1].
According to Poulin et al. "the financial benefit attributable to reuse during the development phase is 80 percent of the cost of developing new code" [2]. The benefits are not only realized in productivity but also in quality; software developed using existing components can be more reliable than those developed from scratch. However, the reusable components must exist before they can be reused. The problem of reuse, therefore, lies in the answer to the following question: What features make modules reusable, and how can one achieve such features in database design models? Software reuse is the use of existing software components to construct new systems [3].
Reusing existing parts or components is a standard part of software engineering and human problem solving in general. However, reuse in software development is more effective if practice formally [1]. Formal reuse implies that reuse must be viewed as a goal to strive for, not just a result that happens by chance. Before reuse can take place, the reusable components must exist in some form, and designers must be aware of their existence and the functionality they provide.
If formal reuse is part of an organization's overall development goals, then the software construction process is different; not only are developers tasked to find and use existing artifacts, they also have to assure that the final product can also be reused in future development.
Software engineering literature lists many different kinds of reuse, but one of the most comprehensive lists is the one provided by Prieto-Diaz [3].
Reuse can also be characterized by how the new system is actually built. A new system may be constructed by putting together existing components (compositional reuse), or by using high- level specifications and application and code generators to produce a new system (generative reuse).
In reuse, whatever artifact is reused, it may be used as-is, or it may be modified or extended to provide additional functionality. The reuse of components without any modification is termed black-box reuse. White-box reuse is when the component is modified before use. According to Prieto-Diaz, white-box reuse is prevalent in the current state of practice [3].
To achieve reusability , economic constraint is the major factor to decide whether an organization should go for a product with reusable components or reusability.

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

Measuring the reuse cost

The software industry is painfully realizing that a software reuse effort if not carefully planned and properly carried out, often times becomes an inhibitor rather than a catalyst to software productivity and quality. In order to be successful, not only must a reuse program be technically sound, It must also be economically worthwhile. After all, reducing costs and increasing quality were the two main factors that drove software reuse into the software mainstream.

There are many informal arguments that make software reuse an appealing and economically viable idea. But reuse is not for free. Reuse of software incurs costs that would not have to be made if software was developed from scratch and not to be used again.

From an economic perspective 'black box reuse' is considered as the only viable way of software reuse. In black box reuse, software assets are not modified internally; the only tailoring takes place via (configuration) parameters.

A reuse metric defines a way of measuring some attribute of developing software with reusable assets. Several software reuse metrics have been developed. The simulator uses the following metrics and models for calculating the costs.

Relative cost of reuse (RCR).

Assume that the cost to develop a new module equals one unit of effort. The portion of this effort that it takes to reuse an equivalent module without modifications is called *relative cost of reuse*.

*Relative cost of writing for reuse (RCWR).*Assume that the cost to develop a new module for one-time use equals one unit of effort. The portion of this effort that it takes to produce an equivalent reusable module is called *relative cost of writing for reuse.* In developing for reuse, extra effort is spent on several tasks. The relative extra effort for different tasks Margono and Rhoads

.

Gaffney and Durek propose a model for making a cost-benefit analysis of reuse **.** Their model assumes that the cost of the reuse program, including the additional cost to build reusable components is amortized across all future projects that will use the component. Their model further assumes a centrally maintained repository that must recover its cost by charging equal sums of money to the first *n* projects that use a component from the repository.

Reuse Econimoc Models

Any significant effort to implement software reuse must include a measurement program which allows the project management to assess how well the project is doing with respect to developing and using reusable software. **Basili [4]** lists the following key questions which a reuse measurement program should address What percentage of a system is made up of reusable components? How many and what changes were made to a reusable component in order to reuse it?3How much effort was required to: locate, understand, adapt, and integrate a reusable component? How often is a given reusable component reused?

The cost of developing reusable components is often a reason given by management for not encouraging the development of reusable components. While it is true that the economics of reuse are such that the initial development of reusable components often requires more effort than without such consideration, accounting for the cost associated with developing the reusable components is often not done fairly. When doing cost/benefit analysis for software reuse, one needs to consider the long-term benefits of reusable components and their associated cost. It has been argued that the cost of a reusable component should be amortized over all projects using the component **[5].**

The benefits of a reusable component apply to every project using the component. Thus, taking a strictly near-term view of reusable components overemphasizes their cost relative to their benefit. The true economics of reusable components lies in the number of times the component is reused across multiple projects.[9]

The simulator uses the SPC economic Model for calculating the costs . The SPC reuse economics model grew out of the Software Productivity Consortium (SPC), a consortium owned by a number of U.S. aerospace companies which builds and supports software tools to improve productivity [5][9].

In this model, the total cost of delivering a software product is considered equal to the cost of developing new software plus the cost of reusing existing software.

The cost of developing a software product relative to all new code (for which C = 1), C, is given by

$$C = (1-R)(1) + (R)(b+E/n)$$

Where ,Ris the proportion of reusable code in a software product ($0 < R < 1$); b is the cost of integrating reusable code relative to the creation and integration of all new code (for which b = 1) ($b \geq 0$);E is the cost of developing reusable code relative to the creation of all new code (for which E = 1) ($E \geq 0$); and *n* is the number of uses over which the cost of the reusable code is to be amortized ($n \geq 1$).

The term (1 -R) (1) indicates the relative cost of developing reusable portion of the software product. If a software product contains no reusable code (R = 0), then C = 1. On the other hand, even if a software product is built entirely on reusable components (R = 1),there is still cost associated with creating the product, i.e.,

C = (b + E/n).

Thus, according to this model, in order for a reuse effort to succeed, one must not only try to maximize R, but one must also try to minimize b and E and at the same time try to maximize n.

The SPC model further defines P, the relative productivity of creating a software product with some amount of reuse, as the inverse of C

P = 1/C

A study by **Favaro [6]** on experiences gained from a project that made use of a popular repository of reusable components indicated that depending on the implementation complexity of a reusable component, no can vary between 1.33 and 12.97. Based on this observation, it may take at least 13 reuses to recoup the investment made in developing the most complex component in the repository!

Unfortunately, there is no commonly accepted metric for assessing success of reuse. Even if the organization has decided to launch a product with reusability the question arises how much reusability should be added to the software. One way to measure the benefits of reuse is to develop the same project once with and once without reuse. If the same crew is used, then there is a learning effect that influences the project's performance. If not the same crew is used, then the expertise of the different individuals on the team affects the projects performance. A way out of this dilemma is by collecting statistics over a large number of projects as this is believed to even out effects of peculiarities of specific projects.

For this a detailed study of the cost-benefit analysis is required to be done with the help of a simulator and in this paper we have used the various decision criterions to reach to some decision.

As the computations involved are large and the data is probabilistic based in future possibilities; we have used different random number generators and a simulator developed in a high level language to help an organization to take an appropriate decision .

The various decision criterions [16] used in the simulator for taking decisions are discussed below:

Posterior (Bayes') Probabilities :

The probabilities used in the expected value criterion are usually determined from historical data. In some cases the probabilities are adjusted using information based on sampling or experimentation . These resulting probabilities are referred to as posterior (or Bayes') probabilities

Decision can be reached based on posterior probabilities in the following steps

Step 1:

The conditional probabilities of the case may be read or generated on a random number generator

$P\{aj|\ bi\}$

Step 2.

Compute the joint probabilities as

$P\{bi,aj\} = P\{\ aj|bi\}P\{bi\}$, *for all i and j*

Given the prior probabilities $P\{b1\}$ and $P\{b2\}$, the join probabilities are determined by multiplying the first and the second rows of the table in step 1 by $P\{b1\}$ and $P\{b2\}$ respectively

Step 3.

Compute absolute probabilities as

$$P\{a_j\} = \sum_{all\,i} P\{b_i, a_j\},\ for\ all\ j$$

Step 4.

Determine the desired posterior probabilities as

$$P\{b_i \mid a_j\} = \frac{P\{b_i \mid a_j\}}{P\{a_j\}}$$

Step 5.

Compute the cost benefit analysis and choose the policy with least loss or maximum profit.

Decision under uncertainty

Decision making under uncertainty involves alternative actions whose payoffs depend on the random states of nature . Specifically the payoff matrix of a decision problem with m alternative actions and *n* states can be represented as

	Y1	*Y2.*	...	*Yn*
x1	*a(x1,y1)*	*a(x1,y2)*	...	*a(x1,yn)*
x2	*a(x2,y1)*	*a(x2,y2)*	...	*a(x2,yn)*
x3	*a(x3,y1)*	*a(x3,y2)*	...	*a(x3,yn)*
.	.	.		.
.	.	.		.
.	.	.		.
xm	*a(xm,y1)*	*a(xm,y2)*	...	*a(xm,yn)*

The element *xi* represents action *i* and the element *Yj* represent the state of nature j . The payoff or the outcome is associated with the action *xi* and state *Yj* is *a (xi,Yj)*

The difference between making a decision under risk and the under uncertainty is that in the case of uncertainty , the probability distribution associated with the states *Yj , j = 1,2,...n,* is either unknown or cannot be determined . This lack of information has led to the development of the following criteria for analyzing the decision problem.

1. Laplace : Laplace criterion is based on the principle of insufficient reason . Because the probability distribution are not known , there is no reason to believe that the probabilities associated with the states of the nature are different . The alternatives are thus evaluated using the optimistic assumption that all states are equally likely to occur ie.

$$P\{y1\} = P\{y2\} = \ldots = P\{Yn\} = 1/n$$

Given that payoff *a(xi,Yj)* represents gain , the best alternative is the one that yields

$$\max_{xi} \left\{ \; 1/n \sum_{j=1}^{n} v(xi, yi) \; \right\}$$

If *a(xi,Yj)* represents loss, then minimization replaces maximization .

2. Maximin (Minimax) : criterion is based in the conservative attitude of making the best of the worst possible conditions . If *a(xi, Yj)* is loss, then we select the action that corresponds to the minimum criterion.

$$\min_{xj} \left\{ \; \max_{yj} a(xi, yi) \; \right\}$$

IF *a(xi,Yj)* is gain , we use the Maximin criterion given by

$$\max_{xj} \left\{ \; \min_{yj} a(xi, yi) \; \right\}$$

3. Savage Regret : criterion aims at moderating conservation in the Minimax (Maximin) criterion by replacing the (gain or loss) payoff matrix *a(xi,Yj)* with a loss (or regret) *r(xi, Yj)* matrix , using the following transformation:

$$r(x_i, y_j) = \begin{cases} a(x_i, y_j) - \min\limits_{x_k} \{a(x_k, y_j)\}, \text{ if } a \text{ is loss} \\ \max\limits_{x_k} \{a(x_k, y_j)\} - a(x_i, y_j), \text{ if } a \text{ is gain} \end{cases}$$

a_k

4. Hurwicz : criterion is designed to reflect decision-making attitudes ranging from the most optimistic to the most pessimistic (or conservative).

Define $0 \le \alpha \le 1$, and assume that a(xi, Yj) represents gain . Then the selected action must be associated with

$$\max_{x_i} \left\{ \alpha \max_{y_j} a(x_i, y_j) + (1-\alpha) \min_{y_j} a(x_i, y_j) \right\}$$

The parameter α is called the index of optimism .

If $\alpha = 0$, the criterion is conservative because it applies the regular Minimax criterion . If $\alpha = 1$, the criterion produces optimistic results because it seeks the best of the best conditions . We can adjust the degree of optimism (or pessimism) through the proper selection of the value of the α in the specified (0,1) range . In the absence of strong feeling of optimism or pessimism , α= 0.5, may be an appropriate choice.

If a(xi, Yj) represents loss, then the criterion is changed to

$$\min_{x_i} \left\{ \alpha \min_{y_j} a(x_i, y_j) + (1-\alpha) \max_{y_j} a(x_i, y_j) \right\}$$

Simulator for calculating cost and Taking Decisions

Simulation is a powerful tool for solving many problems. In this paper we have devised a simulator which will help in taking a decision on a particular software to be developed to make it cost effective. It will help the developer to access

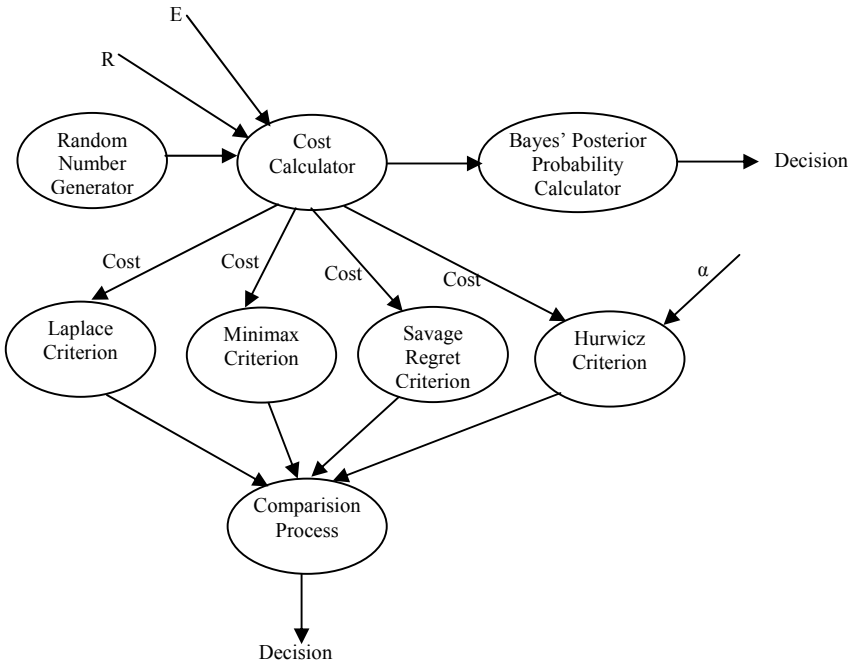


Fig.1 DFD for Cost Benefit Analysis of Reusable Software

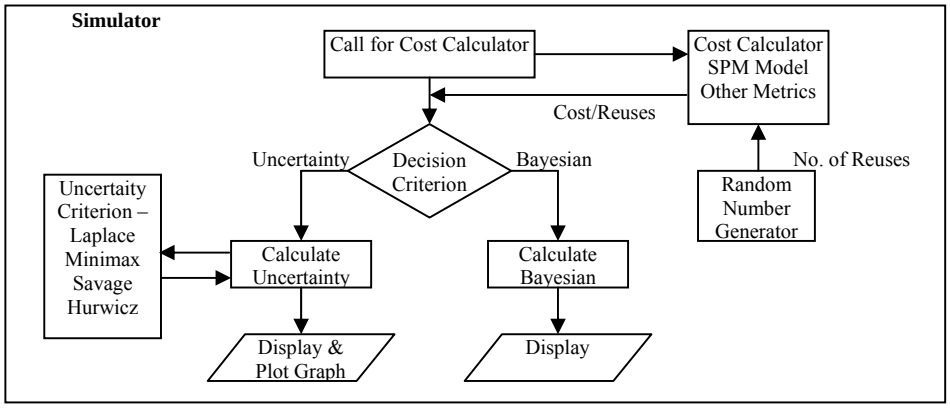


Fig 2. Simulator for Cost-Benefit Analysis

the costs vis-à-vis amount of reusability to be achieved in a software. The decision theory based on Bayse' posterior probability and equations of decision under uncertainty are used in the simulation. For calculating costs the simulator relies on the various metrics and models of cost calculations.

This typical Simulator includes the following components : a user interface, simulation models, a simulation engine, and output files. A user builds and changes a simulation model though the user interface. The user interface allows the user to save these models for later use. The user interface allows the user to specify parameters for the simulation runs and to start the simulation engine. Finally, the user interface provides methods for viewing and summarizing the output from one or more simulation runs.

By running the simulation multiple times, the trends in the project's evolution over time will be examined. By running the simulation with and without incorporating the selected modules, the differences will be analyzed and conclusions drawn about the impact of the process changes on the desired factors and also on Reusability.

Numeric results of the simulation are the direct outputs of deterministic computer programs, so reproducibility is absolute for the specific runs made. The conclusions drawn are thus supported by data that are completely reproducible.

The different cases can be simulated on the simulator which takes into consideration the various factors which effect the decision and the environmental conditions. The inputs are either generated randomly by a random number generator based on Normal Distribution or Poisson Distribution by applying inverse transformation functions in Poisson and that of Box Muller transformation in Normal distribution. The simulator has option of user inputs also.

Algorithms of the Simulator

1. start
2. input the choice –
   a)Bayes' Probability **or**
   b) decision under uncertainty
3. IF Choice is a) Bayes' Probability perform steps 4 to 12.
   ELSE Go to Step 13.
4. Input the cost involved
   Or
   Get the cost calculated by calling cost Calculator module based on the various cost Metrics
5. Input the total number of values and the numeric values of a and m.
6. Read the values of probability for ai
   Or
   Generate random number between 0 and 1 based on Box Muller transformation for the probability of ai
7. Display the table for conditional probabilities as $P\{ai|bi\}$
8. Calculate and display the joint probabilities as $P\{bi,aj\} = P\{ aj|bi\}P\{bi\}$, *for all i and j*
9. Calculate the Absolute probabilities and display the table for
   $P\{aj\} = \sum P\{ bi,aj\}$, *for all j*
10. Display and calculate the Posterior Probabilities based on the Bayes' formula as
    $P\{ bi, aj\} = P\{bi,aj\}/ P\{aj\}$
11. Calculate the cost analysis of the different options in step 9 and display at the decision
12. Stop , END.
13. Input the choice of criterions
14. IF one of the choice is Hurwicz
    Input the value of alpha.
15. Generate the cost matrix from the random number generator based on Box Muller transformations and by calling the Cost calculator Module of the Simulator
16. Calculate the Laplace , Minimax, Savage and Hurwicz parameters
17. Display the results and Decision
18. Stop, END.

Algorithm for the cost calculator Simulator

1. Start
2. Generate Random Number for the number of probable software reuses
3. Calculate the cost of reuse as
   C = (1-R)(1) + (R) (b+E/n).
4. Calculate the value of no the optimum number of reuses for maximum productivity
5. Display results or pass the results to the calling module of the Simulator

Case I:A company wants to launch a software with some initial investment . The company has two options : Option A – Launch the product with optimum reusability . Option B – Launch the product without reusability.If the market conditions are favorable ie the target customers will like the product and the various companies will be opting for the software of the company . The software with option A will Give more profit eg . say 50% (here investment is taken as 100,000) on investment and software with option B gives 15% of the profit .If the product is not liked by the customers ie . if the demand is low , in that case the company will be losing 20% of the investment made in the software due to additional effort of making it more reusable of it chooses option A. If the company chooses option B it will make a small profit of only 5% on investment. The management also hires a consultant to help in this decision and his recommendations are in the form -If demand is high go for option A with 0.9 probability and for 0.5 probability if demand is low. The company has to decide what should be done to maximize profit .

The problem is simulated on the simulator. A random number generator will generate various probabilistic possibilities of posterior probabilities. The decision whether the company should go for option of reusability or not will be based on the Bayes' posterior probability criterion.

Results form the Simulator:

```
Inputs :A1=for reusability,A2=
against reusability P{a1|b1}=
0.9,P{a2|b1)= 0.1,P{a1|b2}= 0.5,
P{a2|b2}= 0.5
```

Tab1e: "For' recommendation

Pb1	Pb2	A	B
0.1	0.9	-833	667
0.2	0.8	173	810
0.3	0.7	1048	935
0.4	0.6	1818	1045
0.5	0.5	-932	652
0.6	0.4	3108	1230
0.7	0.3	3653	1307
0.8	0.2	4146	1378
0.9	0.1	4593	1441

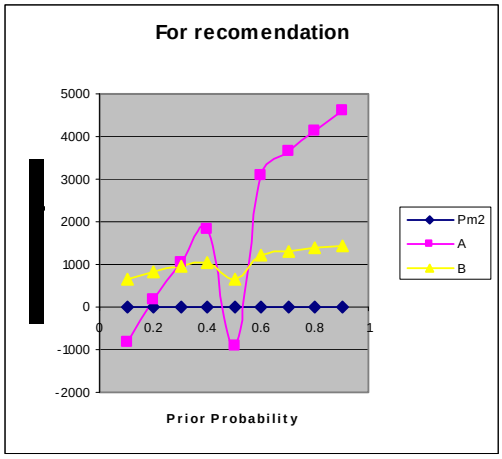


Fig:2 Variations in Optimum policies with different posterior probabilities('Against' Recommendation)

Tab2: "Against' recommendation

Pb1	Pb2	A	B
0.1	0.9	-1847	521
0.2	0.8	-1667	547
0.3	0.7	-1447	578
0.4	0.6	-1176	617
0.5	0.5	-833	667
0.6	0.4	-384	730
0.7	0.3	227	818
0.8	0.2	1111	944
0.9	0.1	2500	1142

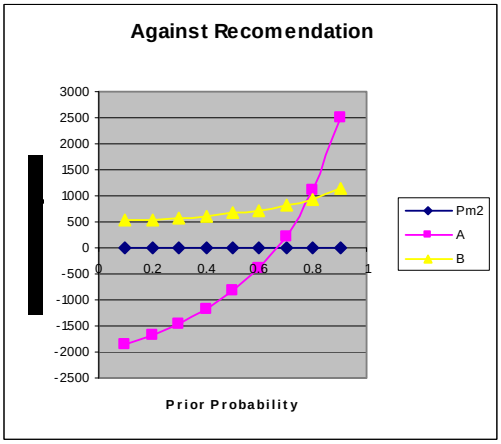


Fig: 3. Variations in Optimum policies with different posterior probabilities('Against' Recommendation)

Decision –
'go for the reuse policy and launch the product with reusability ie. option A'

'Against ' recommendation of the consultant

Option A = 31100
Option B= 731

Decision as simulated –
'go for the software without reuse policy ie. option B'

Case II

A company is preparing to launch a software with different levels of reusability (percentage of the number of lines of reusable code to the total no of lines) . The reuse policy will fall under four categories : 40%, 50%, 70 % and 80% . The cost of achieving higher level of reuse is more . There is demand for the software by different organizations and the on analysis of the requirements of the customers the company will earn profit by modifying the code according to the needs of the customer. The cost of the software will be lowest if it meets the requirements of the customer exactly.

Deviations above or below the ideal demand levels incur additional costs resulting from building surplus reusable code or losing income opportunities when the requirements are not met with reusable code . Let a1 to a4 represent the reuse levels (x1-40%. x2- 50%, x3-70%, x4 – 80%) , and Y1 to Y4(the requirements by users are randomly generated by the Simulator) , the requirements of the customers. The matrix below summarizes the cost matrix in terms of lac Rupees for this situation. We have taken one example case for explanation . The Simulator will take a decision based on several simulation runs with different input data.

Tab4: Cost Matrix for different Requirements

	Y1	Y2	Y3	Y4
x1	7	12	20	27
x2	10	9	14	25
x3	23	20	14	23
x4	32	24	21	17

Laplace .

P {Yj] = ¼, j=1 to 4 , the expected values for the different actions computed by the simulator are as

E{x1}= 16.5,E{x2= **15 ←Optimum,**E{x3} = 20
E{x4}= 23.5

Minimax

The Minimax criterion produces the following matrix

	Y1	Y2	Y3	Y4
x1	7	12	20	27
x2	10	9	14	25
x3	23	20	14	**23**
x4	32	24	21	17

The Minimax is 23 with row x3.

Savage:
The regret matrix is determined by
Subtracting 7,9,14 and 17 from columns
1 to 4 respectively . Thus the matrix as computed by simulator is

	Y1	Y2	Y3	Y4	Row Max
x1	0	3	6	10	10
x2	3	0	0	8	**8**
x3	16	11	0	6	16
x4	25	15	7	0	25

The x3 will give the minimum loss
Hurwicz:
The following table summarizes the computations

Alternative	Row Minimum	Row Maximum	H*
x1	7	27	27 -20α
x2	9	25	25 -16α
x3	14	23	23 – 9α
x4	17	32	32 -15α

* H= α (Row Min) + (1- α) (Row Max)

For α = 0.25

Alternative	Row Minimum	Row Maximum	H*
x1	7	27	22
x2	9	25	21
x3	14	23	**20.75**
x4	17	32	28.25

* H= α (Row Min) + (1- α) (Row Max)

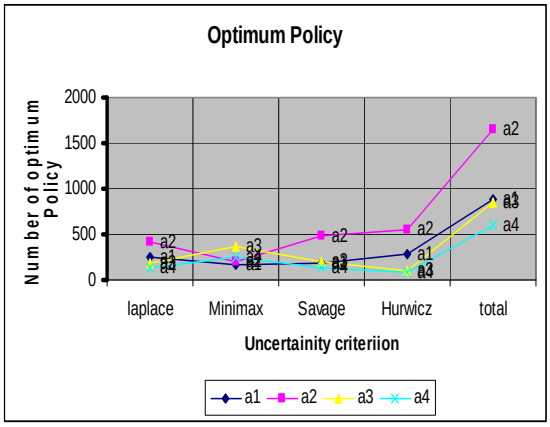
For α = 0.5

Alternative	Row Minimum	Row Maximum	H*
X1	7	27	**17**
X2	9	25	**17**
X3	14	23	18.5
X4	17	32	24.5

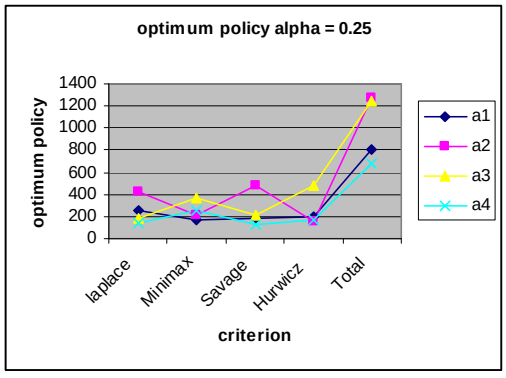
- H= α (Row Min) + (1- α) (Row Max)

```
Results from Simulator
Choice entered All
No of runs =1000


Optimism factor alpha=0.5
```



The choice of x2 is best suited as it is satisfied by Laplace, Savage and Hurwicz when alpha(level of Optimism) is 0.50



The choice of x2 and x3 is best suited. As x2 is satisfied by Laplace & Savage and x3 by Mimimax and Hurwicz when alpha(level of Optimism) is 0.25

Discussions and conclusion :

In the first case there are two policies – one in favor of reusability and the other against it. The results from the simulator give the exhaustive possibilities. The decision varies with the posterior probabilities . If the company goes with reusability it is profitable only if the probabilities i.e pb1>0.6 and pb2>0.4.

The company will not suffer any loss if it does not recommends reusability ie. The product is launched without reusability if pb1> 0.7 and pb2<0.3

In the second case the Simulator after 1000 runs recommends that the policy x2 which is better as per Laplace, Savage and Hurwicz criterion when level of optimism , alpha is 0.5.

In the second the simulator after 1000 recommends that the policies x3 and x2 both are suitable. The policy x2 is supported by Laplace and savage and x3 by minimax and Hurwicz when level of optimism , alpha is 0.25.

Thus the company can decide the optimum policy and Reusable components should be designed with the intent for reuse.

Reference:

[1] Frakes, W. Terry, C. Software Reuse: Metrics and Models. ACM Computing Surveys. vol. 28. No. 2, June 1996.

[2] Poulin et al. The business case for software reuse. IBM Systems Journal. Vol. 32, no. 4. 1993.

[3] Prieto-Diaz, Ruben. Status Report: Software Reusability. IEEE Software. May 1993.

[4] Basili, V. R., Rombach, H. D., Bailey, J., Joo, B.G., Software Reuse: A Framework, Department ofComputer Science, University of Maryland, 1987.

[5] Barnes, B., Durek, T., Gaffney, J., Pyster, A., A.Framework and Economic Foundation for Software Reuse, Software Productivity Consortium SPC-TN-87-011, Hemdon, Virginia, 1987.

[6] Favaro, J., "What Price Reusability? A Case Studyfl in the Proceedings of the First Symposium on Environments and Tools for Ada, Redondo Beach, California, 1990, pp. 115-124.

[7] Measuring Software Reuse: Principles, Practices andEconomic Models, Jeffrey Paulin, Addison Wesley,1997

[8] Gaffney, J. E. Jr. and Durek, T. A., Soflware Reuse- Key to Enhanced Productive@; Some QuantitativeModels, Software Productivity Consortium SPC-TR-88-015, Herndon, Virginia, 1988.

[9] Software Reuse Economics : Cost Benefit Analysis on a Large-scale ADA Project, Johan Margono, Thomas E. Rhoads Computer Sciences Corporation System Sciences Division FAA Advanced Automation SystemU. S. A.

[10] Dai, We. Development of Reusable Components: Preliminary Experience. Proceedings of the 17th International Conference on Software Engineering on Symposium on Software Reusability, 1995, pp. 238 - 246

[11] Daniani E. et al. A Hierarchy-Aware Approach to Faceted Classification of Object-Oriented Component. ACM Transactions on Software Engineering and Methodology. vol. 8, no. 3, July 1999, pp. 215-262

[12] Gillibrand, David, Liu, Kecheng. Quality Metric for Object-Oriented Design. Journal of Object-Oriented Programming. Jan 1998.

[13] Pressman, Roger S. Software Engineering: A Practitioner's Approach. 4 th ed. McGraw-Hill, 1997

[14] Adaptable Simulation Models for Manufacturing. W.Jeffery Herrmann 1 , Edward Lin 1 , Bala Ram 2 , Sanjiv Sarin . 2004
[15] Zage, M. Wayne, Zage, M. Dolores. Evaluating Design Metrics on Large-Scale Software. IEEE Software. 1993.
[16]Taha Hamdy A., Operations research an Introduction, Eighth Edition 2007, Pearson Education .

Prof(Dr.)P.K.Suri received his Ph.D.degree from Faculty Of Engineering Kurukshetra University, Kurukshetra, India and Master's degree from Indian Institute of Technology, Roorkee (formerly known as Roorkee University), India. He is working as Professor in the Department of Computer Science & Applications, Kurukshetra University, Kurukshetra - 136119 (Haryana), India since Oct. 1993. He has earlier worked as Reader, Computer Sc. & Applications, at Bhopal University, Bhopal from 1985-90. He has supervised five Ph.D.'s in Computer Science and thirteen students are working under his supervision. He has more than 100 publications in International / National Journals and Conferences. He is receipient of ''THE GEORGE OOMAN MEMORIAL PRIZE' for the year 1991-92 and a RESEARCH AWAWD – "The Certificate of Merit-2000" for the paper entitles ESMD- An Expert System for Medical Diagnosis from the Institution of Engineers, India. His Teaching and Research include Simulation and Modeling, SQA, Software Reliability , Software Testing & Software Engineering Process, Temporal Databases , Ad Hoc Networks , Grid Computing , and Biomechanics.

Neeraj Garg received his B.E. Degree and Masters in Computer Science and Applications (MCA) Panajb University , Chandigarh and Kurukshetra University, Kurukshetra in the year 1992 and 2001 respectively. Presently he is Head of the Department of MCA department at Maharaja Agresen Institute of Management and Technology , Jagadhri, Haryana, India . He had also worked with various organizations including C-DOT where he had carried out research work in SS#7 Protocol of Telephone Networks. He is co- editor of MAIMT- Journal of ITand Management. His research areas include Simulation and Modeling, Software engineering , System Programming and Networks.