## 1.1: Introduction

The process of building computer software and information systems has been always dictated by different development methodologies. A software development methodology refers to the framework that is used to plan, manage, and control the process of developing an information system.

Software Development Life Cycle (SDLC) has been studied and investigated by many researchers and practitioners all over the world, and numerous models have been proposed, each with its own acknowledged strengths and weaknesses. The Waterfall, spiral, incremental, rational unified process (RUP), rapid application development (RAD), agile software development, and rapid prototyping are few to mention as successful SDLC models, (Munassar, Govardhan et al 2010).

a Computer simulation is a program that tries to simulate an abstract model of a particular system. In practice, simulations can be employed to discover the behavior, to estimate the outcome, and to analyze the operation of systems.

Process simulation has become a powerful technology in support of software project management and process improvement. There is a need for simulation models of software development processes to assist the project manager to determine the best choice of SDLC according to the available resources of software companies.

The research aims to provide a new simulation model and to use this new model to analyze and predicate the cost of SDLC approach.

## 1.2: Problem of the research

There is a need for simulation models to assist the project managers to address the following:

- Inability to accurately predict the appropriate number of developers who works on the project's phases.
- Some SDLC phases delayed due to the dependent of development team members.
- Bottleneck between the arrival and delivery projects.

## 1.3: Objective of the research

The objectives of this research is the following:

- Provide a Systematic evaluation method to predict the team size using the development process simulator.

## 1.4: significant of the research

The significant of research is the following:

- It's Provide an evaluation mechanism for existing process design without executing the actual process.
- A simulation model will allow for evaluating different scenario in project using the simulation approach.
- A proposed model will assist the project manager to assign the resources intelligently on the various SDLC phases.

## 1.5: Hypothesis of the Research

The hypothesis of the research is the following:

- The proposed model will improve the utilization of team members who works on different phases of project.
- The proposed approach will estimate the team size that required to make the company follow up with incoming projects.

## 1.6: The Structure of the Thesis

Chapter one contains the introduction, problem of the research, objective, significant of the research, hypotheses and research methodology.

Chapter two contains of Literature Review and related works about software development process modeling and simulation.

Chapter three contains methodology, data collection, analysis and data preparation.

Chapter four contain the implementation of the simulation model (Iterative/Incremental simulation model) for different size of projects.

Chapter Five contains a research result and discussion,

Chapter Six contains conclusion and suggestion for future work.

## 2.1: Introduction:

The System Development Life Cycle framework provides a sequence of activities for system designers and developers to follow. The ideas about the software development life cycle (SDLC) have been around for a long time and many variations exist, such as the waterfall, spiral, prototype and rapid application development model (RAD).

These variations have many versions. varying from those which are just guiding principles, to rigid systems of development complete with processes, paperwork and people roles. It consists of a set of steps or phases in which each phase of the SDLC uses the results of the previous one. Various software development life cycle models are suitable for specific project related conditions which include organization, requirements stability, risks, budget and duration of project, (Munassar, Govardhan et al 2010).

Each SDLC has its advantages and disadvantages making it suitable for use under specific condition and constraints for specified type of software only. We need to understand which SDLC would generate most successful result when employed for software development.

# literature Review

## 2.2: Software Development Life Cycle(SDLC)

SDLC is a framework that describes the activities performed at each stage of a software development project. It is a methodology for designing, building, and maintaining information and industrial systems. So far, there exist many SDLC models:

### 2.2.1: Waterfall Model

The Waterfall Model is the oldest and the most well-known SDLC model. This model is widely used in government projects and in many major companies. The special feature of this model is its sequential steps. It goes downward through the phases of requirements analysis, design, coding, testing, and maintenance. Moreover, it ensures the design flaws before the development of a product. This model works well for projects in which quality control is a major concern because of its intensive documentation and planning, (Mujumdar, Masiwal et al 2012). Stages that construct this model are not overlapping a stage, which means that the waterfall model begins and ends one stage before starting the next one.

### 2.2.2: Spiral Model

The spiral model is a software development process combines elements of both design and prototyping in stages for the sake of combining the advantages of top down and bottom up concepts. it focuses on risk assessment and minimizing project risk. This is can be achieved by breaking a project into smaller segments, which then provide more ease-of change during the development process. In this model, the development team starts with a small set of requirements and then goes through each development phase for those set of requirements. In this model, each iteration prior to the production version is called a prototype of the application, (Munassar, Govardhan, 2010).

### 2.2.3: Iterative and Incremental Model

This model combines elements of the waterfall model in an iterative fashion. Moreover, each linear sequence produces deliverable increments of the software. The basic requirements are addressed in the first increment, and it is the core product, however, many supplementary features (some known, others unknown) remain undeliverable at this increment. This model constructs a partial implementation of a total system. Then, it slowly adds increased functionality. Therefore, each subsequent release will add a function to the previous one until all designed functionalities are implemented, (Mujumdar, Chawan, et al, 2012)

### 2.2.4: Prototype Model

The basic idea here is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. It is suitable for user interface prototyping, workflow simulation, workload simulation, technology comparisons situation (Qureshi and Hussain ,2008).

### 2.2.5: Rapid Application Development

Rapid Application Development (RAD) is an incremental software process model which is a "high-speed" adaptation of the linear sequential model in which rapid process is achieved by using component-based construction (Chopra and Kapoor 2016).

### 2.2.6: Agile Model

Agile system process life cycle model has developed as a part of reaction in mid of 1990. This software will manage the problem of heavy weight, micro management of any project. Finally, Agile has developed in 2001, called by lightweight method. Agile contain method like scrum, crystal cleaner. Agile use different method

for different type of project but they can share common characteristic among the project.

In Agile SDLC, requirement specification can change frequently because they will understand by the customer and software developer. It is found that cost of this model is very high (Chopra and Kapoor,2016). Developer can control over the cost, if he acquires only needed requirement then cost can be covered by developer. This model is very difficult to implement but with low risk involvements.

### 2.2.7: Rational Unified Process (RUP) Model

It is found that RUP model is not suitable for small projects. IBM Rational Method Composer allows you to easily customize RUP to meet the unique needs of your project. It enables you to select and deploy only the process components you need, and then publish it through your intranet (Kruchten, 2004).

## 2.3-Factors affecting the industry of software

There are many factors affect directly on the software delivery process represent as following:

1. Type of project.
2. Size of project.
3. Duration of project.
4. Complexity of project.
5. Type and level of Expected risk.
6. Understanding level of user requirements.
7. Understanding level of application areas.
8. Involvement of customer.
9. Developer’s experience.
10. Team size.
11. Man, machine interaction.
12. Availability of technology and tools.
13. Software product versions.
14. Required level of reliability.

## 2.4: Related work:

(Bassil,2012) proposed a simulation model for the Waterfall model from the analysis to maintenance phase, the model simulate the different stakeholders involved in the Waterfall which are essential throughout the whole development process. The simulation includes related resources, input, workflow and output. The main problem that many projects do not deliver on time, the major reason for this a project manager are not intelligent assigning the required number of employee and resources on the various activities of the SDLC. for this reason, some phases maybe delayed duo to the insufficient number of worker while other dependent phase may stay idle doing nothing but waiting for other phases to complete consequently.

The proposed simulation model (waterfall) aimed to finding the trade-off of cost, schedule and functionality for the benefit of the project outcome, it helps maximizing the utilization of development process by keeping all employee and resources busy all the time to keep pace with the incoming project and reduce waste idle time.

They used a simulation tool called Simiphony.net, it is a simulator that consist of a working environment and a foundation library that allow the development of new scenario in an efficient manner. The result show that the system reached the optimal state when the total number of project received was equal to the total number of project delivery without any loss in time or schedule.

(Thind and Karambir,2015) were made an empirical study Indian software firm, according to this study they found there are many factors which play an important role in the selection of SDLC model. One of the most important factors in this study is team size. The author proposed A simulation model for the spiral software development life cycle. spiral model is divided into many framework activities, these activities represent segment of the spiral path. They found out many software projects doesn’t deliver on time and within budget and as the expectation of the customer. There is a need to

assignment of expert team member to a various phase of SDLC model because some phases of process model maybe stay idle while other maybe delayed due to the insufficient number of resources. The proposed simulation model use Simiphony.net to simulate the Spiral model. The simulated model shows there are increase of the utilization of different processes by keeping all development team members busy all the time so that help in decrease idle and waste time.

(Fogle and Qu,2015) proposed An Extended Simulation Model for Managing Dynamic Change in Software Development Project because Software project are known for continuing to have a high occurrence of failure despite the many standard estimation tool, metric, and risk management technique that have been developed. One important factors that leads to software failure is the difficulty in reacting appropriately to unexpected changes in the project resources and schedule.

There is a need for change management tool that can track software project change and predicate the probable impact of those dynamic changes on the schedule and budget. The existing project estimation tools have been effective in providing some sort of planning guideline at the beginning of the project but the accuracy rates of those methods are still low by the end of the project cycle.

They found an existing software estimation tools don't model the impact of dynamic change that happen during the software development process and the current tools do not represent the realities of software development life Sycle (SDLC) and are not able to provide a view of the project outcome based on current project status and dynamic change that occur. the root causes of project failure or delay are not known until after the project has been completed.

They used ProjScout model that incorporates the effect of presentation number of communication overhead, training overhead and productivity. The completion time result show that when positive resources and productivity changes are applied earlier in

the project, the completion time is shorter than when those same changes are made late in the project, for small project scenarios the addition of resources reduced the completion time from a base line of 255 hours to 206 hours conversely the loss of resources led to a longer completion time of 279hours.

(Chauhan,2016) proposed Rapid Revision Software Development Life Cycle Model (RR-SDLC Model) Based on Concept of Reusability of Software. The basic objective of SDLC models is to deliver high quality product delivered on time that provide strong control on quality, maximize the profit in terms of cutting cost on product development. They find that a traditional SDLC model provide sequential or hierarchical development approach which causes high cost and time in completion of project to release the final product. The studies and research on SDLC models show there is a need of more adaptable type of SDLC model which can improve quality, reusability and rapid releasing the software product, one of the most common factor among all study is releasing the product in time, reduce the cost and providing more satisfaction to acquire the product.

The proposed RR-SDLC model emphasizes on developing product from some similar old product or new component in it as to meet the requirement of a dissimilarity by modifying the existing similar product or adding new component. The proposed model has been compared with Traditional SDLC model for Test Failure to determine the best case and worst case, the result show the chance of error due to error handler configuration failure, infrastructure and use awareness also get reduce. The proposed model can be considered as an approach for creating new SDLC model that is more practical.

(Srinivasan and Agila,2014) applied the concept of clemency brass derives itself from the core concept of project management in software engineering. software how-so-ever efficient and effective cannot be consider commercially successful until and unless the software remains in the market for sufficiently long duration, in order

to recover the cost that incurrent during development and deployment of the software. The clemency brass process is relatively new but rapidly growing discipline within software engineering of managing software release. As software system, software development process and resources become more specialized and complex.

The proposed simulation model is build using the Simiphony.net simulation tool and with the concept of clemency brass, a project in simiphony.net is made out of collection of modeling element linked to each other by logical relationship. A new SDLC model is evolved making the fullest use of clemency brass, it increases the effective and software life in the market.

(Abdulaziz,2015) proposed a new and effective Agile Software Development method, the method has been satisfactory experimented by the Royal Saudi Air Force (RSFA) called "Software Project Management" (SPM), this tool gives the development team to control and manage software project resources more effectively by increasing team collaboration and productivity thus decreasing the amount of time needed to complete the project.

The proposed model aims to understand the business model of organization and identified the problem and find the best solution then implementation which transform the model into an executable programming code. then the test that evaluates the system for quality purpose, this include finding bugs and ensuring that requirement is met next deployment which aims to deliver the system to the end user.

The proposed (SPM) Methodology is easy to amend during the development process to achieve customer's needs and their requirement which a specified time-scale and deliver. SPM involves customer satisfaction, therefore changing requirement are welcomed, software delivered frequently in small pieces, developer and client work together in self-organizing team. The methodology is simple, easy to apply and understand.

SPM contains the characterizes of Agile i.e. project is broken into small modules to short development cycles. This generate fixed periods to achieve development prioritizes requirement, minimize risk, is incremental people oriented and encourages team collaboration.

(Aggarwal, Prakash, et al 2008) proposed a Content Management System Effort Estimation model (CMSEEM) for the current technologies using which a piece of work can be estimated more accurately.

The data collected from twelve completed projects taken from industry and seventy different projects completed by the students. These projects are categorized based on their size and total/build effort ratio. The size of the project is estimated by using the modified object point analysis approach. The complexity of the project is determined by using a set of questionnaires which has to be filled by the project managers after completing the initial requirement analysis. The nominal size estimated after object point analysis is finalized using the adjustment factors which are calculated by considering the different characteristics of the system such as production and general system characteristics.

The estimated effort is further phase wise distributed for better scheduling of the project. The proposed model is refined using the linear regression approach. Finally, the model is validated using another questionnaire which has to be filled after completing the project. The proposed model shows a great improvement as compared to the earlier models used in effort estimation of Web CMS projects.

(Lawanna,2014) proposed a simulation technique to determine the appropriate number of Programmers in the Process of Software Testing. They found one of the main problems in software engineering is to determine the appropriate numbers of programmer working through the software-development life cycle, particularly in the process of coding, testing, and maintenance. The high numbers of the programmer

increase the cost of developing software. However, the small teams cause another problem, especially in the process of testing software. They focused on two main drawbacks remained in the software maintenance. The first, how many programmers are suitable in the process of maintenance. The second, how many bugs will be occurred. The reasons that fail the software are not only the new bugs or faults, but it includes the requirement specifications from managers, users, stakeholders, and the lines of code. Therefore, the objective of the proposed model is to determine the appropriate numbers of programmers whereas the changes from requirement specification, lines of code, and bugs are being occurred. Another is to define the possibility of bugs, which can be occurred in the process of software maintenance.

The Author applies seven programs written by C language in the experiment. The seven programs cooperatively are called the Siemens Programs that are well-known and often used in the field of software maintenance. First, they produced test cases by black-box technique, after producing a set of test cases, the specialists created another set of test cases by hand due to the white-box technique.

The important of the proposed method are the high number of programmers are expensive and costly. The proposed model presents the methods of finding the suitable number of programmers. Simulation technique concerns mathematical model to avoid the errors of using irrelevant or subjective factors such as quality of programmers, skill, and knowledge. According to the purpose model, testing times are varied from 1 to 24 hours (overnight) regarding the requirement specification by the users.

This assumption is considered for planning and preparing the programmers, which are suitable in the process of testing codes. Codes approximately are 150 lines in order to avoid the complexity of fixing errors such as bugs, each program consists difference lines of code depending on the types of software and the business objectives. the results show that sometimes the lower numbers of programmers can finish the testing codes within the less time. This may be because of their abilities and experiences.

(Suri and Bhushan,2007) proposed a simulation model for time /effort estimation of software development process. This simulator will be an asset to affordably keep track of time during the process of development and thus to satisfy the client in this era of competitive market of software.

The total mean time of software development has been found and computed by varying the average time of completion of an activity. The input for the simulator has been derived by using an algorithm for generating pseudo random numbers which follows Erlang-6 distribution. The motivation for incorporating simulation into software development process for time estimation with simulated data lead to useful information for the completion of the project well in time.

Timing being very critical in software development, if a delay happens in the development activity, the market could be taken over by the competitors. Also, if a 'bug' filled product is launched in a short period of time (quicker than the competitors), it may affect the reputation of the company. So, there should be a trade-off between the development time and the quality of the product.

Customers don't expect a bug free product but they expect a user-friendly product. That results in customer satisfaction.

## 3.1: Research Methodology

This chapter of research illustrate the methodology that has been followed , this figure represents each process has been done to complete the process of simulation

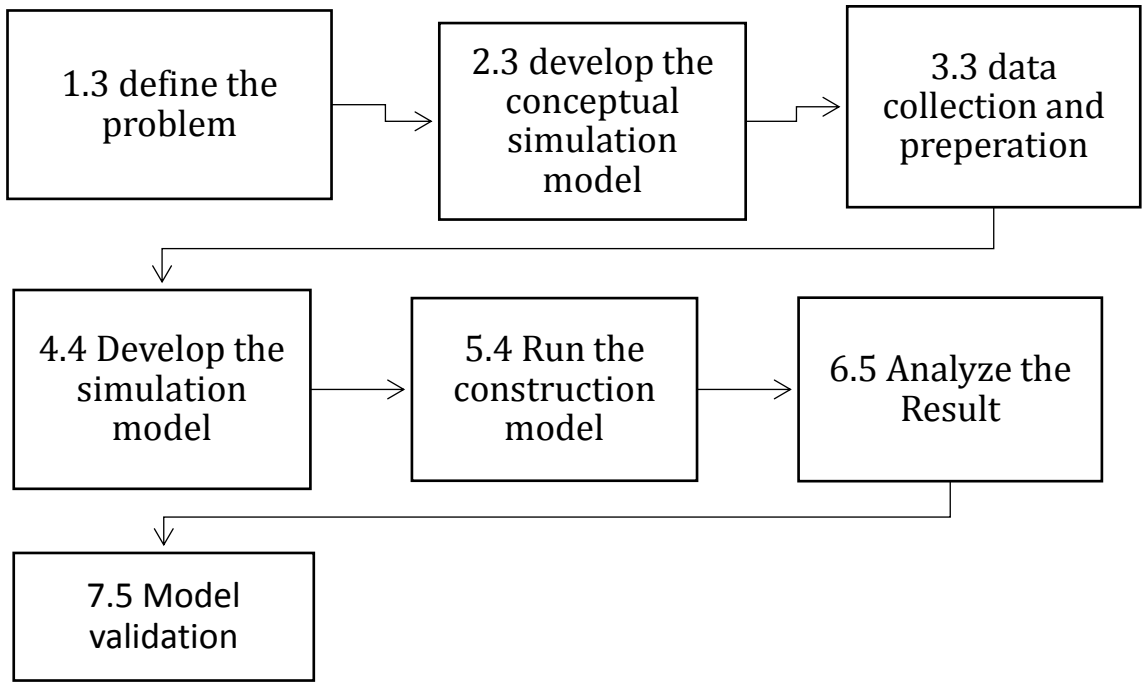


Fig 3.1 The simulation model framework.

**1.3: define the problem**

This phase describes the problem which the proposed model will solve it, this problem represents of the delay that happen during the software development process in software companies, the study shows many software projects doesn't deliver on time.

**2.3: develop the conceptual Simulation model**

In this phase, a conceptual model built to simulate the most famous methodology that it used by the software development companies in Sudan-Khartoum and create the work flow diagram for it, and specify each phase and the requirement behind it.

### 3.3: data collection & preparation

This phase represents the required number of team for each phase and the required time to deliver different scale projects per day and the probability of delay for each phase during the development process. This data prepared to be an input for the simulation model and described it in triangular formula in upper limit and lower limit per day.

### 4.4: Develop the simulation model

In this phase, a simulation model built using (Simphony.net) simulation tool which provide a simulation environment for modelling and simulating different kind of systems. The proposed simulation model for Iterative/Incremental model executed as following steps:

- The Iterative/Incremental model Divided into independent phases.
- Understand the concept and the requirements that lie behind every phase.
- Define the resources, tasks, entities, and the work flow of every model's phase.
- Simulate each phase apart.
- Integrate the whole phases together, simulate the system, and record results.

### 5.4 Run the construction model

The model has been executed for 5 increments about 3000 milliseconds until the result reach to stable result.

### 6.5 Analyze of Result

In this phase, the simulation model set out a statistic result, this result explains the maximum utilization of resources and the number of arrival and deliver projects.

Also, it shows a graphical representation for resources utilization that assigned for each phase in the simulation model.

### 7.5 Model validation

There are two type of validation observable and Non-observable. The Non-observable type has been used in this model.

Non-observable System divided to:

1- Explore Model Behavior

The simulation model output behavior can be explored either qualitatively or quantitatively.

2- Comparison to Other Models

There are three basic approaches used in comparing the simulation model output behavior to either the system output behavior or another model output behavior:

a) the use of graphs to make a subjective decision,
b) the use of confidence intervals to make an objective decision,
c) the use of hypothesis tests to make an objective decision.

In this simulation model graphs have been used because it's the most commonly used approach for operational validity.

## 3.2: Data Analysis

A questionnaire and personal interviews were conducted to collect information from 15 software development companies in Khartoum -Sudan to know team size that make the companies in continues development with incoming project. A Statistical analysis program SPSS (Statistical Package for Social Studies) has been used to analysis the data and prepared it for the simulation model.

**Table 3.1 The Criteria used to specify the Size of project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | Number of Modules | 9 | 60.0 |
| | Degree of complexity | 2 | 13.3 |
| | Number of Modules& Complexity | 4 | 26.7 |
| | Total | 15 | 100.0 |

From the above table 60% of companies measure the size of project depends on the number of modules, 26% measure it depends on number of modules and complexity,13% measure it depends on degree of complexity.

**Table 3.2 The numbers of Modules that build the small scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | One Module | 13 | 86.7 |
| | Between 2-3 Modules | 2 | 13.3 |
| | Total | 15 | 100.0 |

a module is a part of a program. Programs are composed of one or more independently developed modules that are not combined until the program is linked. A single module can contain one or several routines.

From the above table 86.7% of companies said the small-scale project consist of one modules and 13.3% said the small-scale project consist between 2-3 modules.

**Table 3.3 The numbers of Modules that build the medium scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 4-5 Modules | 7 | 46.7 |
| | 6-7 Modules | 8 | 53.3 |
| | Total | 15 | 100.0 |

From the above table 53.3% of companies said the medium-scale project consist 6-7 modules and 46.7% said the medium-scale project consist between 4-5 modules.

**Table 3.4 The numbers of Modules that build the Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | More than 9 Modules | 15 | 100.0 |

From the above table, the large-scale project consists of more than 9 modules

**Table 3.5 The Size of projects a company developing it**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | Small & Medium scale | 2 | 13.3 |
| | Small&Meduim&Large scale | 13 | 86.7 |
| | Total | 15 | 100.0 |

From the above table 86.7% of companies implement the three types of project and 13.3% implement small and medium scale project.

**Table 3.6 The methodology that a company follow it**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | Rapid Prototyping Model | 3 | 20.0 |
| | Iterative/Incremental Model | 9 | 60.0 |
| | Agile(XP) | 3 | 20.0 |
| | Total | 15 | 100.0 |

From the above table 60% of companies follow Iterative/Incremental model and 20% follow Rapid prototyping model and 20% follow Agile(XP).

**Table 3.7 How many Business Analyst required for Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1 Business Analyst | 5 | 33.3 |
| | 2 Business Analyst | 10 | 66.7 |
| | Total | 15 | 100.0 |

The analysis shows 66.7% of companies assign 2 Business Analysis for small-scale project and 33.3% of companies assign 1 Business Analysis for small-scale project

**Table 3.8 How many Designer required for Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 2 Designer | 13 | 86.7 |
| | 3 Designer | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies assign 2 Designer for small-scale project and 13.3% of companies assign 3Designer for small-scale project.

**Table 3.9 How many Developer required for Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1 Developer | 3 | 20.0 |
| | 2 Developers | 12 | 80.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies assign 2 Developer for small-scale project and 20% of companies assign 1 Developer for small-scale project.

**Table 3.10 How many Tester required for Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1 Tester | 11 | 73.3 |
| | 2 Testers | 4 | 26.7 |
| | Total | 15 | 100.0 |

The analysis shows 73.3% of companies assign 1 Tester for small-scale project and 26.7% of companies assign 2 Testers for small-scale project.

**Table 3.11 How many Maintains required for Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1Maintiant | 12 | 80.0 |
| | 2 Maintains | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies assign 1 maintenance Man for small-scale project and 26.7% of companies assign 2 Testers for small-scale project.

**Table 3.12 Numbers of Business Analyst required for Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 2 Business Analyst | 4 | 26.7 |
| | 3 Business Analyst | 11 | 73.3 |
| | Total | 15 | 100.0 |

The analysis shows 73.3% of companies assign 3 Business Analysis for Medium-scale projects and 26.7% of companies assign 2 Business Analysis for Medium-scale projects.

**Table 3.13 Numbers of Designer required for Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 2 Designer | 6 | 40.0 |
| | 3 Designer | 9 | 60.0 |
| | Total | 15 | 100.0 |

The analysis shows 60% of companies assign 3 Designer for Medium-scale projects and 40% of companies assign 2 Designer for Medium-scale projects.

| Table 3.14 Numbers of Developer required for Medium-Scale project | | | |
|---|---|---|---|
| | | Frequency | Percent |
| Valid | 2 Developers | 3 | 20.0 |
| | 3 Developers | 12 | 80.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies assign 3 Developers for Medium-scale projects and 20% of companies assign 2 Developers for Medium-scale projects.

**Table 3.15 Numbers of Tester required for Medium-Scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 2 Testers | 12 | 80.0 |
| | 3 Testers | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies assign 2 Testers for Medium-scale projects and 20% of companies assign 3 Testers for Medium-scale projects.

**Table 3.16 Numbers of maintainers required for Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1Maintant | 3 | 20.0 |
| | 2 maintainers | 10 | 66.7 |
| | 3 maintainers | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 66.7% of companies assign 2 maintainers for Medium-scale projects and 20% of companies assign 1Maintant for Medium-scale projects and 13.3% of companies assign 3 maintainers for Medium-scale projects.

**Table 3.17 Numbers of Business Analyst required for Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 7 Business Analysts | 14 | 93.3 |
| | 4 Business Analysts | 1 | 6.7 |
| | Total | 15 | 100.0 |

The analysis shows 93.3% of companies assign 7 Business Analysis for Large-scale projects and 6.7% of companies assign 4 Business Analysis for Large-scale projects.

**Table 3.18 Numbers of Designer required for Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 7 Designer | 12 | 80.0 |
| | 4 Designer | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies assign 7 Developer for Large-scale projects and 20% of companies assign 4 Designers for Large-scale projects.

**Table 3.19 Numbers of Developers required for Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 7 Developer | 4 | 26.7 |
| | 10 Developer | 11 | 73.3 |
| | Total | 15 | 100.0 |

The analysis shows 73.3% of companies assign 10 Developers for Large-scale projects and 26.7% of companies assign 7 Developers for Large-scale projects.

**Table 3.20 Numbers of Tester required for Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 3 Testers | 4 | 26.7 |
| | 5 Testers | 11 | 73.3 |
| | Total | 15 | 100.0 |

The analysis shows 73.3% of companies assign 5 Testers for Large-scale projects and 26.7% of companies assign 3 Testers for Large-scale projects.

**Table 3.21 Numbers of maintainers Man required for Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 3 **maintainers** | 6 | 40.0 |
| | 5 **maintainers** | 9 | 60.0 |
| | Total | 15 | 100.0 |

The analysis shows 60% of companies assign 5 Maintenance Man for Large-scale projects and 40% of companies assign 3 Maintenance Man for Large-scale projects.

**Table 3.22 the delay when software release**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | Yes | 15 | 100.0 |

The analysis shows all software project doesn't deliver on time.

**Table 3.23 The reasons of delay in software delivery**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | poor Estimate of delivery time by project manager | 5 | 33.3 |
| | poor estimation of project size and requirement | 8 | 53.3 |
| | the quality of team and their skills | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 53.3% of project delay because poor estimation of project size and 33.3% of delay because the poor estimation of release time of the project and 13.3% of delay return to the quality of team and their skills.

**Table 3.24 The days required for Business Analysis in Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1-3 days | 13 | 86.7 |
| | 4-6 days | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies required 1-3 days for Business analysis phase in small-scale project and 13.3% of companies required 4-6 days for Business analysis.

**Table 3.25 The days required for Design in Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1-3 days | 2 | 13.3 |
| | 4-6 days | 13 | 86.7 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies required 1-3 days for Design phase in small-scale project and 13.3% of companies required 4-6 days for Design phase.

**Table 3.26 The days required for Development in Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 12 | 80.0 |
| | 7-9 days | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 10-15 days for Development phase in small-scale project and 20% of companies required 7-9 days for Development phase.

**Table 3.27 The days required for Testing in Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1-3 days | 12 | 80.0 |
| | 4-6 days | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 1-3 days for Testing phase in small-scale project and 20% of companies required 4-6 days for Testing phase.

**Table 3.28 The days required for maintainers in Small-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 1-3 days | 12 | 80.0 |
| | 4-6 days | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 1-3 days for Maintenance phase in small-scale project and 20% of companies required 4-6 days for Maintenance phase.

**Table 3.29 The days required for Business Analysis in Medium-scale projects**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 12 | 80.0 |
| | 16-20 days | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 10-15 days for Business analysis phase in Medium-scale project and 20% of companies required 16-20 days for Business analysis phase.

**Table 3.30The days required for Design in Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 3 | 20.0 |
| | 16-20 days | 12 | 80.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 16-20 days for Design phase for Medium-scale project and 20% of companies required 10-15 days for Design phase.

**Table 3.31 The days required for Development in Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 2 | 13.3 |
| | 30-60 days | 13 | 86.7 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies required 30-60 days for Development phase for Medium-scale project and 13.3% of companies required 10-15 days for Development phase.

**Table 3.32 The days required for Testing in Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 13 | 86.7 |
| | 7-10 days | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies required 10-15 days for Testing phase for Medium-scale project and 13.3% of companies required 7-10 days for Testing phase.

**Table 3.33 The days required for maintainers in Medium-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 10-15 days | 3 | 20.0 |
| | 7-10 days | 12 | 80.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 7-10 days for Maintenance phase for Medium-scale project and 20% of companies required 10-15 days for Maintenance phase.

**Table 3.34 The days required for Business Analysts in Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 30-40 days | 13 | 86.7 |
| | 15-30 days | 2 | 13.3 |
| | Total | 15 | 100.0 |

The analysis shows 86.7% of companies required 30-40 days for Business analysis phase in Large-scale project and 13.3% of companies required 15-30 days for Business analysis phase.

**Table 3.35 The days required for Design in Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 30-40 days | 3 | 20.0 |
| | 40-60 days | 12 | 80.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 40-60 days for Design phase in Large-scale project and 20% of companies required 30-40 days for Design phase.

**Table 3.36 The days required for Development in Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 60-90 days | 1 | 6.7 |
| | 60-150 days | 14 | 93.3 |
| | Total | 15 | 100.0 |

The analysis shows 93.3% of companies required 60-150 days for Development phase in Large-scale project and 6.7% of companies required 60-90 days for Development phase.

**Table 3.37The days required for Testing in Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 30-40 days | 12 | 80.0 |
| | 15-30 days | 3 | 20.0 |
| | Total | 15 | 100.0 |

The analysis shows 80% of companies required 30-40 days for Testing phase in Large-scale project and 20% of companies required 15-30 days for Testing phase.

**Table 3.38 The days required for maintainers in Large-scale project**

| | | Frequency | Percent |
|---|---|---|---|
| Valid | 30-40 days | 1 | 6.7 |
| | 40-60 days | 1 | 6.7 |
| | 15-30 days | 13 | 86.7 |
| | Total | 15 | 100.0 |

The analysis shows 86% of companies required 15-30 days for Maintenance phase in Large-scale project and 6.7% of companies required 30-40 and 40-60 days for Maintenance phase.

**The probability of Delay:**

The study shows each size of projects upon completion have probability of delay: for small scale project about 5%, for medium scale project about 10% and for large scale project about 20%.

**Factors that considered in this proposed model:**

1- Team Size

2- Duration of project

3- Scale of project

4- Understand level of user requirement

## 3.3: Data preparation

From the study, a project arrives randomly at a software firm with inter-arrival time from a Triangular distribution with a lower limit of 20 days, an upper limit of 60 days, and a mode of 40 days. The probability density function is then given as:

x= number of days

a=lower limit of days b= average days c= upper limit of days

$$f(x) = \begin{cases} 0 & \text{for } x < 20 \\ \dfrac{2(x-a)}{(b-a)(c-a)} & \text{for } 20 \le x \le 40 \\ \dfrac{2}{b-a} & \text{for } x = 40 \\ \dfrac{2(b-x)}{(b-a)(b-c)} & \text{for } 40 < x \le 60 \\ 0 & \text{for } 60 < x \end{cases}$$

the projects divided into three groups based on their scale, the study found that 60% of the projects are small-scale projects, 30% are medium-scale projects, and 10% are large-scale projects. Each project will require a different mix of specialists, employees, and resources to be delivered based on the scale of the project, the study shows companies assign the resources on project as following:

The average of available resources at the software companies are the following:

- 7 Business Analyst
- 7 Designers
- 10 Developers
- 5 Testers
- 5 Maintenances

In the Iterative/Incremental Model there exist the following tasks:

- Business Analysis
- Design
- Implementation
- Testing
- Maintenance

The model has been built by assuming the team member work as employee full time per day.

The duration for every phase to be completed is defined as follows:

**3.3.1: Small Scale-Project:**

The business analysis phase requires a Uniform distribution with a lower limit of 1 days and an upper limit of 3 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 1 \le x \le 3 \\ 0 & \text{for } x < 1 \text{ or } x > 3 \end{cases}$$

The design phase requires a Uniform distribution with a lower limit of 4 days and an upper limit of 6 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 4 \le x \le 6 \\ 0 & \text{for } x < 4 \text{ or } x > 6 \end{cases}.$$

The development phase requires a Uniform distribution with a lower limit of 10 days and an upper limit of 15days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 10 \le x \le 15 \\ 0 & \text{for } x < 10 \text{ or } x > 15 \end{cases}.$$

The Testing phase requires a Uniform distribution with a lower limit of 1 days and an upper limit of 3 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 1 \le x \le 3 \\ 0 & \text{for } x < 1 \text{ or } x > 3 \end{cases}.$$

The Maintenance phase requires a Uniform distribution with a lower limit of 1 days and an upper limit of 3 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 1 \le x \le 3 \\ 0 & \text{for } x < 1 \text{ or } x > 3 \end{cases}.$$

### 3.3.2: Medium Scale-Project:

The business analysis phase requires a Uniform distribution with a lower limit of 10 days and an upper limit of 15 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 10 \le x \le 15 \\ 0 & \text{for } x < 10 \text{ or } x > 15 \end{cases}.$$

The design phase requires a Uniform distribution with a lower limit of 16 days and an upper limit of 20 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 16 \le x \le 20 \\ 0 & \text{for } x < 16 \text{ or } x > 20 \end{cases}$$

The development phase requires a Uniform distribution with a lower limit of 30 days and an upper limit of 60 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 30 \le x \le 60 \\ 0 & \text{for } x < 30 \text{ or } x > 60 \end{cases}$$

The Testing phase requires a Uniform distribution with a lower limit of 10 days and an upper limit of 15 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 10 \le x \le 15 \\ 0 & \text{for } x < 10 \text{ or } x > 15 \end{cases}$$

The Maintenance phase requires a Uniform distribution with a lower limit of 7 days and an upper limit of 10 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 7 \le x \le 10 \\ 0 & \text{for } x < 7 \text{ or } x > 10 \end{cases}$$

### 3.3.3: Large Scale-Project:

The business analysis phase requires a Uniform distribution with a lower limit of 30 days and an upper limit of 40 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 30 \le x \le 40 \\ 0 & \text{for } x < 30 \text{ or } x > 40 \end{cases}$$

The design phase requires a Uniform distribution with a lower limit of 40 days and an upper limit of 60 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 40 \leq x \leq 60 \\ 0 & \text{for } x < 40 \text{ or } x > 60 \end{cases}$$

The development phase requires a Uniform distribution with a lower limit of 60 days and an upper limit of 150 days.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 60 \leq x \leq 150 \\ 0 & \text{for } x < 60 \text{ or } x > 150 \end{cases}$$

The Testing phase requires a Uniform distribution with a lower limit of 30 days and an upper limit of 40 days

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 30 \leq x \leq 40 \\ 0 & \text{for } x < 30 \text{ or } x > 40 \end{cases}$$

The Maintenance phase requires a Uniform distribution with a lower limit of 15 days and an upper limit of 30 days

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } 15 \leq x \leq 30 \\ 0 & \text{for } x < 15 \text{ or } x > 30 \end{cases}$$

## 4.1: The proposed simulation Model

The proposed model consists of many modelling elements like capture, release, task, file, counter and resources. For this model resources are the expert team members assigned on different phases of the Iterative/Incremental model. The simulator Simphony.NET tool is used with an Iterative/Incremental model to determine the optimized expert team members.

A project in Simphony.NET is made out of a collection of modeling elements linked to each other by logical relationships. The study shows a complete software product delivered in 5 increments of the Iterative/Incremental model. and for each increment of the simulation model, necessary and optimal resources can be determined.

The simulation model for different phases of Iterative/Incremental model for Small-Scale projects is shown as following:

## 4.2: Simulation model for Small-Scale projects

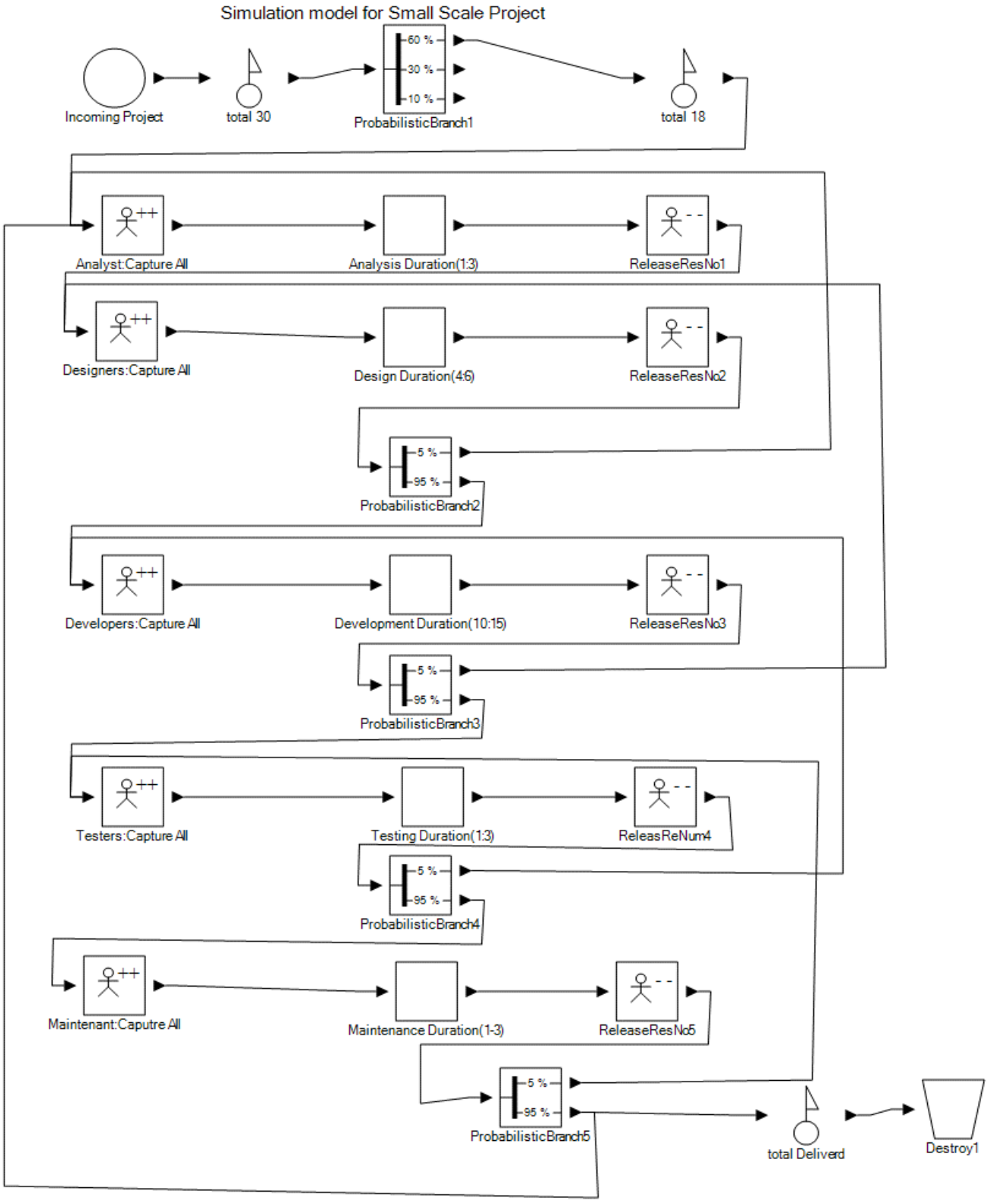


Fig4.2 Simulation for small-scale project.

At very beginning of the model a counter element counts the total number of projects received. At the end of the simulation model another counter counts the number of delivered projects.

There are total of 30 projects arrive randomly in a software firm every 20-60 day. A probabilistic branch element exists at beginning of the model with 3 core branches:

- branch 1 shows that 60% projects are of small scale.
- branch 2 shows that 30% projects are of medium scale.
- branch 3 shows that 10% projects are of large scale.

Additionally, several probability branch elements exist between the different tasks of the model whose purpose is to simulate the probability of delay that an Iterative/Incremental model's task might exhibit after completion.

The probability element has two branches:

- Branch 5 with Prob=0.05 denotes that 5% of the small-scale projects are subject to delay.
- branch 2 with Prob=0.95 denotes that 95% of the small-scale projects will not exhibit delay after the completion of every phase.

These branches simulate the recursive property of the Iterative/Incremental model to loop over the preceding task if an error was found in the current task.

The model starts with a new entity element which sets the number of incoming projects and a counter that counts the number of projects being received, and ends with another counter that counts the number of projects being delivered.

Each phase of the model is modelled with a capture element and a release element. The capture element links a specific resource to a specific task and the release modelling element releases the resources (team members) from the task element when it is completed. The task element content the duration of each phase.

## 4.3: Simulation model for Medium-Scale projects

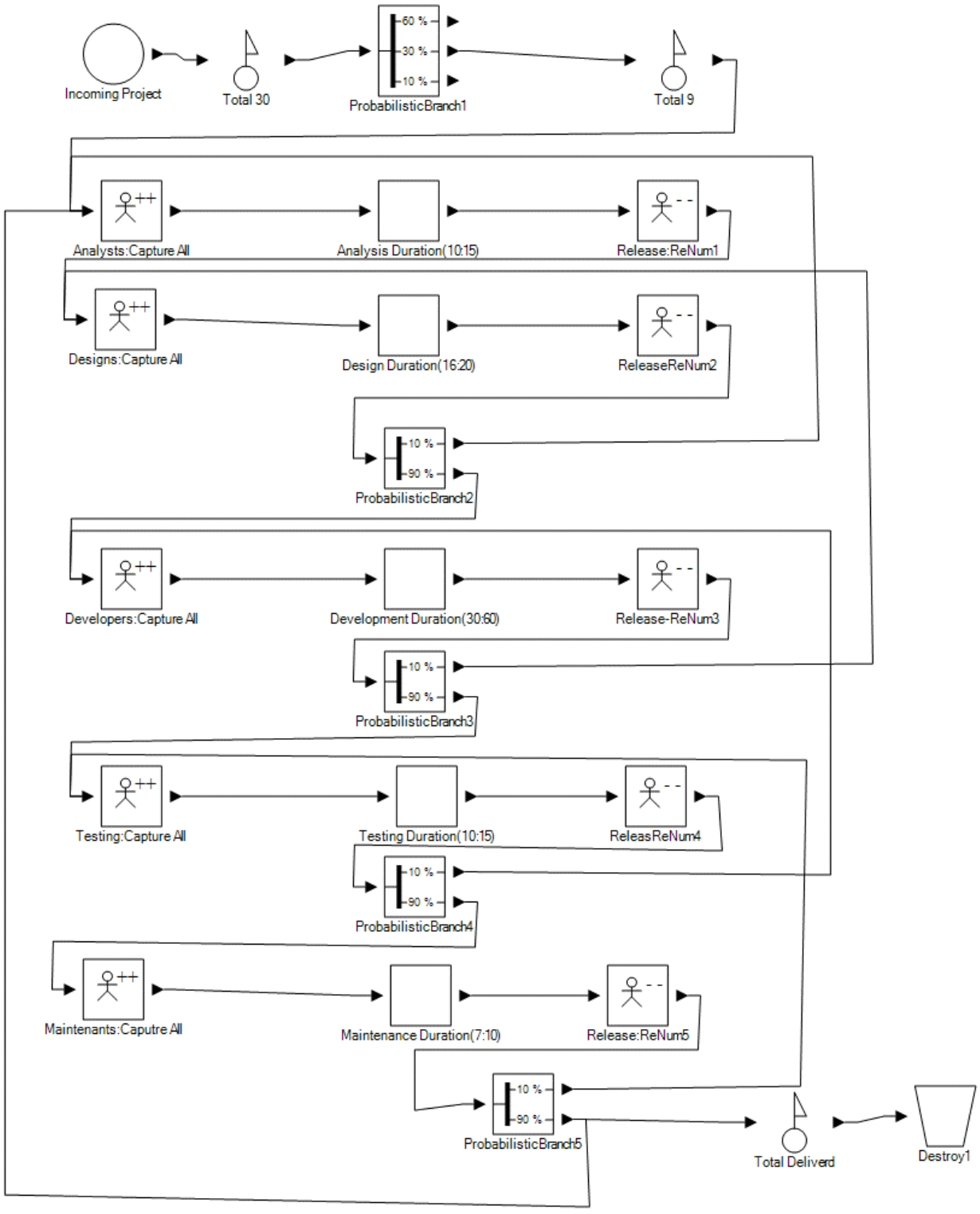


Fig4.3 Simulation for Medium scale project.

## 4.4: Simulation model for Large-Scale projects

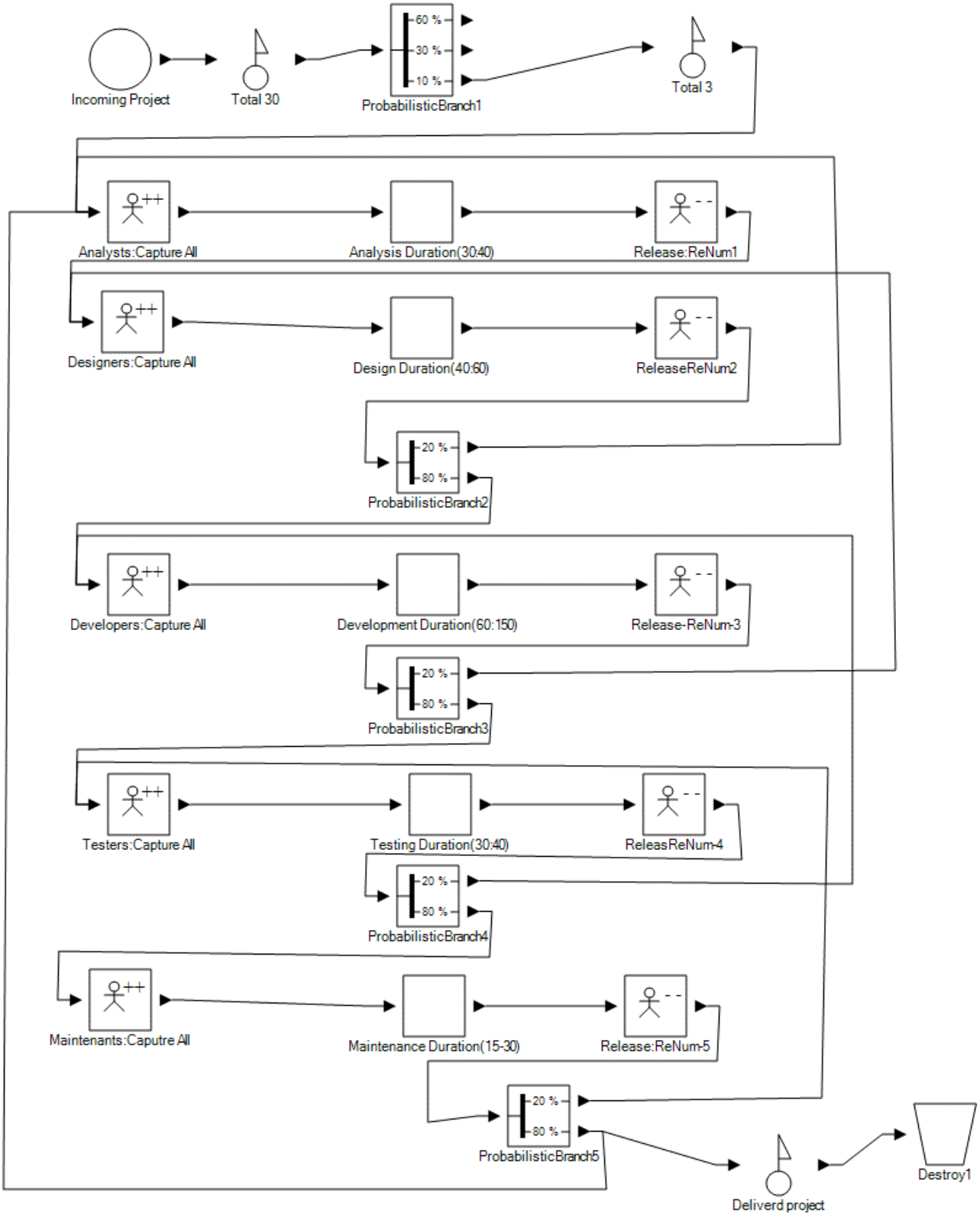


Fig4.4 Simulation for Large Scale project.

## 5.1: Result of simulation model

Using the simphony.NET environment, the simulator executed 5 times, for 3000 milliseconds with 30 incoming projects. Following shows the statistics obtained after running the model including the number of arrival projects and the average utilization for each phase, for different scale projects.

Table 5.41 Number of arrival projects to the simulation model

| Type of project delivered | Numbers of project | Percentage |
|---|---|---|
| Small Scale project | 18 | 60% |
| Medium scale project | 9 | 30% |
| Large Scale Project | 3 | 10% |

### 5.1.1: Resource utilization for Small-Scale project:

Table 5.42 Average utilization for small-scale projects

Resources

| Element Name | Average Utilization | Standard Deviation | Maximum Utilization | Current Utilization | Current Capacity |
|---|---|---|---|---|---|
| ResAnalyst | 6.7 % | 1.1 % | 7.9 % | 28.6 % | 7.000 |
| ResDesign | 17.3 % | 2.7 % | 20.1 % | 40.0 % | 7.000 |
| ResDeveloper | 28.7 % | 4.3 % | 32.1 % | 32.0 % | 10.000 |
| ResMaintenace | 4.4 % | 0.7 % | 4.9 % | 4.0 % | 5.000 |
| ResTester | 4.7 % | 0.7 % | 5.5 % | 8.0 % | 5.000 |

From the above table the average utilization for Analysts 6.7% with St. Deviation 1.1% and 17.3% for designers with St. deviation 2.7%, and 28.7% for developers with St. deviation 4.3% and 4.7% for testers with St. deviation 0.7% and 4.4% for main tenants with St. deviation 0.7%.

### 5.1.2: Resource utilization for Medium-Scale project:

Table 5.43 Average utilization for medium-scale projects

Resources

| Element Name | Average Utilization | Standard Deviation | Maximum Utilization | Current Utilization | Current Capacity |
|---|---|---|---|---|---|
| ResAnalyst | 20.7 % | 5.9 % | 31.1 % | 25.7 % | 7.000 |
| ResDesign | 30.7 % | 8.7 % | 44.8 % | 60.0 % | 7.000 |
| ResDeveloper | 46.2 % | 11.0 % | 60.0 % | 90.0 % | 10.000 |
| ResMaintenace | 9.7 % | 2.7 % | 13.2 % | 16.0 % | 5.000 |
| ResTester | 15.8 % | 3.4 % | 20.1 % | 24.0 % | 5.000 |

From the above table the average utilization for Analysts 20.7% with St. Deviation 5.9%, and 30.7% for designers with St. deviation 8.7% and 46.2% for developers with St. deviation 11.0% and 15.8% for testers with St. deviation 3.4% and 9.7% for main tenants with St. deviation 2.7%.

### 5.1.3: Resource utilization for Large-Scale project:

Table 5.44 Average utilization for large-scale projects

Resources

| Element Name | Average Utilization | Standard Deviation | Maximum Utilization | Current Utilization | Current Capacity |
|---|---|---|---|---|---|
| ResAnalyst | 32.3 % | 13.0 % | 50.0 % | 40.0 % | 7.000 |
| ResDesign | 46.4 % | 23.8 % | 77.2 % | 80.0 % | 7.000 |
| ResDeveloper | 47.0 % | 24.0 % | 79.8 % | 80.0 % | 10.000 |
| ResMaintenace | 8.6 % | 6.2 % | 19.1 % | 0.0 % | 5.000 |
| ResTester | 15.5 % | 11.6 % | 35.5 % | 20.0 % | 5.000 |

From the above table the average utilization for Analysts 32.3% with St. Deviation 13.0% and 46.4% for designers with St. deviation 23.8% ,and 47.0% for developers with St. deviation 24.0% and 15.5% for testers with St. deviation 11.6% and 8.6% for main tenants with St. deviation 6.2%.

The results obtained after running the simulation model clearly showed the assigned team size doesn't arise to the optimal situation, the number of arrival projects doesn't equal the number of delivered project, the loss of delivered projects as following:

Table 5.45 Numbers of arrival and delivered projects.

| Type of project | No. of Arrival project | No. of Delivered project | No. of Loss project |
|---|---|---|---|
| Small-Scale | 18 | 11 | 7 |
| Medium-Scale | 9 | 8 | 1 |
| Large-Scale | 3 | 3 | 0 |

After increase the team size for the critical phases for different scale project the model arises to the optimal status by assign the resources as following:

Table 5.46 the optimal numbers of Resources for different scale-project.

| Resources | Small-scale | Medium scale | Large scale |
|---|---|---|---|
| Analysts | 2 | 3 | 7 |
| Designers | 4 | 4 | 7 |
| Developers | 4 | 4 | 5 |
| Testers | 2 | 2 | 10 |
| Maintenance | 2 | 2 | 5 |

## 5.2: Model validation

After successfully running the simulation model for a different scale project, different graphs for each type of resources utilization is generated by using the simulator simphony.net 4.0 to validate the model, it's the most commonly used approach for operational validity. The cumulative probability on Y-axis relative to the utilization of various resources on X-axis is used to present average utilization of resources (Analysts, Designers, Developers, Testers and Maintenance).

### 5.2.1: Graphical Representation for the Small-Scale project:

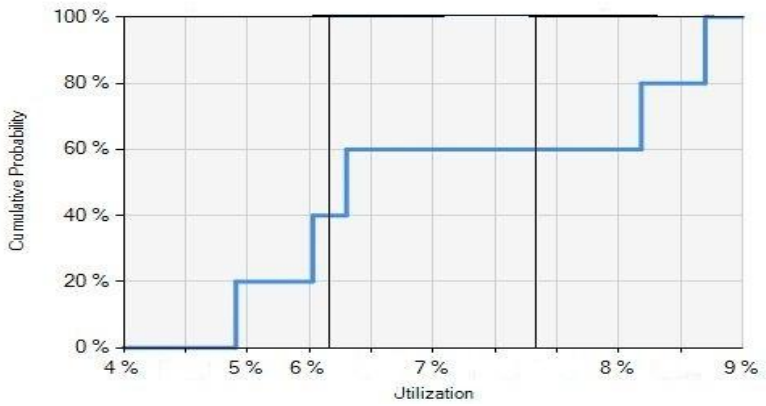


**Fig5.5 Utilization of Analysts**

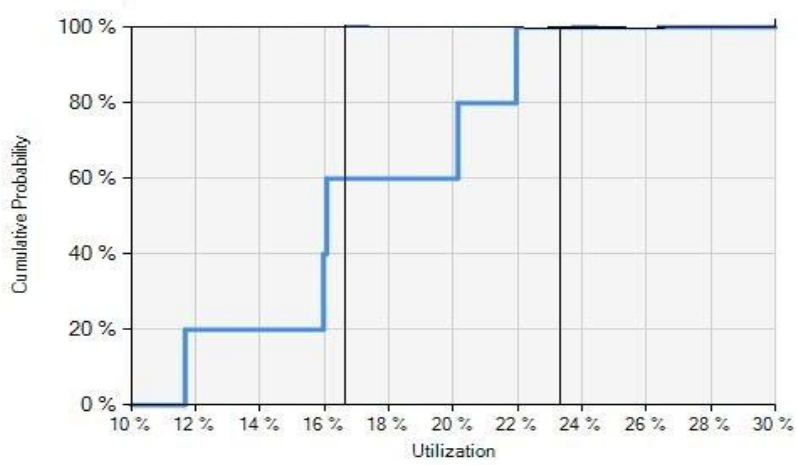


**Fig5.6 Utilization of Designers**

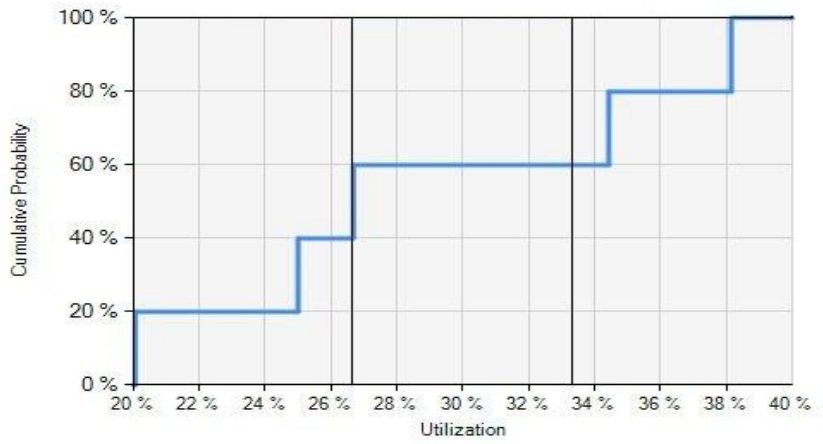


**Fig5.7 Utilization of Developers**

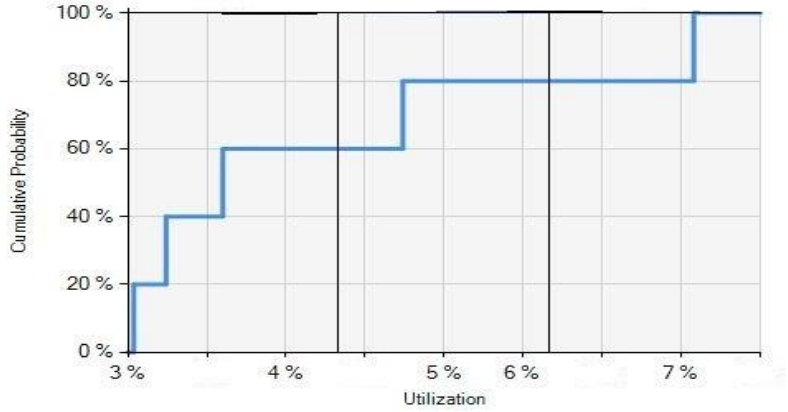


**Fig5.8 Utilization of Testers**

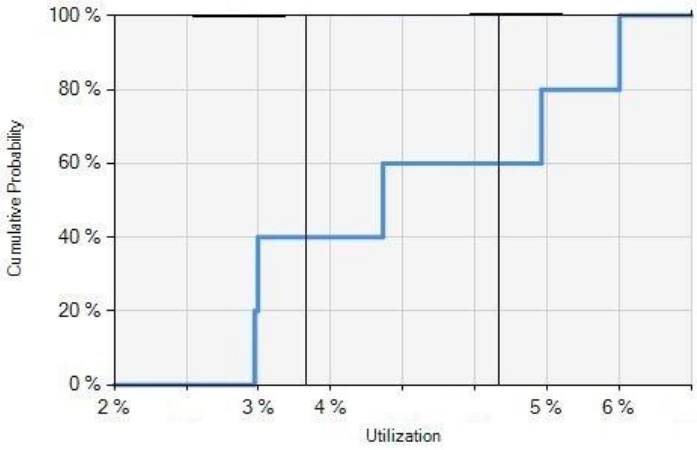


**Fig5.9 Utilization of Maintenance**

The above graphs show the validation for small scale project, when we mapping the rate of utilization for each phase for this graphs with the average utilization after assign the team in different scale of project we find there is similarity of result output and the different is not too much.

### 5.2.2: Graphical Representation for Medium-Scale Project:

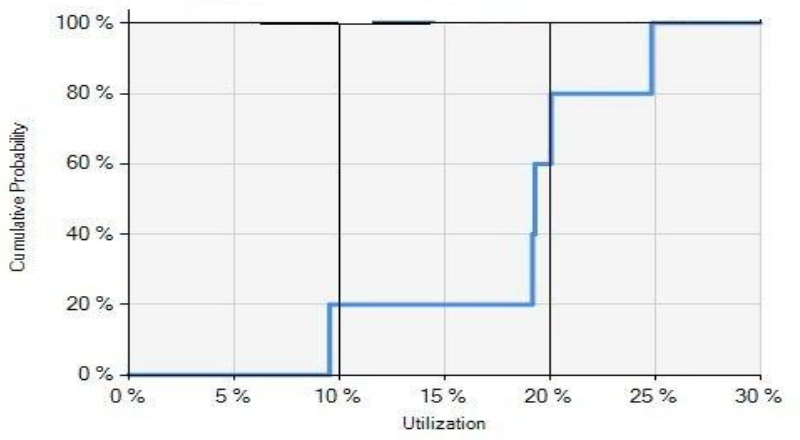


**Fig5.10 Utilization of Analysts**

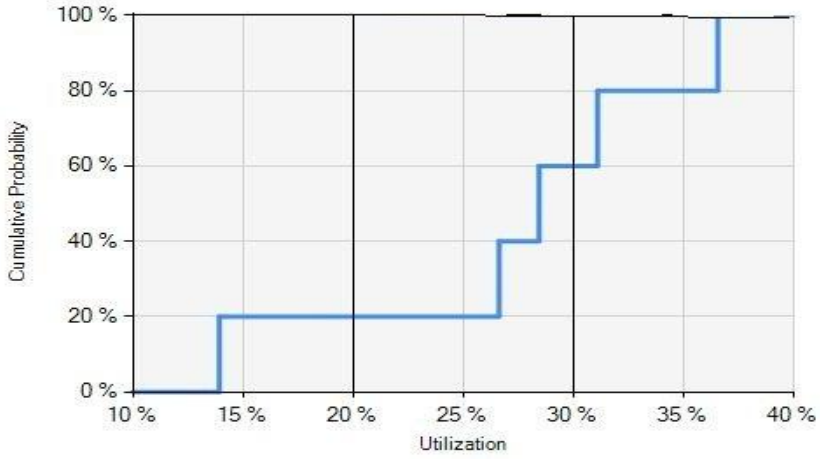


**Fig5.11 Utilization of Designers**

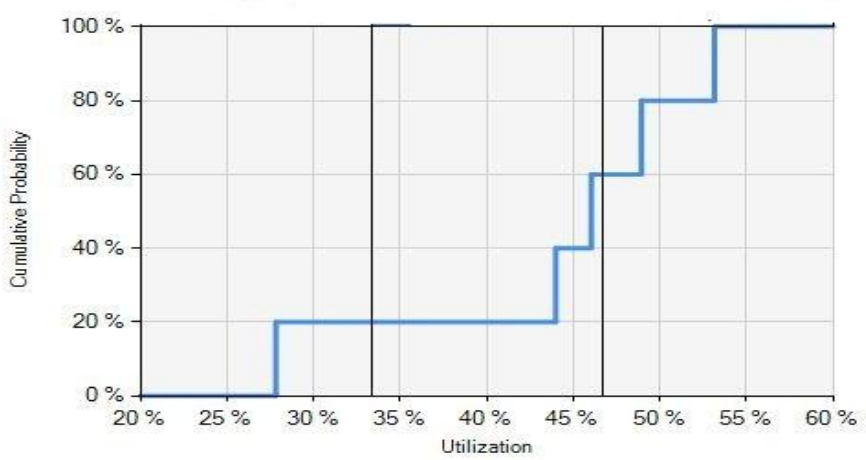


**Fig5.12 Utilization of Developers**

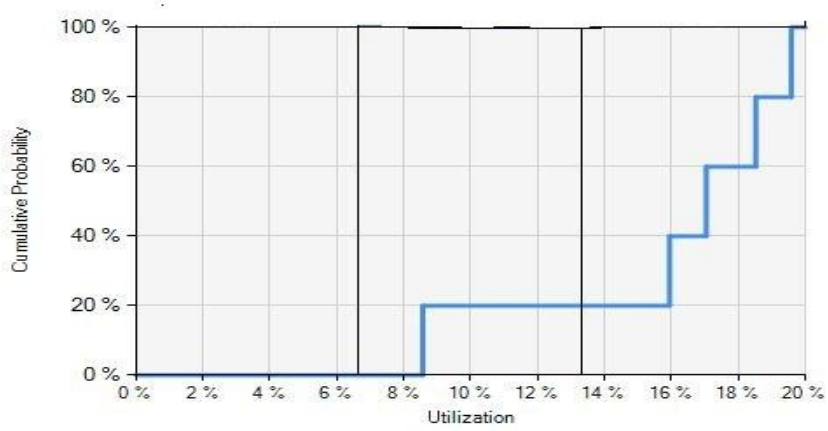


**Fig5.13 Utilization of Testers**

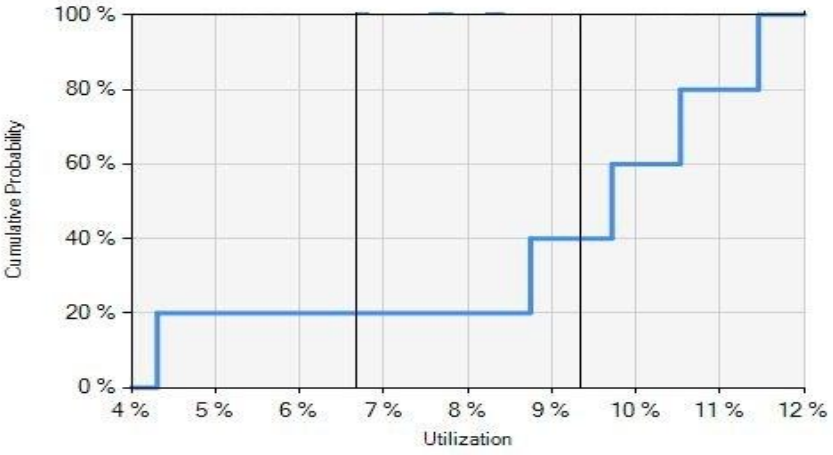


**Fig5.14 Utilization of Maintenance**

### 5.2.3: Graphical Representation for Large-Scale Projects:

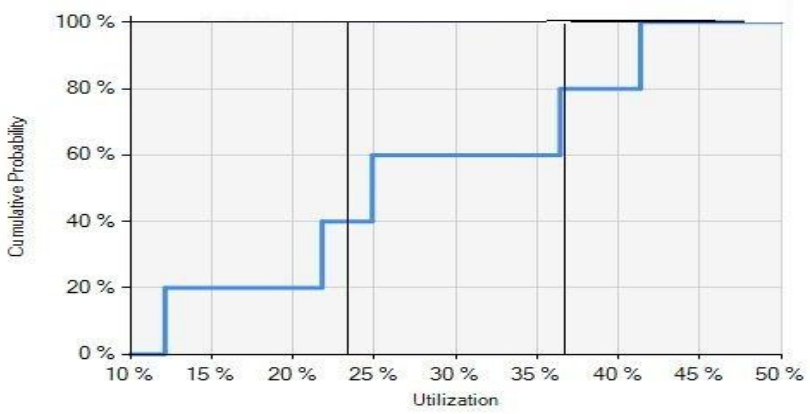


**Fig5.15 Utilization of Analysts**

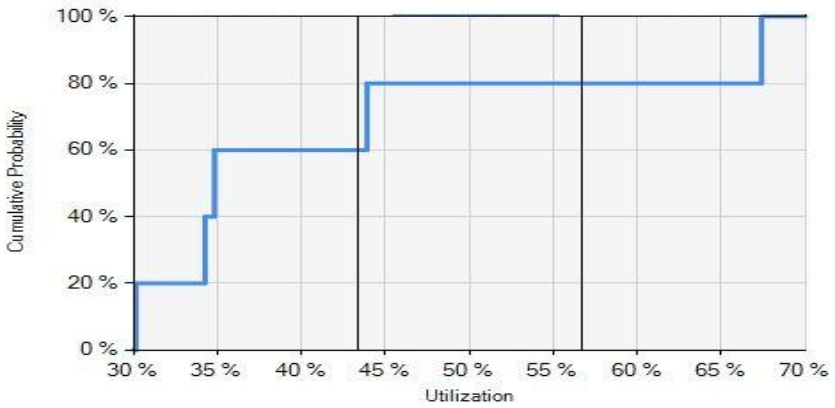


**Fig5.16 Utilization of Designers**

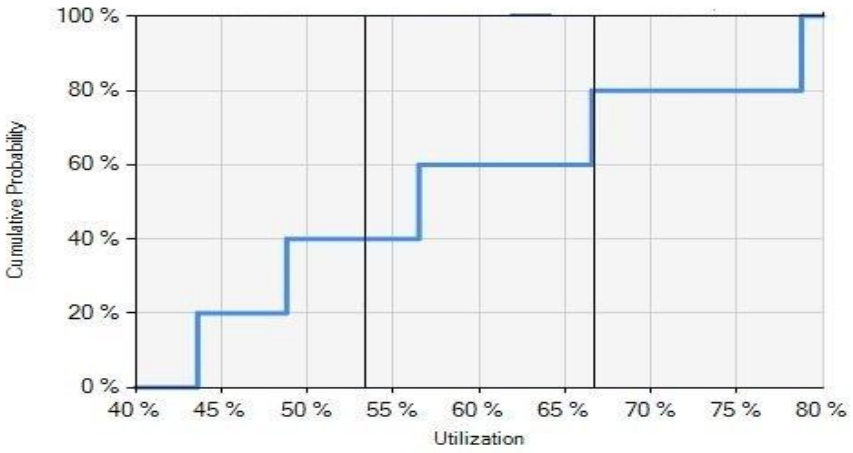


**Fig5.17 Utilization of Developers**

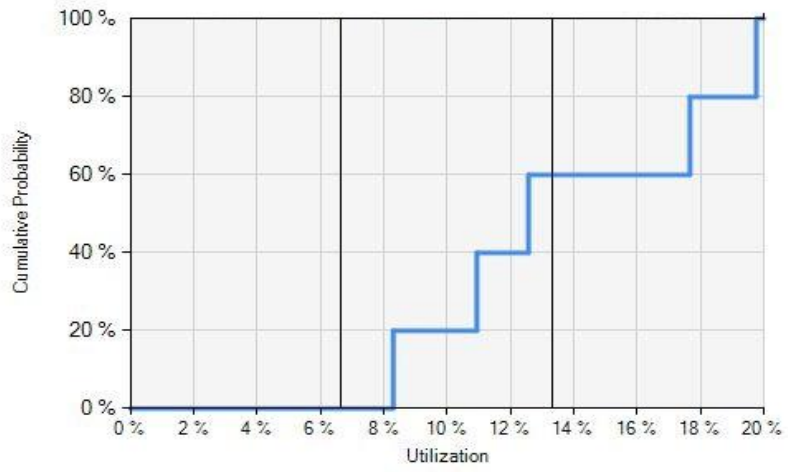


**Fig5.18 Utilization of Testers**

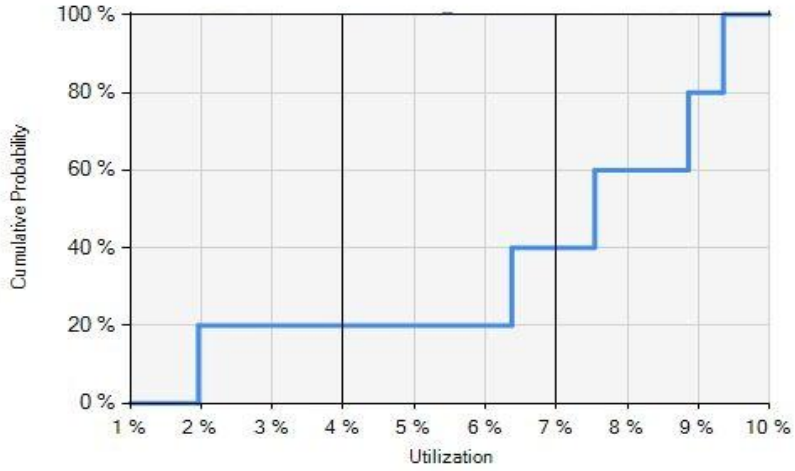


**Fig5.19 Utilization of Maintenance**

## 5.3: Result Discussion

the simulation result shows the model doesn't arise to the optimal situation to deliver all the incoming project according to the specific time and assigned resources.

When we increase the number of resources for small-scale project and medium-scale project that makes the simulation model arise to the optimal situation to deliver all arrival projects without any loss. Experiments shows the optimal number of resources that will make the company working with incoming project without bottleneck of arrival and delivery projects as the following:

Table 5.46 the optimal numbers of Resources for different scale-project.

| Required team member | Small scale | Medium Scale |
|---|---|---|
| Analysts | 2 | 4 |
| Designers | 4 | 4 |
| Developers | 4 | 4 |
| Tester | 2 | 2 |
| Maintenance | 2 | 2 |

## 6.1: Conclusions

The Research proposed a simulation model for simulating the famous software development methodology in Sudanese software development companies to enhance their practice by assign the resources (team members) intelligently. Its aim to assist the project managers to determining the optimal number of resources required for each phase to deliver the software projects on time.

## 6.2: Suggestion for Future work

The thesis proposed a simulation model and this model applied on the most famous software development life cycle (Iterative/Incremental model) in Sudanese software development companies.

This model simulates each scale of project independently (small, medium and large scale), As future work there is a need to integrate all this models in single model that can simulate all of them.

Also, other SDLCs models are to be simulated to allowing project managers to select the best among them to support their decision-making and planning needs.

## References:

**Abdulaziz, F,A.,2015**. A Software Project Management Innovation (SPM) Methodology: A Novel Method for Agile Software Development" Conference at Harvard, in Boston, USA.

**Aggarwal, N., Prakash, N. and Sofat, S., 2008**. Content Management System Effort Estimation Model based on Object Point Analysis. International Journal of Computer Science and Engineering.

**Bassil, Y., 2012.** A simulation model for the waterfall software development life cycle.

**Chopra, M. and Kapoor, N., 2016**. Choosing the Right Approach: Comparative Study of Software Process Models. International Journal.

**Chauhan ,M., J.,2016**. Rapid Revision Software Development Life Cycle Model (RR-SDLC Model) Based on Concept of Reusability of Software , International Journal of Advanced Research in Computer and Communication Engineering.

**Fogle, A. and Qu, Y., 2015.** An Extended Simulation Model for Managing Dynamic Changes in Software Development Projects. International Journal of Modeling and Optimization.

**Hurst, J., 2014.** Comparing Software Development Life Cycles. SANNS Software Security.

**Ian Sommerville,2010**. Software Engineering, Addison Wesley .

Kruchten, P., 2004. The rational unified process: an introduction. Addison-Wesley Professional.

**Kumar, N,Zadgaonkar, A.S. and Shukla, A., 2013**. Evolving a new software development life cycle model SDLC-2013 with client satisfaction. International **Journal of Soft Computing and Engineering (IJSCE).Lawanna, A., 2014** Simulation Techniques for Determining Numbers of Programmers in the Process of Software Testing.

**Munassar, N.M.A. and Govardhan, A., 2010**. A comparison between five models of software engineering. IJCSI International Journal of Computer Science Issues.

**Mujumdar, A., Masiwal, G. and Chawan, P.M., 2012**. Analysis of various software process models. International Journal of Engineering Research and Applications.

**Mishra, A. and Dubey, D., 2013.** A comparative study of different software development life cycle models in different scenarios. International Journal of Advance research in computer science and management studies.

**Qureshi, M.R.J. and Hussain, S.A., 2008.** An adaptive software development process model. Advances in Engineering Software.

**Suri, P.K. and Bhushan, B., 2007**. Simulator for Time Estimation of Software Development Process.

**Srinivasan, J. and Agila, R., 2014.** software development life cycle model incorporated with clemency brass.

**Thind,S. Karambir,2015.** A Simulation Model for the Spiral Software Development Life Cycle" International Journal of Innovative Research in Computer and Communication Engineering.