5-1-2013

Ikriya: Simulating Software Quality Enhancement With Selected Replacement Policies

Sindhu Dharani Murthy
Western Kentucky University, sindhu.naydu@gmail.com

IKRIYA:
SIMULATING SOFTWARE QUALITY ENHANCEMENT WITH SELECTED REPLACEMENT POLICIES

A Thesis
Presented to
The Faculty of the Department of Computer Science
Western Kentucky University
Bowling Green, Kentucky

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Sindhu Dharani Murthy

May 2013

IKRIYA:
SIMULATING SOFTWARE QUALITY ENHANCEMENT
WITH SELECTED REPLACEMENT POLICIES

Date Recommended 2 April 2013

[illegible]
Dr. David W. Erbach, Director of Thesis

[illegible]
Dr. Qi Li

[illegible]
Dr. Rong Yang

[illegible] 4-30-13
Dean, Graduate Studies and Research Date

Dedicated to my mother, Kanchana, who is, and will always be, my greatest inspiration.

ACKNOWLEDGMENTS

I would like to convey my deepest gratitude and respect for my thesis and research advisor, Dr. David Erbach, for his overwhelming support and encouragement. This research would not have been possible without him. I would like to thank him for his patience and for trusting me with this research. Dr. David Erbach’s valuable comments helped me to develop a broader perspective of my thesis topic, and in learning beyond the textbooks. I have learned a lot from him and his teachings have been truly inspiring.

I gratefully acknowledge Dr. Qi Li and Dr. Rong Yang for their supervision and valuable time invested in reading and providing corrections to this thesis. I am very thankful that in the middle of their busy schedules, they accepted my invitation to be a part of my committee.

I would like to specially thank Dr. Guangming Xing for having faith in me and supporting me during difficult times.

I would like to thank and mention the people who have been my pillars of support, and who have always found time to help me. I thank Soujanya Siddavaram, Venkata Aditya Korada and Mohnish Thallavajhula for staying by my side during the good times and bad. Last but not least, I would like to thank my family, my parents and especially my brother Hari Prasad for absolute confidence in me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

IKRIYA:
SIMULATING SOFTWARE QUALITY ENHANCEMENT
WITH SELECTED REPLACEMENT POLICIES

Sindhu Dharani Murthy May 2013 65 Pages

Directed by: Dr. David W. Erbach, Dr. Qi Li, and Dr. Rong Yang

Department of Computer Science Western Kentucky University

The quality of information systems in any organization helps to determine the efficiency of the organization. Many organizations maintain a custom software portfolio, whose quality is important to the organization. Management would like to optimize the portfolio's quality.

Decisions about software replacement or enhancement are made based on organizational needs and priorities. The development resources allocated help in determining the quality of new software, and should be put to optimal use. Enhancing existing software might sound cheap and easy but it is not always efficient.

This thesis proposes a simulation model - iKriya - for this problem. It explores the consequences of various development and maintenance policies which might be applied. These depend on the state of existing software portfolio, the queue and properties of proposed projects, and the resources available. Optimal decisions are made by the simulator by taking the above mentioned factors into consideration.

1. INTRODUCTION TO THE PROBLEM

Organizations' products and services are increasingly supported by software. Consumers' needs and preferences change rapidly and this has led to a need for new software to support new products. The rapid change in technology makes software which is not improved gradually become outdated or obsolete. To satisfy developing consumer preferences and needs, new software should often be developed. This puts pressure on the people who manage technical services to develop new software quickly and often. But it is not efficient or easy to replace software with new each time it gets outdated.

Developing software requires the use of scarce and expensive resources. Hence organizations must evaluate existing software and try to identify places where software can be upgraded, since either new software can be developed or old software can be enhanced.

Software tends to "age" and loses some of its value because the organization's needs change, but the software stays the same. Organizations need to choose between developing new software and upgrading an existing one. Often it's much more feasible to enhance existing software rather than to develop new software. But sometimes a replacement is necessary.

Company managers have limited resources, so not everything can be done according to users' desires. Managers make decisions about new development or enhancement of old software based on the overall quality of the existing software and the urgency of proposed projects.

The following are examples of the different development policies could be applied:

- Divide resources equally between replacement and new development,
- Do all the projects which are considered to be urgent,
- Replace software which has fallen below a certain quality standard.

Managers would like to understand the result of different policies on the overall software portfolio. The purpose of this thesis is to develop a managerial tool which models the effects of crucial decisions in improving the quality of an organization's software. This is done by making a model which checks the software portfolio quality under different conditions.

The results from the simulations help in generating a profile for the software. Based on the profile, the core components of the software can be replaced or enhanced with better components. Replacements are done in order to somehow improve the quality of the software. If replacements are not possible then the necessary enhancements are developed. The goal is to understand the effects of different enhancement and replacement policies.

In order to enhance the overall software quality, we first need to test its current functionality. The major project of this thesis is to construct a tool which assists in making fundamental decisions, such as reuse of existing software by upgrading it, and not developing new software instead. Organizational resources of money and effort may be saved by following this procedure. This is especially helpful for organizations with limited resources or a restriction on the usage of resources. Details about the problem, model and the conclusions possible are examined in the later sections of the thesis.

1.1 THE MANAGERIAL DILEMMA

INFORMATION SYSTEMS

Companies use Information Systems (IS) to support major business functions and activities. IS helps to guide managers in decision making, coordination, control, and problem analysis.

The practical dilemma in IS management has at its source:

- There is always too much work to do.
- There are never sufficient resources.

Information Systems decision support software helps managers in overcoming these issues.

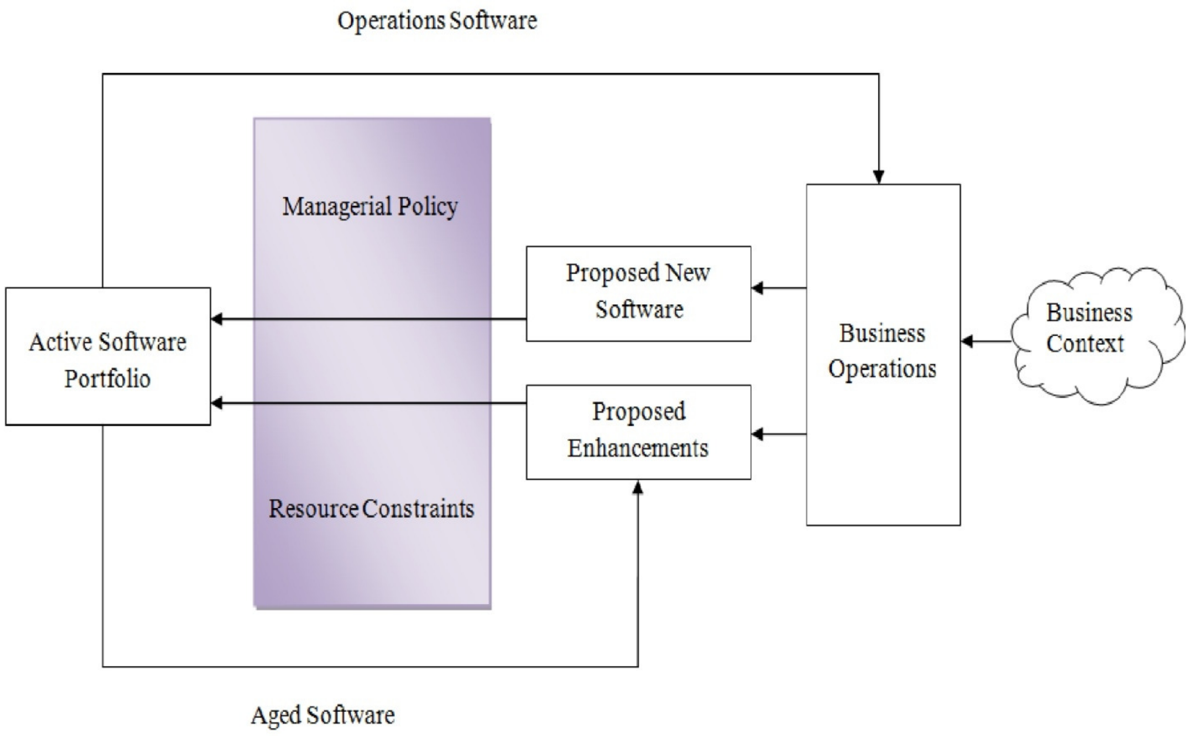


Figure 1.1: Information Systems manager's responsibilities

BUSINESS CONTEXT

A schematic diagram which shows the business context is given in figure 1.1.

Customer requests are the source of proposals received by the development team for processing. Business context indicates the area which generates requests for improvements in software. In a typical organization, the business context is customers interacting with the company.

BUSINESS OPERATIONS

Business operations are the activities that an organization undertakes to deliver products to end users. Efficient business operations help to reduce cost and improve customer satisfaction. Business operation analysis is initiated by organizations to evaluate and improve their process. Prototypes or models are built to understand the project better and recognize steps to improve efficiency.

MANAGERIAL POLICY AND RESOURCE CONSTRAINTS

After analysis of project requests, the projects are either chosen to be updated, replaced with new software, or ignored if they are not important enough. The manager has to make this decision whether to improve or replace software to support a product or service. The decision is crucial due to the availability of limited resources.

A project request is evaluated by a review team and is analyzed from different perspectives. A final proposal for the project is put together by the team. The proposed profile collects the following information:

- Size and complexity of the project
- Value or demand of the project in the current market status
- Resources required to complete the project

- The deadline for submission of a completed product

After consideration of this information, a solution is selected.

SOFTWARE PORTFOLIO

By a software portfolio, I mean all of the custom software developed by the organization. The existing portfolio is used for evaluating the importance of new or enhanced software. Computer scientists use *Program profiling* to analyze a *program* and generate a profile for the program behavior. In this thesis, *Portfolio profiling* refers to analyzing an organization's custom software for characteristics and quality. These depend on the value, Return on Investment (ROI), complexity, age, etc. of existing software. My simulator *iKriya* develops a quality profile from a simulated portfolio.

The portfolio helps in understanding consequences of a project before it is released to the developers. If a project needs to be implemented, it is added to the proposed portfolio enhancements.

1.2 *iKriya*

I have named my simulator *iKriya*. *Kriya* is a Sanskrit word which means a task or a job. Since my simulator performs virtual tasks, I have named it *iKriya.* The following are some of the details of *iKriya*.

CLASS DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. The class diagram of *iKriya* is illustrated below:

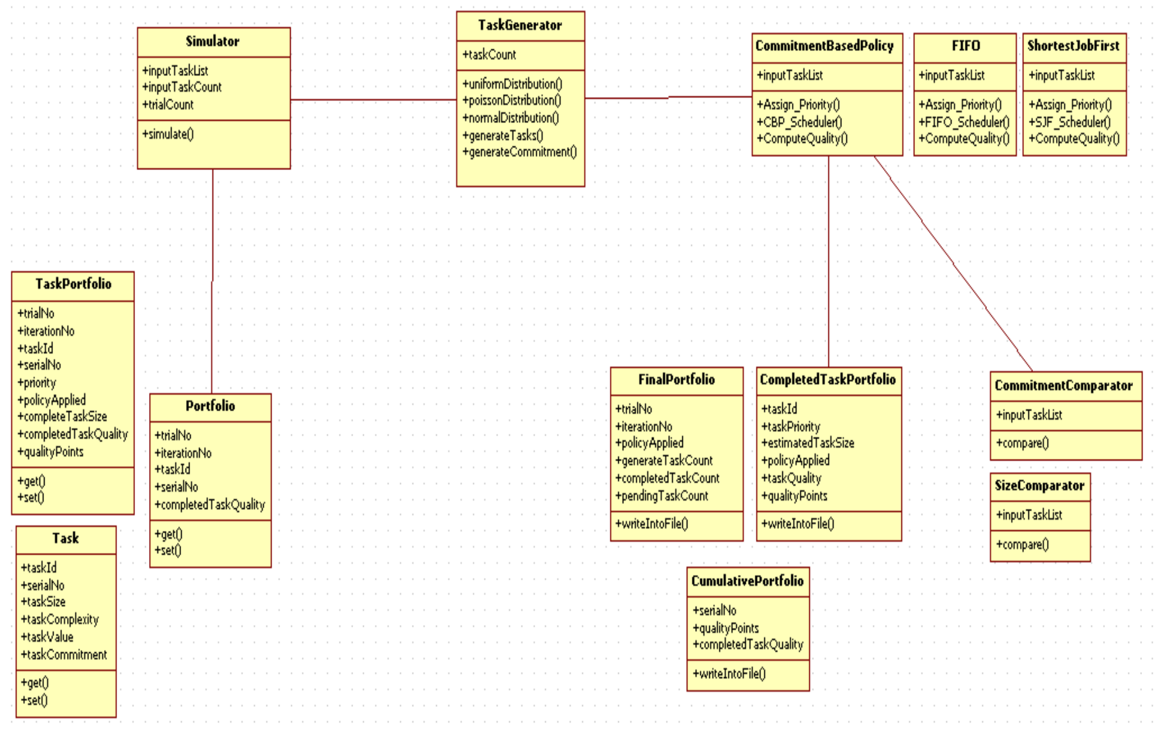


Figure 1.2: Class diagram of the Simulator

iKriya simulates the manager in taking IS development decisions based on input parameters and scheduling policies. It is used to simulate the problem under the assumption that *policies* are developed to determine when the software should be renewed or replaced. After deciding whether the software should be enhanced or replaced with new software, the resources needed for the chosen task are allocated.

The amount of resources to be allocated depends on the choices made. The four main responsibilities of *iKriya* to:

- Generate tasks
- Select a policy for task scheduling
- Do the tasks
- Generate a profile of the current portfolio

The simulator generates a profile in a virtual company over much iteration. The responsibility of *iKriya* is to analyze scheduling policies and simulate the development and maintenance tasks. Completed new tasks are added to the portfolio. The simulator analyzes newly completed tasks as well as the existing portfolio in order to generate a portfolio quality assessment. This analysis is based on following information about the components.

- Average Size
- Value
- Quality
- Estimated ROI
- Urgency

The final task portfolio consists of a numerical evaluation of the incoming task properties, the resources applied, and the quality of completed task. The simulator produces the following information to work with:

- Completed Task Size
- Completed Task Quality
- Remaining Task Queue, which consists of new tasks and incomplete tasks from previous iterations
- Estimated work units
- Quality of the completed tasks after that particular iteration
- Quality points of completed tasks
- Average completed task quality after finishing all iterations
- The information helps to understand the following details:
- Maximum utilization of the available resources
- The quality of the tasks that will be completed with the available resources
- Quality points if the tasks that were completed denote the quality in terms of size of the tasks along with the resources employed to complete them
- The list of tasks that were not chosen to be processed due to lack of resources
- Sometimes tasks are not chosen to be implemented because processing them would be unfeasible.

This information is also projected in the form of a graph for better understanding of the choices taken and their corresponding consequences.

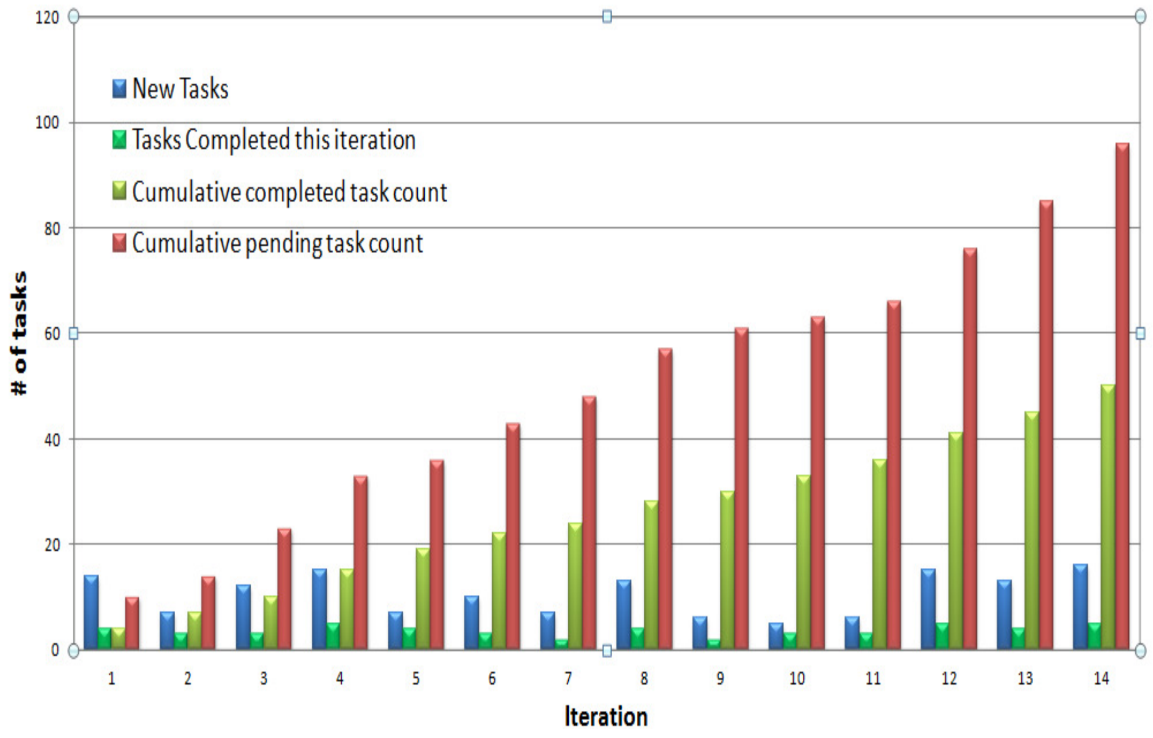


Figure 1.3: Sample portfolio output generated by the Simulator

In the above figure, all the tasks have been completed using Commitment Based Scheduling Policy. In this sample case, most of the generated tasks are completed. Hence, the backlog is at least three times lesser than the completed task count.

DOMAIN MODEL

The domain model is created in order to represent the vocabulary and key concepts of the problem domain. The domain model also identifies the relationships among all the entities within the scope of the problem domain, and commonly identifies their attributes. A domain model that encapsulates methods within the entities is more properly associated with object oriented models. The domain model provides a structural view of the domain that can be complemented by other dynamic views, such as use case models. The domain model of *iKriya* is depicted below.

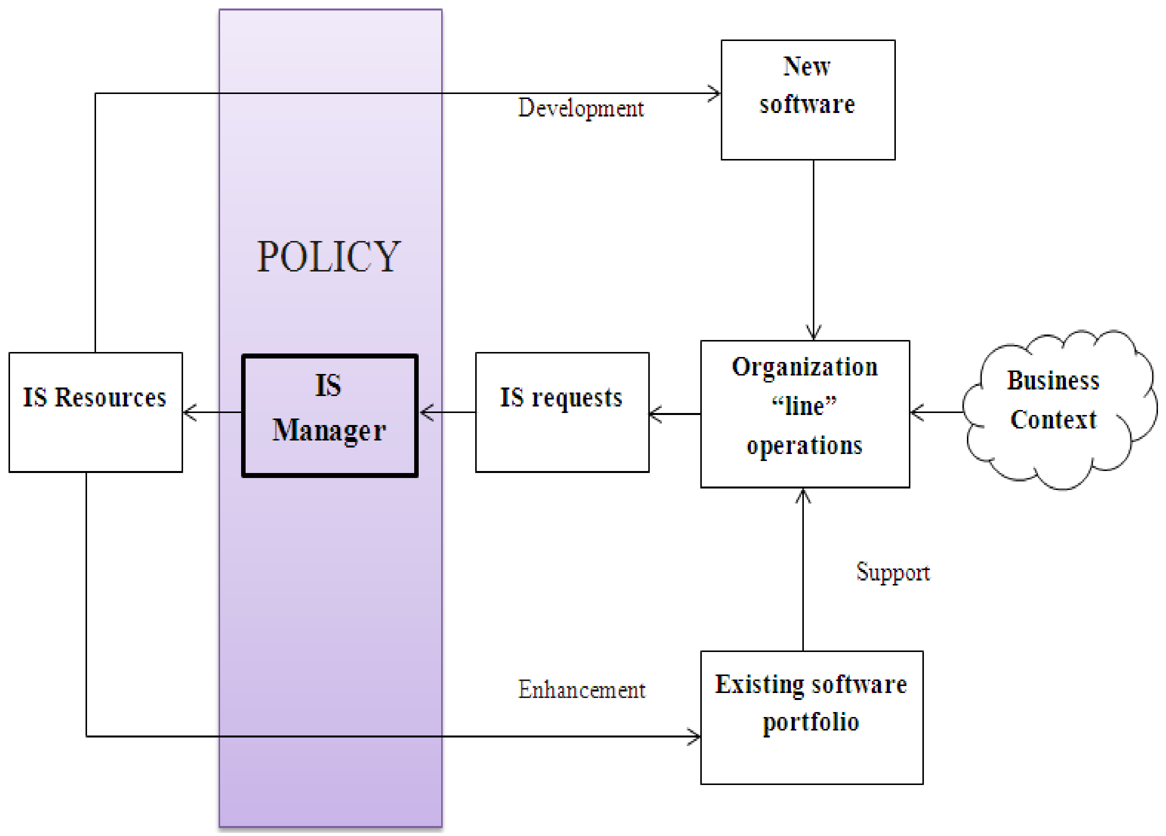


Figure 1.4: Domain model of the simulator

Any request arises from the business context. The requests are analyzed via organizational operations. All feasible requests are forwarded for implementation. The most suitable scheduling policy is applied for implementing the requests. The IS manager assigns the available resources to the chosen projects for implementation. Once a task is completed, it is further evaluated to check for all the requirements are met. The project that satisfies all the requirements is forwarded to the business context. Any project that does not satisfy the requirements is once again added to the project request for improvement.

Software can either be enhanced or developed from scratch and delivered to the end user. The testing team checks the operations of the end product and produces a carefully scrutinized report. The quality of the end product is given high importance during evaluation. The available resources and scheduling policy determine the quality of the finished project.

The simulator generates tasks with few significant parameters. The generated tasks imitate business context requests faced in organizations. The task requests are assigned a commitment value based on the current available resources and the corresponding task parameters. Tasks with high commitment are given high priority for completion. The final completed tasks are checked for quality. The quality of the tasks depends on the following parameters:

- Commitment towards the task
- Size of the task
- Available resources

ARCHITECTURE DIAGRAM

An architectural model (in software) is a rich and rigorous diagram, created using available standards, in which the primary concern is to illustrate a specific set of tradeoffs inherent in the structure and design of a system or ecosystem. Software architects use architectural models to communicate with others and seek peer feedback. An architectural model is an expression of a viewpoint in software architecture. The architecture diagram of *iKriya* is shown below.

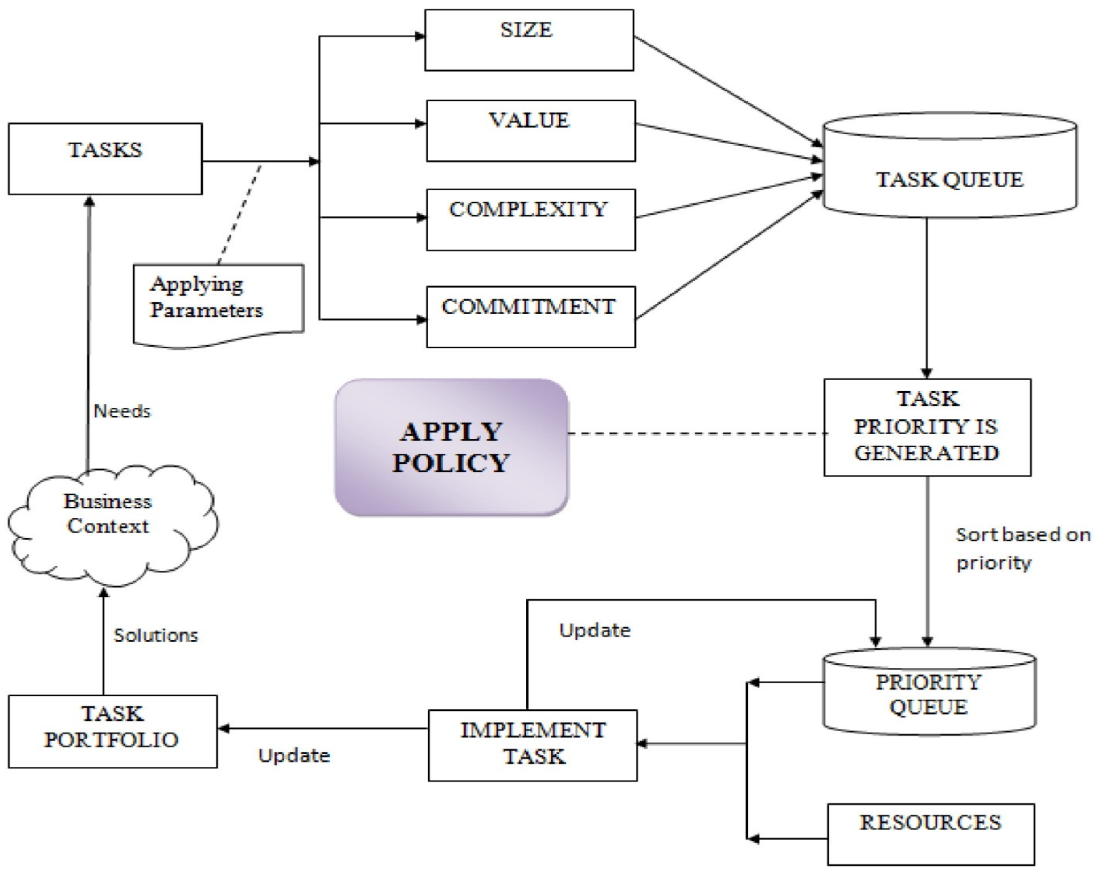


Figure 1.5: System architecture diagram

The architecture diagram shows the internal working of the simulator. The working of the simulator is described in the following steps:

1. Each new incoming task is identified using different parameters. Every task consists of four characteristics namely:

• SIZE	(Developer weeks. 1- 50)	
• COMPLEXITY	(1 = low	5 = high)
• VALUE	(1.0 = low	5.0 = high)
• COMMITMENT	(0.5 = low, 1.0, 1.5 = high)	

2. Now the tasks are added to the task queue for processing. The task queue consists of new requests as well as pending requests from the previous iterations.
3. Now a scheduling policy is chosen to be applied for the current task queue.
4. The scheduling policy determines the priority of the tasks to be completed and is added to the priority queue.
5. Tasks with higher priority are chosen for processing and sufficient resources are allocated for completion.
6. Once tasks are implemented, they are evaluated in order to generate a task portfolio.
7. The portfolio is further analyzed by the business context for any area of improvement. If all requirements are addressed, the tasks are sent back to the customers.

TASK PARAMETERS

A task is chosen for scheduling based on its primary parameters. Each manager might have different parameter list and criterion for selection. These parameters should be selected carefully since they play a vital role in completing tasks and producing a completed result. The factors essential for scheduling are:

1. Size of tasks
2. Complexity of tasks
3. Value of the tasks
4. Commitment

Using these parameters, the tasks are classified and scheduled to the developmental staff for performance/evaluation. The parameters are explained in detailed in the next section.

Using this data, the tasks are prioritized, and different scheduling algorithms are applied for execution. During simulation, the tasks and their parameters are generated in two ways namely:

1. Default
2. User defined

The default task generation method allows the simulator to generate random data. It does not need user to intervene. The task count, iteration count and parameter values are all determined by the simulator. In the user defined simulation, the user is allowed to make key choices like task count and value of each task.

1.3 MATHEMATICAL AND STATISTICAL BACKGROUND

These features must be analyzed by the simulator in order to generate the profile for the software (task). In the simulator, we initially consider three types of distributions for randomly generating the task cost, which include:

- Uniform Distribution
- Normal Distribution
- Poisson Distribution

The uniform distribution is defined as the distribution of a random variable in which each value has the same probability of occurrence. This is also known as rectangular or continuous distribution. The resulting graph resembles a rectangular object, where each element occurs at least once uniformly.

Normal distribution is considered once the uniform distribution is finished. Normal distribution is a continuous distribution for any random variable with equal and finite mean, median and mode. It is also called "Gaussian distribution," which produces a bell shaped curve. The normal distribution is generated using the following formula:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where

μ = Mean

σ^2 = Variance

e = 2.7182

Poisson distribution is a discrete random variable distribution, which gives the probability of a number of independent events occurring in fixed time. The Poisson distribution arises when you count a number of events across time or over an area. The Poisson distribution is defined using the formula:

$$f(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where

λ	=	Mean number of success in a given time period
k	=	Number of success we are interested in
e	=	2.7 182

2. PROJECT MANAGEMENT TOOLS

Project management software helps in project planning, organizing and managing available resources. Based on the level of software sophistication, many factors such as planning, decision making, scheduling and, resource allocation vary. There are several PC based project management tools and software available. These tools are being employed in several organizations with different business types. The several approaches of project management software are:

1. Desktop: The software is being implemented as a program, which runs on the desktop, and such tools are generally single user applications.
2. Web based: Any project management tool which is implemented as a web application falls into this category. Such applications are accessed using a web browser.
3. Personal: Personal project management tools are single user systems which are implemented and used at home.
4. Single user: Single user applications are programmed only for an individual user. These applications are useful in small organizations with very few people involved in project management.
5. Collaborative: Collaborative systems allow multiple users. This application also allows concurrency. For example, more than one user can modify the project plan at once. Finally, the changes are synchronized with other schedules when the application is committed.
6. Integrated: As its name suggests, integrated systems combine project management and project planning with other aspects in an organization.

7. Non-specialized tools: In some cases, software that is not project specific is also used to manage and schedule projects.

This section elaborates on some of the existing project management software and their applications.

a. Clarizen, Inc.

Clarizen is a project management tool that is deployed as a web based application. It was founded in 2005, and its first project management tool was released in 2007. Clarizen is a web based collaborative project management software which combines work management and team collaboration in a web application.

Clarizen supports the following functionalities:

- Project scheduling
- Collaborative planning
- Time tracking
- Budget management
- Budget tracking

b. LiquidPlanner

LiquidPlanner, Inc. is a Washington-based project management software founded in the year 2006. LiquidPlanner is a platform independent, online project management tool that was first released to the public in the year 2008. LiquidPlanner was developed using Ruby on Rails and a uses probability based scheduling engine.

The important features of this software are:

- Project Management
- Scheduling

- Collaboration
- Integrated time tracking
- Liquid Planner analytics: Reports are generated based on the past and current results. These reports help project managers and clients understand the problem better.

c. dotProject

dotProject is a multi-user and multi-language project management tool. It is a free and open source software developed by a community of volunteer programmers. dotProject was developed in the year 2001 to replace Microsoft Project. dotProject is an open source tool which uses a Gantt chart to represent the final report.

d. PlanBox

PlanBox is an agile project management tool released in the year 2009. It allows multiple users with different business functions to plan, collaborate and, deliver a completed project. Apart from project management, PlanBox also offers implementation of an agile life cycle into the organization.

PlanBox offers the following features:

- To – do lists
- Drag and drop prioritization
- Time tracking
- Reporting
- Feedback loop
- Bug tracking

e. TaskJuggler

TaskJuggler is a Linux or UNIX based software management tool which was initiated in the year 2001. It is free software, developed in C++ at first, but the later versions of this software were developed in Ruby. The TaskJuggler schedules tasks based on a heuristic algorithm and the final report is represented using Gantt charts.
The TaskJuggler supports the manager in all phases of software development, from requirement analysis to status tracking.

TaskJuggler uses a plain text project description which is written in hierarchical and declarative programming language. This makes the software complex and difficult to understand and is considered to be one of the major drawbacks of the tool.

f. Project KickStart

Project KickStart is a desktop based project management tool developed by Experience in Software Inc. in California. The first version of the software was released in the year 1992 for DOS based systems. It uses a wizard interface for project management. And the result is produced in the form of a Gantt chart which contains goals, tasks and assignments for each developer.

The project management wizard identifies the following information before project planning:

- Phases
- Goals
- Obstacles
- Personnel assignments

The major difference between the available tools and the proposed simulator is that none of these tools specify “which task should be done?” The project management tools available support a manager’s decision and help in carrying out task scheduling and report generation.

The proposed simulator has minimal manager intervention and determines which task should be scheduled for processing. Hence, the simulator takes up the entire responsibility of decision making, relieving the manager of this burden.

The manager sends a list of tasks to the simulator, which schedules tasks based on some scheduling criteria. Once the task is completed, the simulator returns the software portfolio which is verified by the manager. Based on the portfolio the manager decides whether to release the product to the line personnel or send it back to the simulator for re-scheduling. Thus the simulator helps the manager by making crucial decisions such as developing a new software or enhancing the existing software. The decision is made based on project properties by applying robust scheduling policy; this makes the scheduling precise.

3. ANALYSIS OF THE PROBLEM

The need for a smart management decision has led to the development of this thesis. There are several problems in an organization which influence the management decision. And it is the duty of the manager to take all issues into consideration before coming to a conclusion. The factors that influence the judgment are:

- Computers becoming cheaper
- More demand for limited resources which are quite expensive
- Software is given highest priority etc.

All of these issues directly affect the managerial decision. For example, if software becomes obsolete, it can either be upgraded or replaced. This decision depends on the current value of the software, complexity of the software which determines the resources required and urgency of releasing the new or upgraded product.

As time goes by, the value of software goes down; on the other hand, the software developer and resources become more expensive. In general, the price of software is lower when compared to the resources required to develop the software. Software tends to be cheaper due to growing competition among several organizations. Hence the available resources must be used optimally and to the fullest. The final quality of the software depends on the quality of the resources used during development.

The drop in price of old software shows that the value of software reduces at a large rate over time. And in some cases, developing new software is preferable over enhancing existing software. This crucial decision needs to be made by the manager considering the profit of selling new software over selling upgraded software.

The average salary of software developers among various organizations is depicted in the following graph. The salary of a software developer generally ranges from $40,000 to $80,000 annually. Since the costs of resources are growing significantly the manager must make decisions which utilize the maximum of the available resources.

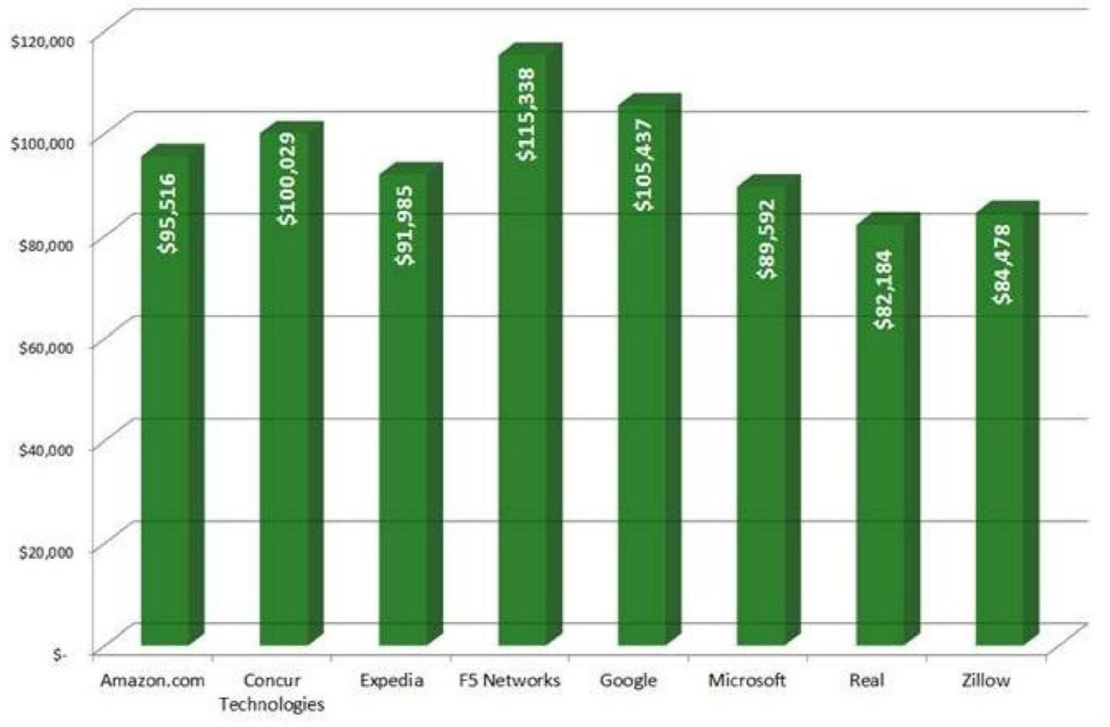


Figure 3.1: Software engineer’s average base salary

3.1 WHAT ARE THE PROBLEMS?

A summary of the problems specified above are illustrated in the diagram given below:

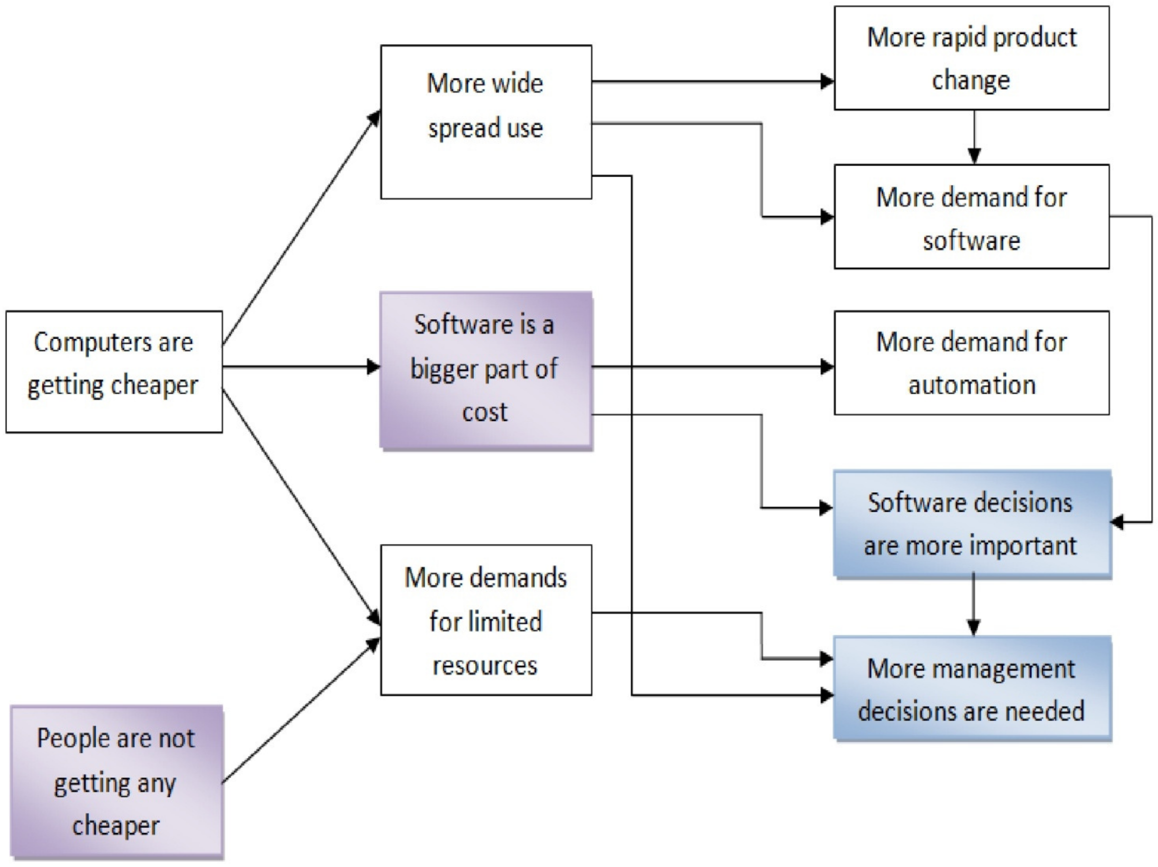

Computers are getting cheaper
People are not getting any cheaper
More wide spread use
Software is a bigger part of cost
More demands for limited resources
More rapid product change
More demand for software
More demand for automation
Software decisions are more important
More management decisions are needed

Figure 3.2: Need for efficient managerial decisions

The cost of computers is becoming lower due to competition and the availability of cheaper hardware. The compensation provided for any software developer tends to increase over time depending on the experience gained by the developer. Software is commonly used since it reduces the workload and never gets tired. Hence software is the biggest part of any product and it must be delivered to perfection.

The manager must consider the above factors and also the growing demand for limited resources before arriving to any conclusion. Any task should be completed with the available resources. According to the above graphs, it is evident that manpower is not always affordable. Hence employing new developers or buying new resources is not a feasible solution.

The demand for automation is also due to the high maintenance of software developers. Simulator or automation software tends to be more efficient and robust. This has also contributed to the growing demand for software. But the major issue with software is that it tends to age quickly, thereby making it outdated. High demand for software and intense competition among organizations has reduced the price of software.

In conclusion, the cost of hardware has gone down but the cost of resources has grown to a large extent. And the applications and demand for software is widespread. These compel the manager to develop software using available resources and the final outcome is expected to be efficient. The final decision of whether developing a new software or upgrading one already in existence lies in the hands of the manger, who considers all of the factors mentioned above and makes the final call.

3.2 HOW TO SIMPLFY THE PROBLEMS?

The aim of the simulator is to simplify the difficulties faced by the manager while determining whether new software should be developed or the available software should be improved. The simulator assists the manager by considering all essential factors as mentioned in the previous chapters. And several policies are applied to the incoming project requests. The simulator finally projects the results of applying several policies to the project requests. The output graphs display information regarding the resources available and required, as well as the quality at which tasks are completed. The simulator considers the project requests as tasks entering the task list for processing. Each incoming task has the following the features:

- Size
- Complexity
- Value

The user can choose to generate tasks with default values or user defined values and these parameters are used to evaluate the commitment and priority. Any task with high priority is chosen be completed using any of the following scheduling algorithms:

- Shortest Job First Algorithm
- First In First Out Algorithm
- Commitment Based Policy

Each task is recognized, evaluated and processed depending on its parameters and scheduling policy. The simulator finally projects the results which clearly describe the consequences of the applied policy in terms of resources utilized and quality of the completed task.

The output of the simulator consists of the following results:

a. List of completed task and pending tasks

b. Size of the completed and pending tasks

c. Quality of completed task

d. Quality points of completed tasks

e. Quality of completed tasks after the end of all iteration

These results help in determining the effects of applying different scheduling policies upon a set of incoming tasks. This helps the manager in choosing the tasks that need to be completed and the appropriate scheduling policy that needs to be applied.

4. ANALYSIS OF THE SIMULATOR

APPLICATION OF TOOLS IN THE THESIS

The purpose of defining and understanding these distributions is to implement them in our simulator. They are used to describe the characteristics of the tasks.

- Size: Size is one of the most important feature which uses uniform distribution. It can be the “hours of work” involved in developing the task and it must be considered within a particular range.
- Value: Each project is associated with a value, and it is generated using Gaussian distribution. Estimating the value is an essential task since it contributes majorly in determining whether to upgrade or abandon the software (task).
- Complexity: The complexity applies Poisson distribution, and it helps in estimating how difficult it would be to replace some of the tasks.
- Commitment: The commitment parameter is determined based on the other parameters mentioned above.

After determining the primary characteristics of a task, we must prioritize the task according to its corresponding features. All of the characteristics above play an important role in ranking the tasks.

Each task arrives with time spacing (probably Poisson) and later the highest priority task is selected and evaluated. It is essential to prioritize the tasks because the high priority task must be answered first, followed by the lower priority tasks. Resolving lower priority tasks while withholding higher priority tasks could result in some serious problems; hence prioritizing is a very important duty of the simulator.

After ordering the tasks according to their value, they must be examined by the simulator to determine the changes that need to be made to improve and advance the available task.

The tasks are prioritized by considering the three attributes along with the resources available. Poisson and normal distribution generation functions may produce raw values in decimal format, which are not suitable for further processing. Hence each value must be applied to a method to convert it into a suitable form for further processing.

The size of the decimal values is applied to a function which converts them into hour count. This thereby shows the number of hours required to complete a particular task, assuming that each employee works for eight hours per day.

Similarly, the complexity parameter is normally distributed which is applied to a method. This method converts the decimal value to corresponding integer values ranging only from zero to ten; where zero is the lowest and ten is the highest complexity a task can be assigned.

Value attribute is also similarly processed to produce only absolute values. Once the manipulations are completed, the parameters are applied to another function which provides the priority for each task. This method considers the newly updated values to produce an integer priority, where the highest priority task should be considered first, followed by the lower priority tasks.

There are many scheduling algorithms available to determine the order in which the tasks must be executed. Here we consider three algorithms:

- Priority Based Scheduling
- Shortest Job First Scheduling

- First In First Out

After understanding different scheduling techniques, the priority based scheduling is considered for further processing. The tasks are now sorted according to their priority and added to the “priority queue” for processing. The priority queue is an abstract data type which contains the highest priority element at the beginning of the queue. The queue is also implemented physically using linked list data structure within the system. When a new element is added to the queue, it is automatically sorted and inserted into its appropriate position. Priority queues are semi sorted lists, which contains the highest priority element at the beginning. Once an element is removed from the queue for processing, it is replaced by the next highest priority element and the same procedure continues.

Finally a portfolio is put together, which consists of information regarding the operations performed over all the iterations. The portfolio is a brief summary which describes the consequences of the choices made. It is the most essential part of the thesis, since it helps in deciding the usefulness of making a particular managerial decision.

4.1 DESCRIPTION OF THE SIMULATOR

The limited resources in the organization must be used optimally, and the purpose of this simulator is to replicate the situation that occurs in any organization. The primary components of the simulator are:

- Task generator
- Scheduler
- Task portfolio generator

The manager must make vital organizational decisions based on the incoming requests. The resulting decision helps in enhancing the quality of the tasks and proper utilization of the available resources. The mechanisms used in the simulator are critical and help in reproducing the environment that occurs in organizations. Each tool plays an essential role in the work of the simulator.

The first and most essential module of the simulator is the task generator, which helps in generating the incoming tasks.

TASK GENERATOR

The input in an organization is a list of project/software requests sent by the customers. The customer in this context would be any line personnel who request the manager to improve the quality of a particular product. The line personnel advertise and sell the product to an end user, hence seeking perfection. They verify each product from various perspectives and check for quality. If a particular product is not efficient then it is forwarded to the manager for reassessment and improvement. This process continues until the line personnel are satisfied with the resultant software.

The task generator in the simulator generates tasks with varying parameters and sends them to the scheduler for further processing. This section elaborates on the working of the task generator module. The task generator allows user defined or default values for each parameter of a particular task.

Initially the process begins by requesting the number of tasks that need to generate in a particular iteration. The mean value for the number of tasks is requested from the user. The user input is the mean value for the number of tasks to be generated which is applied to the Poisson distribution function.

Each task in this simulator is classified based on four parameters:

1. Size
2. Value
3. Complexity
4. Commitment

Size parameter specifies the approximate number of weeks required to complete a particular project. It varies from 1 to 50 developer weeks and is uniformly distributed. The task generator allows the user defined values or default values.

The default value that is assigned for the size parameter is 50 and the user specified value can vary from 1 to 50. These values are specified using a slider which has a maximum value 50 and minimum value 1. The input taken for the size parameter is now used in the uniform distribution function as the mean value. And thus the incoming task size varies uniformly within a particular range.

Value parameter is used to determine the significance of a particular task and it is a double field which varies from 1.0 to 5.0. Gaussian distribution is applied for this parameter, which takes 4.0 as the default mean for the distribution. The user is also allowed to enter any mean varying from 1.0 to 5.0 via the slider.

Complexity specifies the level of difficulty a task in a particular task and it is a double value which ranges from 1.0 to 5.0. The default task mean value assigned by the simulator is 3.0. Any user is allowed to enter his/her own task mean value within the particular range and this input is used for generating task complexity which is Poisson distributed. When the tasks are applied to the scheduler, the complexity is equated to a new range based on the original complexity.

The mapping is done as given below,

Complexity	**Numerical Equivalent**	**Initial Complexity**
EASY	0.50	1
	0.75	2
MEDIUM	1.00	3
	1.25	4
HARD	1.50	5

Table 1: Complexity mapping

Commitment parameter is evaluated based on the other values specified above. Size, complexity, and value are used to derive the commitment parameter for each task. This parameter is quite crucial since it specifies the level of commitment the manager and developer team have over a particular task.

The commitment is a double field varying from 0.5 to 1.5. Here 0.5 is the least and 1.5 is the highest commitment a manager and team can have towards a project or task. Tasks with commitment value of 1.5 are given highest priority and are scheduled for improvement.

The commitment for a particular task is given based on these rules:

1. If the value is greater than 3 or complexity is less than 3 and the size below 25 weeks then the particular task is assigned highest commitment value of 1.5.
2. In the second case, if the size, complexity and value all have mediocre importance then the complexity is assigned to 1.0.
3. If the value is less than 3 and complexity is above 3 or if the size is above 40 then those tasks are given minimal commitment of 0.5.

The task generator module produces tasks with varying size, value, and complexity based on the uniform, normal, and Poisson distributions respectively. These features collectively help in recreating situations that occurs in any organization.

The structure of the tasks generated is illustrated using graphs by the simulator. The profile and the difficulty distribution for each task in terms of size, complexity and value are projected in the graph. And if the number of task generated exceeds 20, then a sample graph is displayed considering every fifth task from the queue.

WORKING OF THE TASK GENERATOR

The task generator begins by retrieving the mean value for the number of task that needs to be generated in a particular iteration. Followed by it, the user can either choose to give his/her own values or default value. This choice is made by selecting either of the two available radio buttons.

1. Default task generator: The following are the specifications of the default task generator:

- Size with a mean of 30
- Complexity with mean of 3
- Value with mean of 2

2. User defined task generator: The end user can pick any value within the following range:

- Size : 1 to 50 weeks
- Complexity : 1 to 5
- Value : 1.0 to 5.0

SCHEDULER

The scheduler aims at performing each incoming task. The way in which tasks are chosen depends on the scheduling policy applied. An appropriate policy must be chosen for maximum utilization of the available resources. The policy chosen is majorly influenced by the following factors:

1. Resources available,
2. Parameters of the incoming tasks

As described earlier, the simulator implements three types of scheduling algorithms to tackle the incoming tasks. The workings of the scheduling algorithms are discussed in this section.

1. SHORTEST JOB FIRST ALGORITHM

As its name suggests, this scheduling algorithm chooses the tasks with the smallest size for completion. The simulator sorts the available tasks according to their size and picks the task with smallest size first. The resources necessary for the chosen task is evaluated using the following formula:

$$Estimated\ Work\ Units = Task\ Size * Task\ Complexity * Commitment$$

In this formula, complexity of task indicates the numerical equivalent of the task complexity. Whenever a new task is scheduled for completion, adequate resources necessary to complete the task is allocated. The next task waiting in the queue to be executed is not chosen until the necessary resources are available. As tasks are chosen to be completed, the available resources count is decremented. Hence, this process continues until the available resource dries out.

2. COMMITMENT BASED POLICY

The second scheduling policy available in the simulator is the Commitment Based Policy (CBP), which solely depends on the commitment attribute. The commitment is fourth and the most essential attribute of a task. An organization can have three levels of commitment towards a task and it is corresponding numerical values are given below:

Commitment	**Numerical Equivalent**	**Priority Equivalent**
High	1.5	3
Normal	1.0	2
Low	0.5	1

Table 2: Commitment range mapping

3. FIRST IN FIRST OUT SCHEDULING

This is a simple yet efficient scheduling technique. Here the tasks are scheduled in the order in which they occur. The task which arrives first is scheduled first followed by the other tasks. In this technique higher priority tasks may not be solved first due to the order of arrival.

The simulator by default distributes all three commitment values. That is, the tasks generated contain high, low, and normal commitment values in the following range:

- 20% of the tasks are given high commitment
- 40% of the tasks are given normal commitment and
- 40% of the tasks are given low commitment

Once the commitment values are assigned, the tasks are now sorted according to priority. The priority depends on the commitment of the organization towards the task. The priority values are also described in the table above. The task with highest priority is chosen to be scheduled by the simulator.

In the commitment based policy, all the tasks which are quite critical are chosen for completion. Unlike the shortest job first policy, which processes only the jobs which are small, the commitment based policy gives importance to critical tasks. Hence tasks with higher significance do not have to wait until smaller tasks of lower significance are completed.

The sample results gathered after several iterations have shown that the commitment based policy performs much better than the shortest job first policy. These results were concluded based on several runs with the default task generator values. The values generated were scheduled using both scheduling policies and the results were plotted to get to this conclusion.

PORTFOLIO GENERATOR

The last and the most crucial process of the simulation is generating a portfolio. The portfolio helps in understanding the working of all the above mentioned components. It is a collective report of task processing. The portfolio generated by the simulator helps in understanding the working and results of the preferences made during simulation.

Every task is identified using the following basic information:

- TaskId: It is an identifier which consists of trial number, iteration number, and, task number. This information is separated using the dot symbol (.)
- TaskSerialNo. : It is a five digit number, and it helps in uniquely identifying each task.
- TaskPriority: It is the priority value assigned for a task. The priority is given a value of 1, 2, or 3.
- TaskParameters: The task size, value, and complexity information are also displayed in the portfolio.
- PolicyApplied: This field specifies the type of the scheduling policy applied for the particular task list.
- TaskGeneratedOn: The iteration on which a particular task was generated is specified in this field
- TaskCompletedOn: The iteration on which a particular task is completed is specified in this field. Not all tasks are completed on the iteration in which they were created. This information is essential since it does influence the quality of the tasks after completion.

The portfolio produces numerical results which help in comprehending the process of task scheduling. These values play a vital role in making the final managerial decision.

- Estimated Work Units: This is one of the most important fields in the portfolio. The estimated work unit specifies the work units or resources required to finish a particular task. In this simulation, it is assumed to be measured in terms of weeks.
- Completed Task Quality: Each task is expected to be completed at a particular quality. And this parameter specifies the quality at which a task will be processed. It directly depends on the priority assigned to a task. The priority of a task is based on several external factors apart from the internal factors present in the task. The completed task priority and their corresponding task quality is specified in the following table:

Task Priority	**Task Quality**
3	1.5
2	1.0
1	0.5

Table 3: Mapping Task Quality based on Task Priority

The assessed quality of completed tasks tends to go down as time goes by. The quality of any completed task goes down by 2% per iteration after the task is completed. A final task quality list is displayed with the modified task quality.

A mean task quality is also produced, to better describe the quality of the completed tasks based on the simulation policy chosen. If the mean value is between 1 and 1.5, then all the tasks are completed at reasonable quality. But if the mean value of the tasks completed is less than 1.0, then some tasks are completed at poor quality.

- Quality Points: The third and most essential value is the quality points. This parameter specifies the quality of the completed software in terms of the resources allocated for completion.

Using all this information the quality of the completed task is determined. It depends greatly on the scheduling policy. The manager can now choose the most feasible policy among the three options. *iKriya* therefore helps by acting as a trial run before simulating the actual decisions. Therefore the risk of loss or wastage of resources due to uncertainty is diminished.

4.2 DESIGN DESCRIPTION

In software engineering theory, the individuals who carry out roles are called *actors*. The actors considered here are:

1. Line personnel
2. Manager
3. System development staff

In general, the line personnel are the people who have direct interaction with the day to day operations of the organization. It is their job to make sure that the responsibilities delegated to other personnel are properly being performed. They carry out the basic functions of the organization, and the primary activities of business such as production, sales, or marketing, etc. It is the responsibility of the line personnel to attain basic objectives and ensure proper implementation of any project. They match the final end product to the requirements specified by the customer and try to market the completed product.

The line personnel acts as a customer of the IS Department in this simulation. It is the duty of the line staff to bring in projects in the form of business needs which are incomplete or should be evaluated.

The projects are selected for execution or completion based on different policies proposed by the simulator. A final completed project is produced as the output along with a daily report profile. This final report is used by the line staff for improving the project.

The Manager is the next most important actor in the system. It is the manager's responsibility to schedule the incoming projects for execution. Each project has several parameters out of which some primary parameters are derived for prioritizing projects.

The Manager simulates the highest authority here, and is responsible for proper execution and the outcome of the simulation. The manager decides how the tasks should be prioritized and the parameters that must be considered for prioritization. As discussed earlier the four basic parameters that are taken into consideration are value, size, complexity and ROI.

In a real context, there may be other factors that raise the priority of any task such as:

1. Prize offered for completion of a project
2. Authority of the calling line personnel
3. “Age” of a waiting project
4. Urgency of the particular project etc.

These factors are not considered in this simulation.

All the factors play an important role in scheduling tasks. The age of a project in the queue should be considered in order to avoid starvation of tasks. Though the basic parameters play a significant role in allocating projects, there are other subtle factors that must be taken into consideration during scheduling and it make a major difference. Hence the manager takes the vital decision of selecting the appropriate parameters for executing tasks.

The managers should be skillful, knowledgeable, and experienced enough to analyze tasks from different perspectives before scheduling them to the system development staff.

The chosen tasks are now investigated by the System Development Staff (Programmers) for detailed analysis. This helps in providing a report which is used for further enhancement. The system development staff should be technically sound and experienced in testing finished projects. The report generated by the system development staff is again verified by the manager for correctness before sending it to the line personnel.

Apart from the physical participants, there are some abstract participants which are critical in this simulation. These abstract participants are the input or outcome of the simulation which are utilized by the physical participants.

TASK PROCESSING

After populating elements into the priority queue, any task may be processed depending on the resources available. In order to process each task, remove the task from the list and evaluate the availability of the assets. If there are enough of resources, then process the tasks and remove it from the queue. Every completed task is then added to the "done tasks" list. After all this, then the resources available must be checked again to consider the next task. This process continues until all the tasks are processed or else resources available for the iteration are exhausted.

The priority queue data structure fits this need perfectly, because each time it places the highest priority task at the top of the queue. It is not time consuming like sorting since it does not need to completely sort the queue to determine the largest value among a set of values.

We have to begin by considering the available resources at the start of a day and proceed to the end of the day.

Initially iKriya will check the…

- resources available and
- number of tasks available in the queue (task backlog).

Then it will pick the highest priority task and assign resources, if available. If not, it will go to the next task that can be processed with the available resources. Each time the completed tasks are added to another list and removed from the current queue. At the end of the iteration, update the prior tasks by determining their states.

These states will be:

1. Finished
2. Started but incomplete
3. Not yet started

Now iKriya will update the available resources depending on the tasks assigned. New tasks are also generated and added to the priority queue based on their priority. This process continues the next iteration using the updated values.

Software is prioritized based on the three major factors, namely, size, complexity and value. But these factors are considered in different levels at different situations. For example, some organizations may consider the value of the product more than the other two issues. The feature which should be measured more than the other is determined by several other external factors.

Whenever a customer specifies his requirements to any organization, several external factors are taken into consideration apart from the basic three factors (size, value and complexity). To further support my argument I would like to use some of the information collected from several organizations located in India.

The decision to develop a project is taken by the entire organization which also includes the quality assurance team. The task of the quality assurance team is to consider every new project and analyze it in different perspectives.

Higher significance is given to the key customers, who share the business profit along with the organization since they ensure profit for the organization. For the top customers in the market, the requirements are not given much importance. The organization tries to fulfill the needs of priority customer at any cost. The revenue gained from the projects provided by such customers is very high and secure. Once the project is analyzed, the requirements are gathered and the duties are deployed to the developers for implementation.

Projects from smaller players (customers) and new customers is also accepted after carefully screening it from different viewpoints such as:

- Company reputation
- Financial strength
- Competitors
- The management technique followed by the organization
- Ability to return the product on time
- Technical feasibility
- Commercial feasibility
- Business volume etc.

Each new project's present and future standing in the market is carefully analyzed by the QA team before taking the final decision. The above criterion help an organization in increasing their ranking among competitors and also help in providing leverage to an organization in the global level. Hence it is clear that apart from size, complexity and value there are other factors which contribute in deciding whether to undertake a software project or not. A survey was taken from 50 executives belonging to different organizations and different departments such as:

- Development
- Testing
- Marketing
- Quality Assurance

Executives from all the above departments play a vital role in project selection and they help in providing perspective and gathering information from different viewpoints which help in taking a through and collective decision.

The most important factors which influence this decision are:

1. Size
2. Complexity
3. Current value in the market
4. Return on investment (ROI)

ROI is defined as a performance measure used to evaluate the efficiency of an investment or to compare the efficiency of a number of different investments. To calculate ROI, the benefit (return) of an investment is divided by the cost of the investment; the result is expressed as a percentage or a ratio.

ASSISGNING PRIORITY

After defining each parameter and determining their corresponding distributions, the next step is to consider these parameters for the process of prioritization. This is a significant procedure in processing the selected software project. Based on the assigned priority, the projects are developed by the executives, hence it is vital. The basic four factors must be carefully analyzed and an infallible formula must be produced. This formula must be applicable on all conditions and must produce an accurate result.

A priority value must be assigned for each project, and the project with the highest priority must be attended prior to the projects with lower priority. And this is done in the following steps:

1. Considering each parameter
2. Manipulating each parameter and assigning a priority value for each software project
3. Adding each software project into a priority queue and sorting it according to the priority. That is, the task with highest priority is performed first followed by the task with the second highest priority and so on.
4. After performing a particular task, it is removed from the priority queue and added to the 'Completed task' list.
5. If the organization run out of time or resources, then the incomplete task are again added to the priority queue and assigned new priority based on the current status of the particular task.
6. At the end of each day, a report must be produced.

The report contains the following information:

- Completed tasks which have been added to the software portfolio.
- Incomplete tasks and their status of completion.
- Resources used.
- Time taken for completion.
- New tasks which need to be implemented.
- Resources available etc.

7. Based on the report, the same procedure is followed the next iteration too.

The report is generated at the end of each iteration is very useful in planning the requirements for the next cycle. The report helps in predicting the flow of the task and keeping the tasks on track. This also helps in completing the tasks within the deadline without causing any delays. The resources required for the next iteration are also provided by the report, which helps in defining the requirements needed for the future. Proper planning will help in completing the tasks within the given due date and with perfection, and avoids unmet needs.

POLICIES

A generalized policy is formulated for determining the priority of each task considering the four parameters and producing a value which determines the task priorities. The task with the highest priority value is implemented first.

- Value and ROI are the two most important factors. It would be appropriate to sum those values in the formula.
- Complexity cannot be considered as a positive parameter because, if the complexity is high, then the priority decreases, because more resources are

needed to complete the task. Yet it is also sometimes essential to perform a task with higher complexity. Hence, this parameter is subtracted from the formula, which reduces but does not eliminate the priority.

- The last parameter is Size. It makes a large difference in determining the priority since a large task may crowd out other needed tasks. A task with higher size is not preferred and has a great influence on the priority. Hence, this parameter is divided into the other parameters in the priority formula.
- The next most important factor is resources**.** It is necessary to have sufficient assets for any project. This parameter is not included in the formula because it does not influence the priority of the tasks. Instead, according to the tasks and their corresponding requirements, the resources must be fetched for completion. Thus it is clear that resources depend on the tasks and not vice versa.

  The task with the highest numeric value denotes higher priority and is implemented first and followed by the tasks with lower numeric value.

Altogether, the formula with all the factors included is given below:

$$Policy = \frac{Value + ROI - Complexity}{Size}$$

Numerical distributions are applied to the primary parameters (value, ROI, complexity and size), such as Normal or Poisson Distributions, and a final output is obtained. Those values are applied in the above formula to determine the level of importance of each task.

The general formula is applied for each task and they are all prioritized according to their output value. Based on the output value, the tasks are added to the priority queue which is updated frequently.

Whenever a new task arrives, its priority is determined using the formula and it added to the queue according to the priority. When a task is completed partially and if it is postponed to the next cycle, the priority must be determined before restarting it again. In case of incomplete tasks the priority value tends to change and the new value must be considered, overriding the previous value.

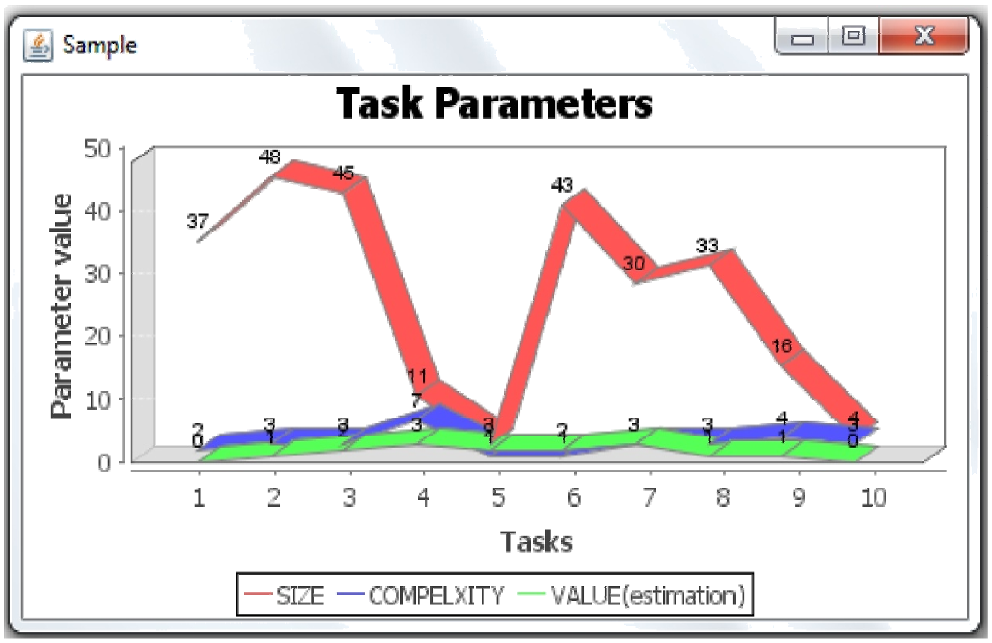


Figure 4.1: Incoming task stream

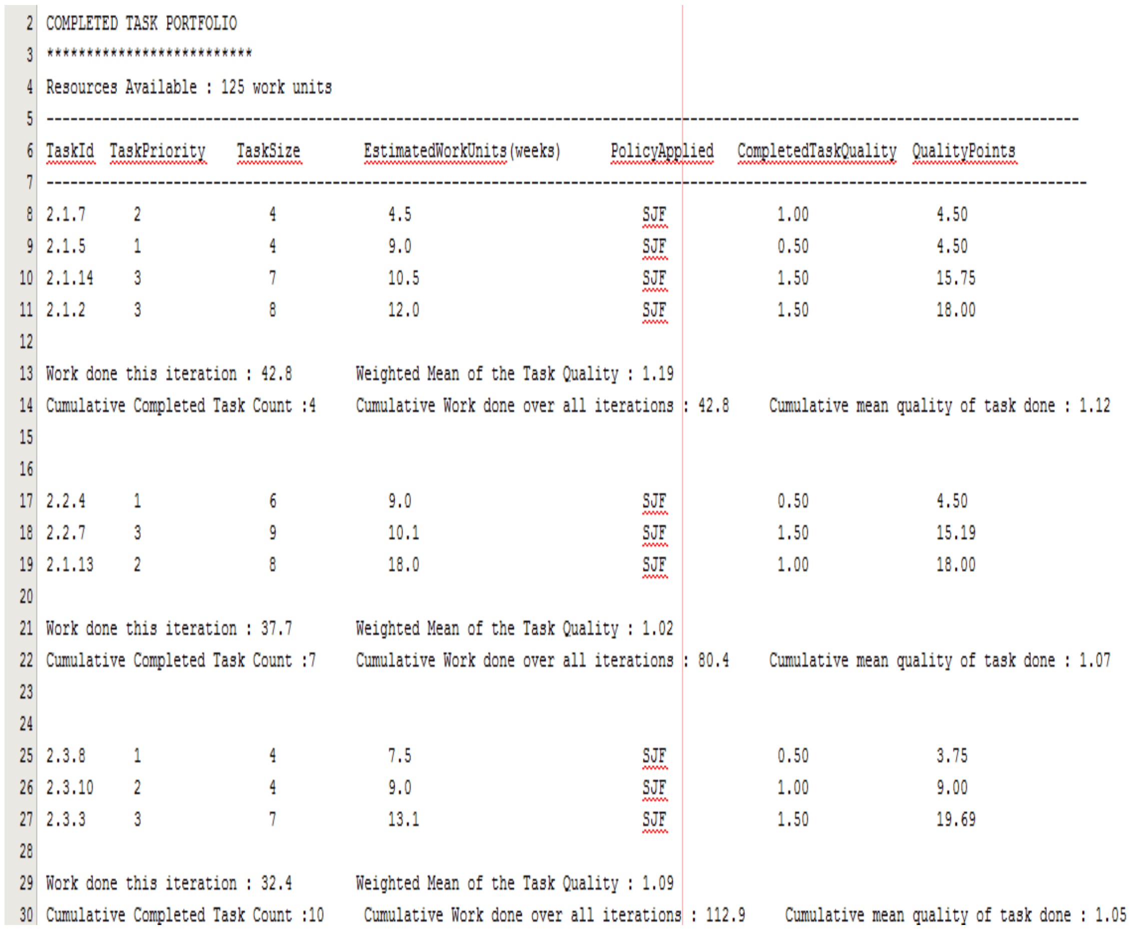
This line chart represents a sample of the tasks generated, here ten tasks and their corresponding parameter data is shown. The red line indicates the size of each task, blue line denotes the complexity and the green line denotes the value of each task. The fourth parameter, commitment is determined based on these value according to the rules mentioned above. The commitment value is now used to determine whether the task should be scheduled for completion or not.

After applying scheduling algorithms, the policies are also applied to the same set of data and comparison is done to determine which algorithm is the best.

4.3 *iKriya* RESULTS

PORTFOLIO SAMPLE

The following is a sample screen shot of the information provided in the portfolio and all these data prove to be very useful in understanding the significance of a scheduling algorithm.



```
COMPLETED TASK PORTFOLIO
**************************
Resources Available : 125 work units
-------------------------------------------------------------------------------------------------------------------
TaskId  TaskPriority   TaskSize       EstimatedWorkUnits(weeks)     PolicyApplied   CompletedTaskQuality  QualityPoints
--------------------------------------------------------------------------------------------------------------------
2.1.7     2              4              4.5                             SJF             1.00               4.50
2.1.5     1              4              9.0                             SJF             0.50               4.50
2.1.14    3              7              10.5                            SJF             1.50               15.75
2.1.2     3              8              12.0                            SJF             1.50               18.00

Work done this iteration : 42.8        Weighted Mean of the Task Quality : 1.19
Cumulative Completed Task Count :4     Cumulative Work done over all iterations : 42.8     Cumulative mean quality of task done : 1.12


2.2.4     1              6              9.0                             SJF             0.50               4.50
2.2.7     3              9              10.1                            SJF             1.50               15.19
2.1.13    2              8              18.0                            SJF             1.00               18.00

Work done this iteration : 37.7        Weighted Mean of the Task Quality : 1.02
Cumulative Completed Task Count :7     Cumulative Work done over all iterations : 80.4     Cumulative mean quality of task done : 1.07


2.3.8     1              4              7.5                             SJF             0.50               3.75
2.3.10    2              4              9.0                             SJF             1.00               9.00
2.3.3     3              7              13.1                            SJF             1.50               19.69

Work done this iteration : 32.4        Weighted Mean of the Task Quality : 1.09
Cumulative Completed Task Count :10     Cumulative Work done over all iterations : 112.9     Cumulative mean quality of task done : 1.05
```

Figure 4.2: Data in portfolio after applying Shortest Job First scheduling policy

FIRST IN FIRST OUT – Not enough resources

The results after applying FIFO scheduling algorithm is shown below. From the graph it is evident that resources available were not sufficient. Hence the backlog of tasks is increasing significantly.

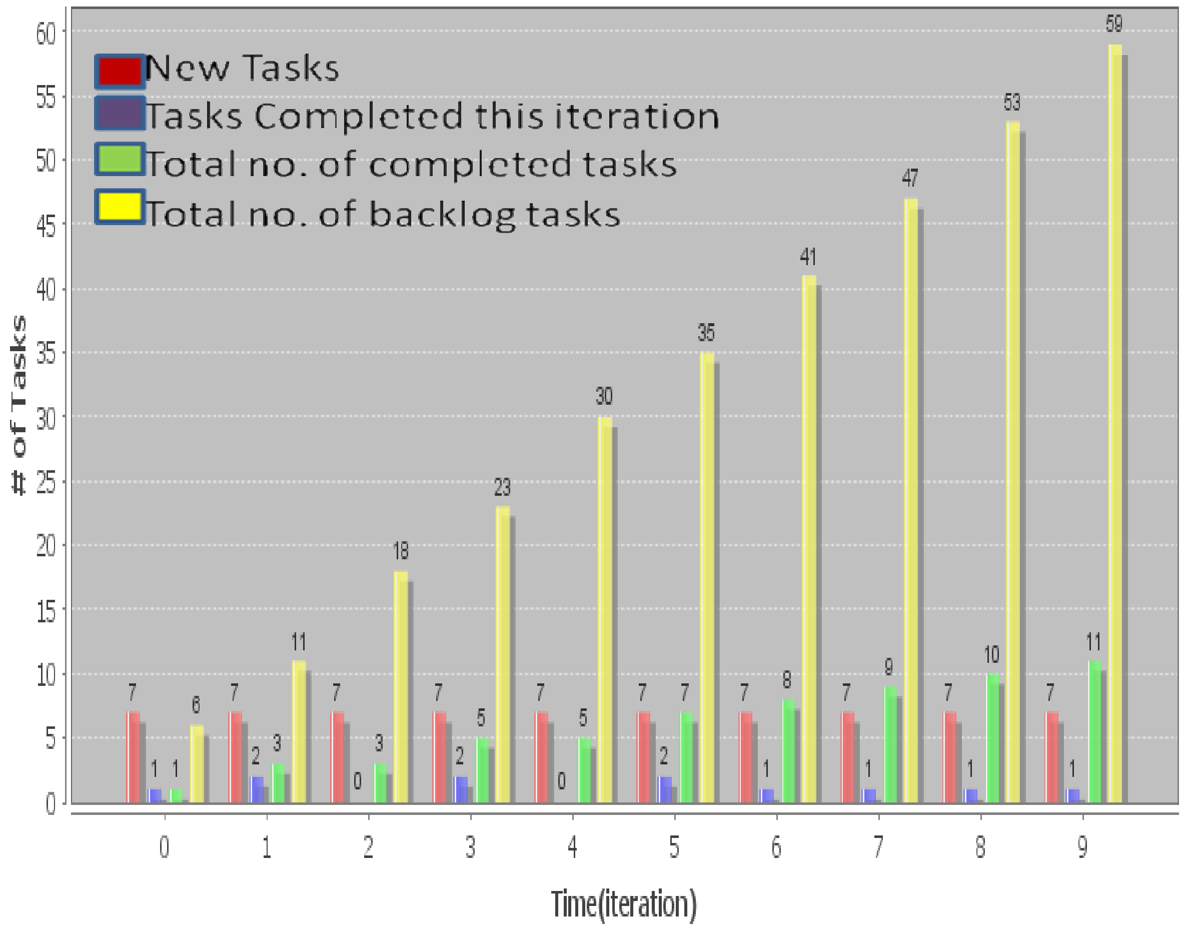


Figure 4.3: FIFO scheduling algorithm with only 10 resources

FIRST IN FIRST OUT – Not “quite” enough resources

The following graph shows the end result after applying FIFO scheduling algorithm. In this trial more number of resources is allocated unlike the previous trail. The result is much better that the former iteration. But the backlog still seems to be increasing.

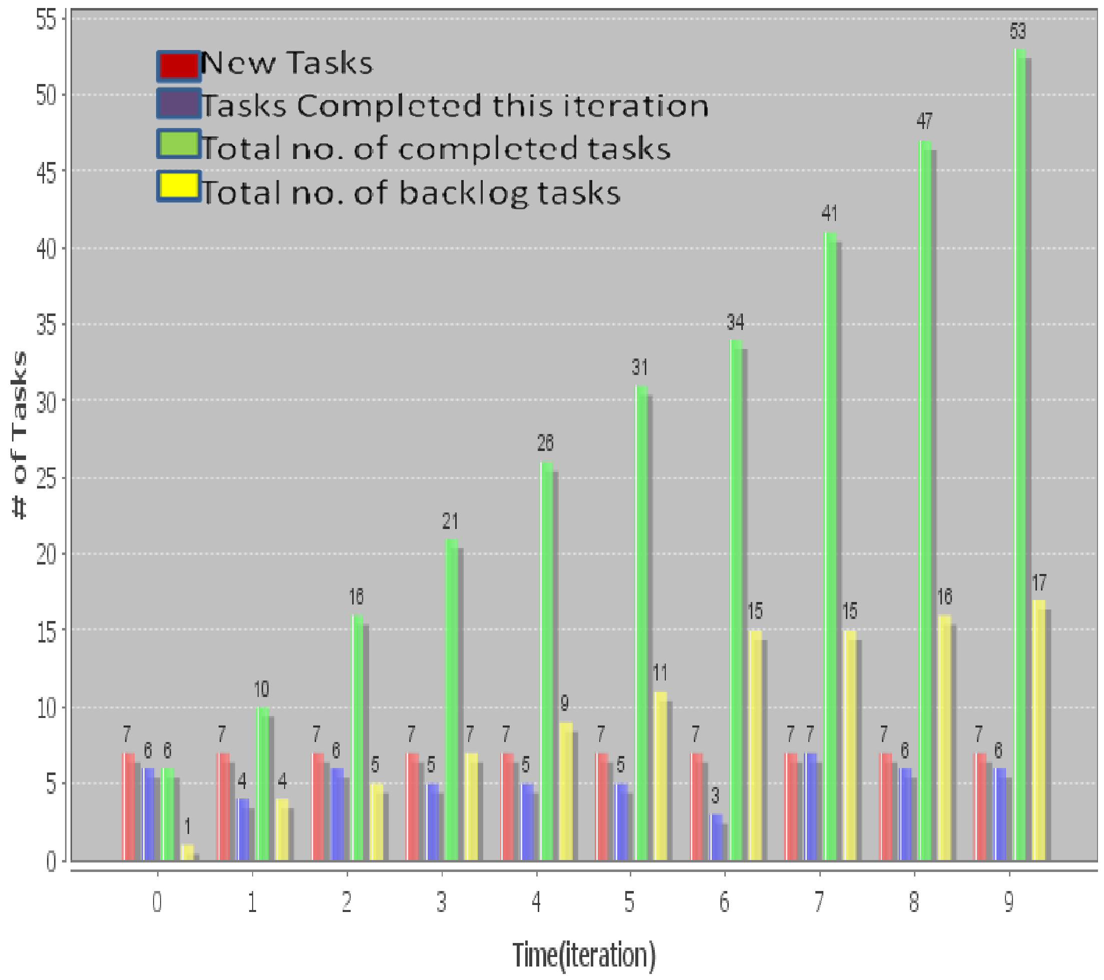


Figure 4.4: FIFO scheduling with 30 resources

COMMITMENT BASED POLICY

Here the commitment based policy was applied for the same input tasks with fewer resources allocated. The end result proved to be quite superior to the other two trials. Though the backlog still persists, the number of completed tasks has also increased.

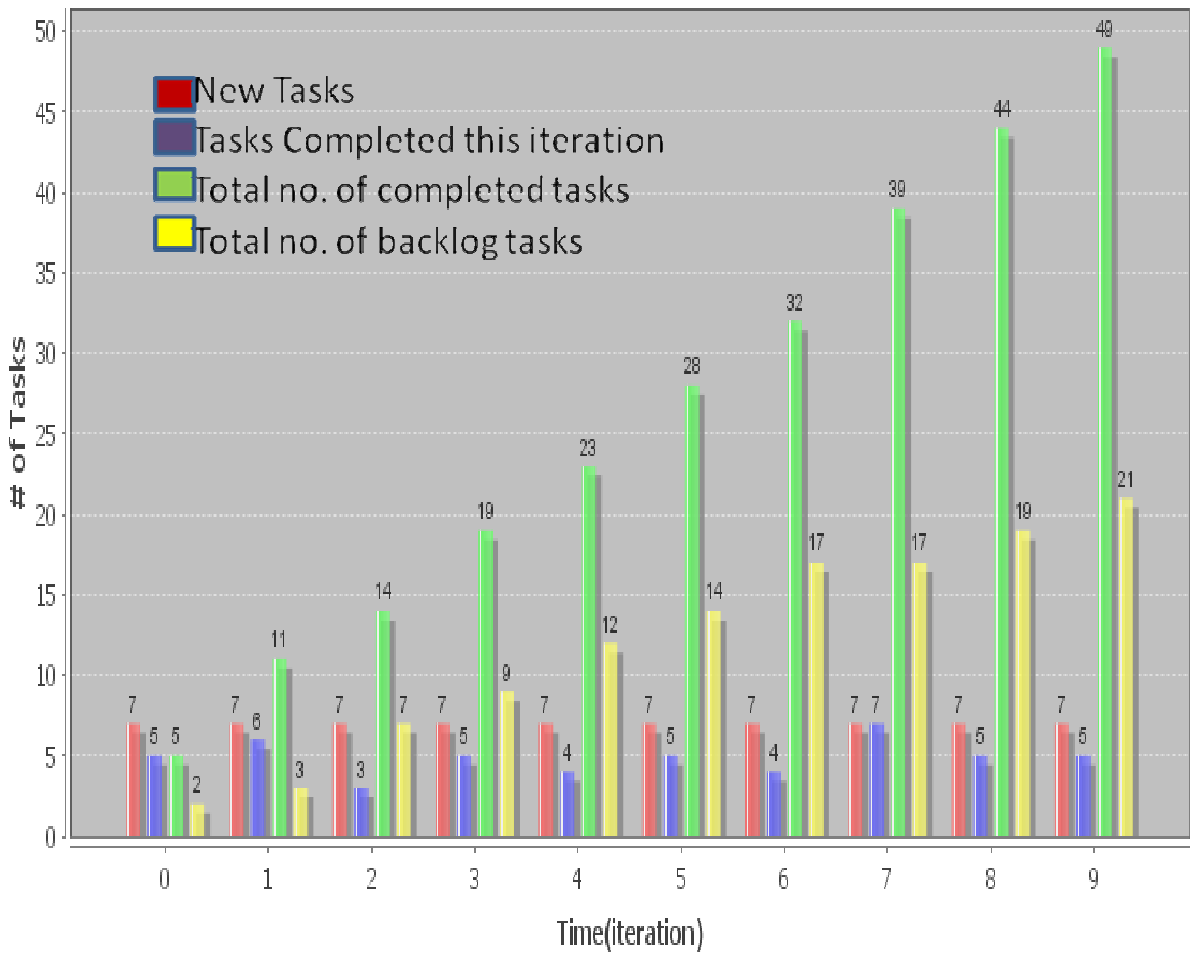


Figure 4.5: Commitment Based Scheduling Policy with 20 resources

RESULTS DERIVED FROM THE PORTFOLIO

A series of graphs are generated from the portfolio. The graphs show distribution of important parameters such as:

- Quality
- Work done

Plotting these parameters help to understand the effects of the scheduling policy on the incoming task requests.

i. SHORTEST JOB FIRST (SJF) SCHEDULING

The following graph shows the work completed after SJF scheduling algorithm was applied. The quality at which the tasks were completed is also depicted in the graph. In SJF scheduling, the work done is restricted to a maximum of 30 work units. But the tasks are completed at reasonably good quality (1 to 1.2).

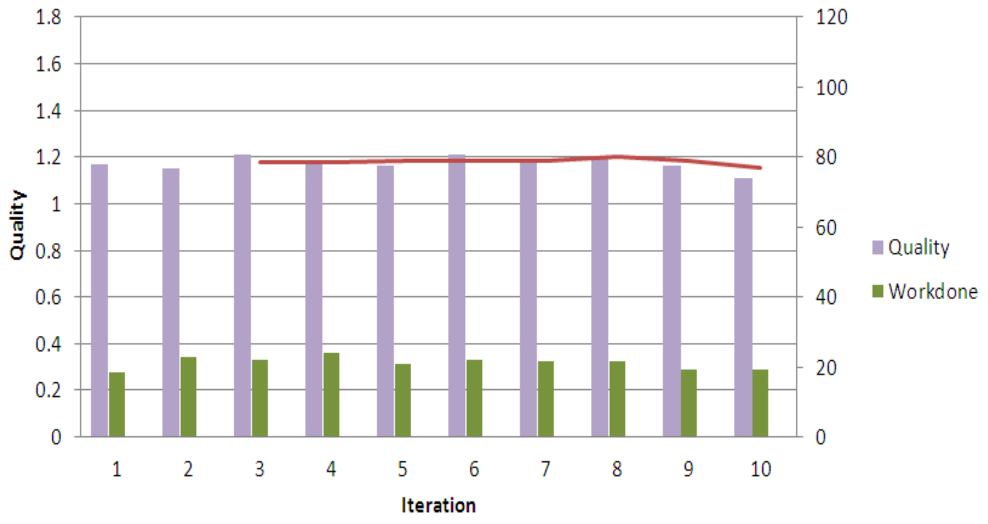


Figure 4.6: Quality and work done after SJF scheduling

ii. FIRST IN FIRST OUT (FIFO) SCHEDULING

A similar task list is also scheduled using the FIFO algorithm and its corresponding results are plotted in a graph. The tasks completed using FIFO algorithm is similar to tasks completed using SJF algorithm. But the quality at which tasks are completed is low (0.6 to 0.9). And the quality is not stable, as there are some ups and downs which are shown clearly in the graph.

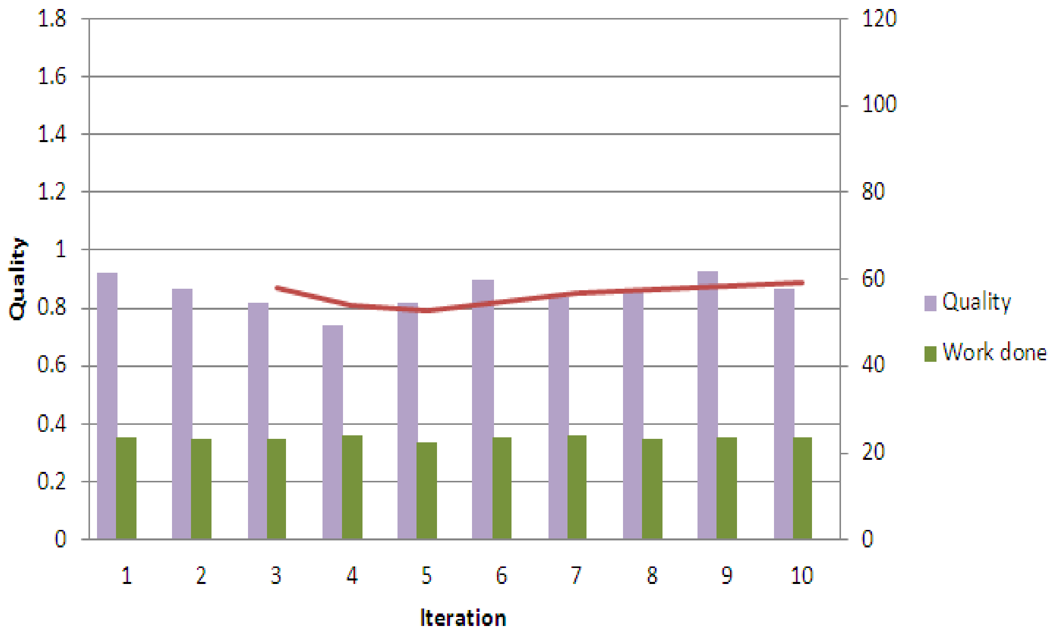


Figure 4.7: Quality and work done after FIFO scheduling

iii. COMMITMENT BASED POLICY (CBP)

Unlike SJF and FIFO, CBP proved to be more efficient. The work done was also significantly large and they were completed at good quality. From this chart it is clear that CBP is more suitable for the incoming task sequence. Sine tasks arrive at varying size and complexity; hence there is sudden raise in the quality of the work done.

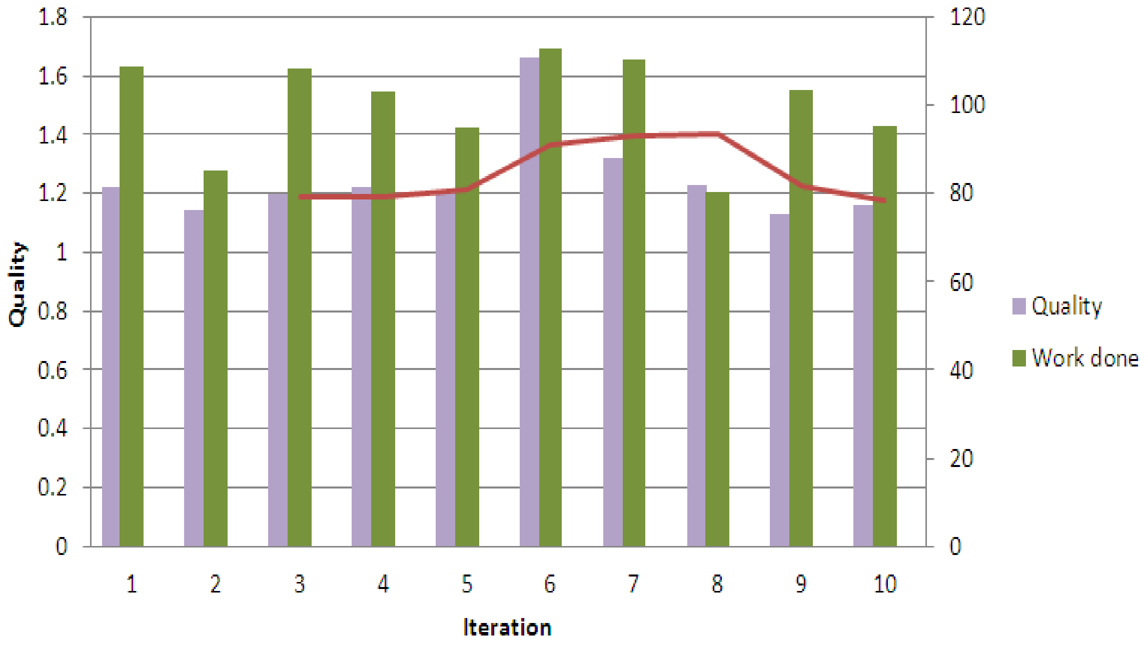


Figure 4.8: Quality and work done after Commitment Based Policy

QUALITY DISTRIBUTION

The quality of tasks being completed by all three scheduling policies is compared. The following graph clearly shows that FIFO scheduling algorithm completes tasks at low quality and the performance of SJF is much better than FIFO. But CBP completes more number of tasks and also at better quality.

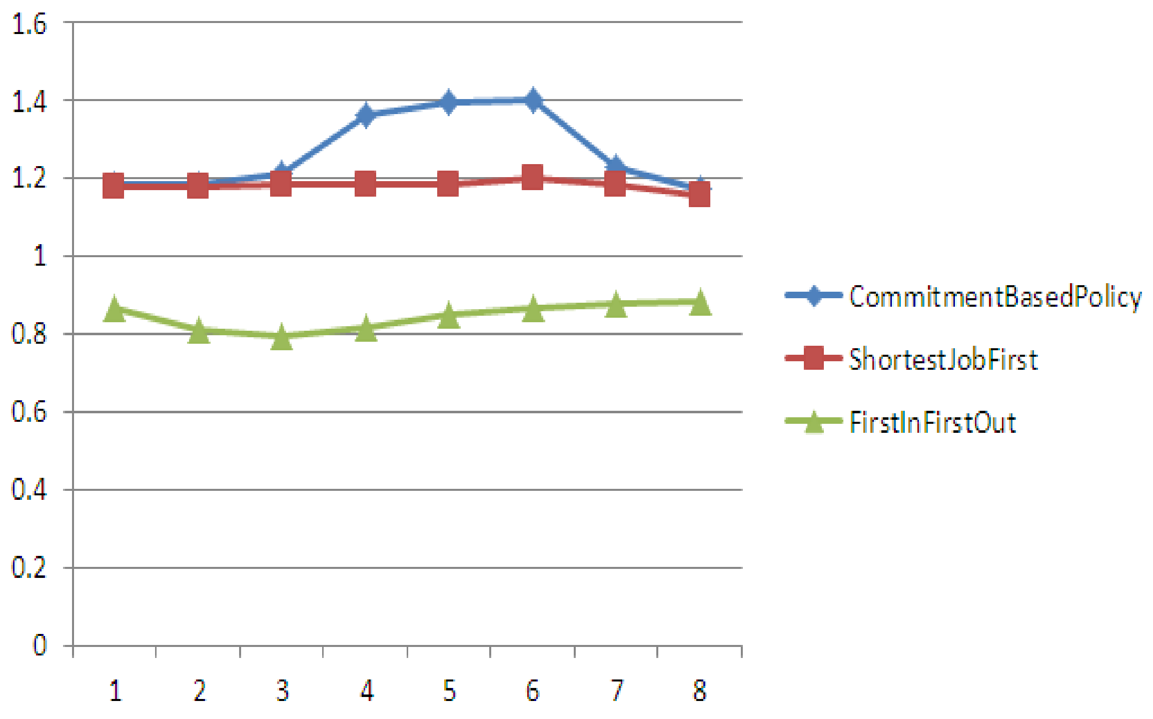


Figure 4.9: Moving average of quality

Hence the portfolio helps in understanding the consequences of the scheduling policies over quality of completed tasks. This assists the manager in choosing the most appropriate scheduling policy.

5. RESULTS AND CONCLUSION

The main outcome of this thesis includes the development of a simulator which mimics the real world functionality of project management in an organization. The simulator simplifies the workload of the manager who is responsible for making critical project management decisions.

The simulator generates tasks which replicate project requests that arrive in any organization. The tasks are generated according to the user's specification or they can also be generated using default values. Once the user makes the particular selection, tasks are generated by the simulator. Each tasks consists of three basic parameters,

1. Size
2. Value
3. Complexity

The values for these parameters are generated using Poisson, Normal and Uniform distribution. The inputs for the distributions are either user specified or default simulator generated values. The generated tasks are now applied to different scheduling policies. The choice of scheduling policy is also made by the user. In this thesis, three scheduling policies are implemented:

1. Shortest Job First algorithm
2. First In First Out
3. Commitment Based Policy

The end user has to make a selection among these three policies. And now the generated task list is scheduled using the chosen policy. During policy scheduling, two other essential parameters are defined. These parameters play a major role in choosing a task which needs to be completed and they are:

- ▫ Commitment (0.5 or 1.0 or 1.5)
- ▫ Priority (1 or 2 or 3)

These attributes are defined for each task based on its corresponding size, complexity and value. The commitment parameter indicates the commitment of development team towards a task. The value 0.5 indicates low commitment towards a particular task and 1.5 indicates high commitment. Any task with commitment of 1.5 is most likely to be chosen for processing. The priority for each task is a numerical value ranging from 1 to 3, where 1 denotes lowest priority and 3 denotes highest priority.

Based on these parameters, resources are allocated to preferred tasks to be completed. Each completed task is further classified using two parameters:

- Quality of task during completion
- Quality Points of task

These features primarily depend upon the following factors:

- Resources available
- Task commitment
- Task Priority

A final report is generated which indicates the cumulative quality of all the completed tasks over several iterations. And the report also indicates the current quality of the completed tasks after several trials and iterations.

The final report and graphical representation facilitate the manager in making realistic decisions. This thesis helps the manager take crucial decision by eliminating uncertainties and errors in decision. All the possible cases are projected by the simulator by applying various scheduling algorithms and applying tasks with different parameter values. The best solution can easily be chosen by the manager over all possible solutions and can be applied in real world cases. Therefore any margin for error is eradicated and a fool proof policy can be applied.

The thesis helps in avoiding any mishaps and loss of resources due to incorrect decision or unexpected results. Hence the simulator assists in taking managerial decisions with optimal utilization of resources without any loss to the organization. The available resources are put to maximum utilization, which is the primary motto behind the thesis.

6. FUTURE SCOPE

Experimental results show the implementation and working of the Shortest Job First algorithm and the Commitment Based Policy scheduling algorithms. The implementation of only two scheduling algorithms is one of the limitations of the available system. Many other popular scheduling algorithms can also be implemented in the form of extensions to the present simulator.

Apart from the defined parameters (Size, Value and Complexity) another critical parameter is also essential for determining the task commitment and priority namely:

- Age of task
- Urgency of task

The age of any tasks specifies the time a task has been waiting in the queue for processing. This is quite an essential factor since any task waiting in the queue for a considerable long duration would lead to loss of task quality. Therefore a task should not be allowed to 'starve'. The starvation issue is addressed by giving priority to tasks which have been in the queue for an extensive time period.

The simulator allows tasks to be generated with the following specifications:

- 20% of task are generated with high priority,
- 40% of tasks are generated with medium priority and
- 40% of tasks are generated with low priority.

This can be further enhanced, allowing user to specify the urgency of tasks before they are generated. In real world each task arrives with different levels of urgency. And this mainly depends on the following factors:

- Demand for a task
- Price offered by the customer(line staff) to complete the task
- Estimated ROI
- Authority of the calling customer(line staff)

Hence all these factors can be taken into consideration while determining task urgency.

BIBLIOGRAPHY

[1] Jenney W. Ross, David F. Feeny: *The evolving role of the CIO*

August 1999, Massachusetts Institute of Technology

[2] Managing Information Technology in Turbulent Times: *Louis Fried*

John Wiley & Sons, Inc., 1995

[3] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne: *Operating Systems Concept, 7th edition.* John Wiley & Sons, Inc., 2005

[4] Richard B. Chase, Nicholas J. Aquilano, F. Robert Jacobs: *Production and Operations Management.* Irwin/McGraw-Hill, 1998