Faculty of Science, Technology and Engineering
La Trobe University

Knowledge Loading as a Factor in Software Project Planning and Estimation – A Consequence of KABASPP

Author: Shivraj Sabale
Student No. 14988185
Supervised by Assoc. Prof. Karl Reed

Course: Master of Software Engineering
Knowledge Loading as a Factor in Software Project Planning and Estimation – A Consequence of KABASPP
October 2006

Acknowledgement

I would like to take this opportunity to thank my supervisor Assoc. Prof. Karl Reed who has not only been an inspiration to this thesis but has been an active participant throughout. His valuable work has provided me with a firm foundation for exploring newer approaches. His advice and directions have proved helpful in canalizing my efforts.

It would have been much harder without his suggestions on writing as it made this survey comprehensive enough to cover major topics and comprehensible enough to cover most readers.

I would also like to acknowledge the backing provided by my Father, Mr. Sampatrao Sabale who has taught me look at a situation rationally and to be absolutely unbiased in the approach.

Table of Contents

Table of Figures

- Chapter 1 shall introduce the KABASPP theory. Knowledge an Acquisition based approach to Software Project Planning forms the basis for this project and has been borrowed from the original theory for reference.

- Chapter 2 shall deal with some basic Software Project Planning Fundamentals. It shall also look into the circumstances under which KABASPP was introduced. It shall further discuss some Current Software Project Planning practices.

- Chapter 3 shall deal with Project Planning issues from Traditional Engineering domains such as Construction and Mechanical Engineering.

- Chapter 4 shall investigate the pervasiveness of KABASPP Domain structure in the current Software Engineering Scenario.

- Chapter 5 will be an extension to Chapter 2 and investigate the pervasiveness of KABASPP process in the current Software Project Planning Techniques.

- Chapter 6 shall extend Chapter 3 and deduce Project Planning Strategies which recognize the criticality of knowledge acquisition. It shall also look at some classic problems in this field and try to relate them with the Software Projects to study the effect.

- Chapter 7 shall tackle one of the chosen units i.e. Manpower Buildup and Team Formation, in software engineering to study the effect of Knowledge Acquisition on it.

- Chapter 8 shall finally contain the Proposal.

- Chapter 9 contains the conclusion for the paper.

Introduction

> *The basic capital resource, the fundamental investment, but also the cost center of a developed economy, is the knowledge worker who puts to work what he has learned in systematic education, that is, concepts, ideas, theories, rather than the man who puts to work manual skill or muscle.*
>
> *Peter F. Drucker [PETE 1]*

A lot has been said and done on Software Project Planning and utilization of resources like the staff and developers to achieve the deadlines. Many models have been introduced to study the staff loading patterns and the amount of effort required during the process.
The underlying assumption in most of these studies has been that the developer or team member is at the maximum knowledge or skill maturity level. General project management wisdom contradicts this assumption and compels us to have a re-look at the factors affecting delays in the planned schedule. The "Knowledge/Skill-factor" needs to be considered while analyzing such situations.

A 5 layer model named KABASPP [Knowledge an acquisition based approach to software project planning] was introduced in 1991 by Assoc. Prof Karl Reed.
This paper lays emphasis on the usage of Skill or "knowledge" available at the beginning of the software project for project planning. It also argues that the skills needed to realize the tasks in the project and the "knowledge" requirements should be the driving factor behind the project plan rather than a generalized "Software Process".

The current Software Development Scenario has seen an alarming rise in delays in the schedules and project over-runs. We shall investigate through this paper if these actually are project failures or Estimation mistakes in which case a concept of "Knowledge Loading' shall be introduced to elaborate the competency deficit in the human resource.

This paper shall investigate the pervasiveness of KABASPP in Software Engineering.
KABASPP as a process and KABASPP as Knowledge Domains need to be studied in order to achieve the final aim of correct knowledge acquisition and to fill up the deficit.
We shall thus be looking for implicit or explicit presence of KABASPP artifacts in the current practices. This shall encompass the entire Software Engineering domain including Testing, Component Based Development and Lifecycle models.
To ensure that the findings get a stronger foundation, we shall cross reference these with the common practices in the Traditional Engineering Domains such as the Construction Engineering.
Finally, we shall deal with the specific topic of Manpower Buildup and Team formation and ways to ensure that a knowledge based approach is used right from the inception of the project/
We shall look at some of the most common, current methods and strategies used for project planning and manpower buildup and study the possibility of the usage of KABASPP for solving some of the problems faced. We shall also introduce the "knowledge factor" and investigate its impact on the overall planning process.

Literature Review

Software project planning and estimation has been one of the most researched issues in software engineering. After all the research that has been done, we can claim that no process is perfect or the "right" way of developing a project. But there is no doubt that there will always be some room for improvement.

It is important to analyze the source for major budget spending for businesses dependent on IT. It is evident nowadays that software costs have overridden the underlying system hardware costs. It is thus important to study various techniques which will help us in improving the current software development approaches and reduce the development cost for software applications.

With the help of KABASPP as an underlying theory we shall strive to achieve such a result. We have made some changes to the Literature Survey since the initial Literature Survey. As we went along, we had to revise our plan for research as some topics showed much more promise than the others. The fundamental objective of investigating the Pervasiveness of KABASPP and showing the impact of Knowledge Acquisition on Software Project Planning still remains.

Some major changes have been made to the Literature review from the last time considering some new venues of interest. The focus of this paper has now been adjusted over the application of KABASPP over one Software Engineering activity and to suggest some new approaches for the same. A basic study of some project planning models has also been included to understand the logic behind such planning decisions. These help us in justifying the suggestions made about the change in approach required.

Chapter 1

KABASPP [Knowledge an acquisition based approach to software Project planning]

A 5 layer model named KABASPP [Knowledge an acquisition based approach to software project planning] was introduced in 1991 by Assoc. Prof Karl Reed.
This paper argues that relative levels of skills or "knowledge" available at the beginning of a software project, and the skills needed to perform the tasks constituting the project rather than some pre-ordained "software process" model should be used to generate a Project Plan. The paper identifies and describes a series of Knowledge or Skill Domains which can be used to develop a Software Project Plan.

The task of creating an artifact can be divided into two sub tasks.

1- To acquire knowledge and skill required to create that artifact.
2- To actually create the artifact. [REE 2]

Thus it is implied that we need to posses the necessary skill and knowledge at the point of creating the software. This is a knowledge or skill acquisition process of a more general kind, involving training or problem analysis as necessary.
An examination of the steps taken during, and of the techniques and tools used, in a software project, suggests that there are a small number of relatively distinct knowledge domains.

The KABASPP Domains have been based on classification of knowledge required to complete a project.

The five domains are

a) *Application Domain:* the physical laws, organizational structures, procedures etc which govern the software artifact to be produced.

b) *Application Solution Domain:* the collection of machine executable descriptions (algorithms) which make it possible to realize the application as software.

c) *Development Environment Domain:* the complete set of tools, techniques and methods used to both develop elements of the application domain and Application solution domain, and to realize them as software.

d) *Run Time Environment Domain:* the set of characteristics, relating to the particular machine environment under which the software must run.

e) *Managerial Domain:* the techniques necessary to plan estimate and manage the project.

[REE 2]

KABASPP Domains and Components are not equivalence classes. However, they provide a clear indication of the categories of skill in the use of or knowledge about subsystems, techniques or tools required in each case. Following are the examples of each kind of knowledge domain. It also exemplifies the disciplines responsible for learning or mastering a particular knowledge domain.

1.1 Applications Domain

Examples

- Acceleration characteristic of a train
- Organizational structure of business
- Rules for issuing air-line tickets or degrees
- Procedures for organizing work flow
- Procedures for design of pressure vessel etc.

- Discipline Responsible
- Commercial systems analysis
- Engineering
- Engineering Design Analysis
- "Knowledge" Engineering

1.2 Application Solution Domain

Examples

- Algorithms for searching lists
- Approximate method for calculating acceleration of train
- Procedure for allocating seats on a vehicle given multiple access
- Path optimization's procedure for routing of information
- Algorithm for rotating graphic images
- Procedure for recovering disc-space
- Sort procedures

Discipline Responsible

- Computer Science
- Graphics
- Artificial Intelligence Software Engineering etc.

[REE 2]

1.3 Development Environment Domain

Examples

- Programming languages
- Methodologies {JSD, SD, Modular Design}
- Tools - CASE, other development aids, Test tools
- O.S. and control language - shell, MCP, DOS, JCL etc
- Utilities - loaders, file manipulation, editors, and configuration managers.
- File structure, Database management systems.

Disciplines Responsible

- Computer Science

- Software Engineering.

1.4 Run-Time Domain

Examples

- Operating Systems interfaces
- Database management systems calls
- Instruction set, external interfaces
- Resource constraints (i.e. profile of available cpu time, input/output,
- Memory for the system).
- Response time
- Device peculiarities
- Hardware Reliability vs. Design goal

Disciplines Responsible

- Computer Science
- Computer Engineering
- Software Engineering.

[REE 2]

1.5 Project Management Domain

Examples

- Estimating
- Project Planning
- Project Organization
- Resource acquisition
- Selection of people and development and run-time domains!

Disciplines Responsible

- Commercial EDP and Software Engineering and KABA.

A brief guideline has been provided to use such a framework.

a – To asses the requirements in terms of the KABASPP domains i.e. to find the requirement of respective knowledge and skill.

b - Now we have to asses the schedule or the deadlines and identify the times at which we would be requiring these components.

c - Look at the current status and asses where we stand in terms of knowledge and skills.

d - Devise a project plan to acquire the needed knowledge and skill by the time it is needed which will ensure timely completion of the project.

[REE 2]

Chapter 2

Traditional Software Project Planning

The objective of this chapter is to understand the thinking behind current Software Project Planning techniques. We shall also see how this approach led to the introduction of a theory like KABASPP and other non-rigid processes.

Before looking at some of the ongoing research in the non-rigid process engineering domain, let us look at some basic fundamentals of software project planning and estimation. Project planning has been around for ages in various Engineering and non-Engineering disciplines. Introduction to project planning has not seen many years pass by and thus we can summarize some of the popular techniques here.

Software process is one such entity which needs to be studied in detail to understand the procedure of converting the raw material (read knowledge) into the final software product. It also enlightens the various management control procedures and outlines the workflow.

There are many such process frameworks and process models which are existent in the industry. One such process model is the waterfall model. Originally introduced by Winston Royce, the waterfall model included a provision for “feedback loops”. These were although ignored in the common usage and this model is now followed as a strictly

linear model. It followed the process of sequential flow of project activities like requirement gathering, design, coding and testing. The model can be seen in figure 14. [ROY 4]

Many process models followed including the Incremental Model which followed the steps in the waterfall model in an incremental fashion. The product was developed in smaller increments and each increment development followed a sequential model.

Another model which gained popularity was the Spiral model which was proposed by Boehm was an evolutionary model. Other models like the RAD and Prototyping were also followed. A detailed study of these models will expose some inherent fundamental principles on the basis of which the tasks in these models are organized.

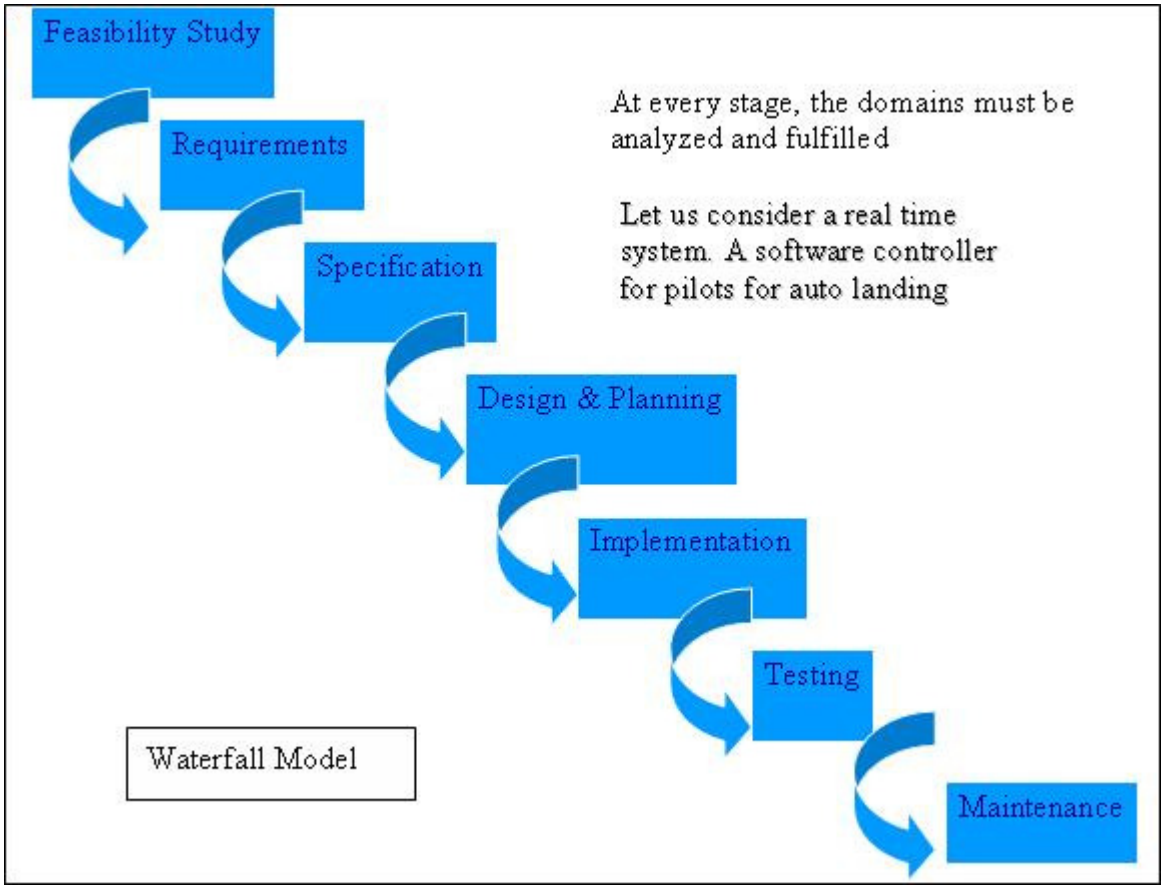

Figure 1: Royce's Waterfall Model

These tasks have a high level of dependency on knowledge acquisition and availability of the required skills. Let us study the Waterfall Model and see if we can identify some knowledge domains in the tasks as laid down in KABASPP.

2.1 Feasibility Study

- *Application domain* (Domain analysis) must be done in detail to understand the implications of the s/w and its criticality.
- *Application solution domain* for studying the possible Algorithms and methods for performing graphical/ path optimization calculations.
- *Dev-Environment domain* to see the availability of the tools/techniques/platforms required.
- *Run-Time Environment* domain to study the possible OS interfaces, h/w constraints, response times
- *Management domain* considerations like staffing/costing/scheduling bottlenecks.

2.2 Requirement gathering

- Requirements are said to be "good" only if they have been done by a person with enough *application domain* experience.
- The 2 domains *Development environment/Runtime domain* are would mainly deal with the h/w, s/w requirements
- *Management domain* would focus on the other resource, staff.
- Management domain is also essential for scheduling, cost and effort requirements.

2.3 Planning-Design-Implementation

- Goes without saying that the following domains need to be worked upon
- *Application solution* for algorithms and possible processes
- *Development environment domain* to ensure the timely availability of tools and techniques to carry out the development
- *Run time environment* to ensure that the s/w runs well in the environment it is to be used.

2.4 Testing

- *Application domain* knowledge is necessary for designing the test cases and deciding the testing strategy
- *Application solution domain* needs to be studied to see what algorithms were followed and the possible flaws.
- *Development environment* for testing tools.

- *Run time environment* to mimic the user's workspace
- *Management domain* to analyze the errors and perform quality management.

2.5 Maintenance

- *Run time environment domain* to understand the actual run environment

This study has lead to some major findings and also lead on to the initiation for KABASPP. During the late 80s a large amount of sequencing problems was uncovered in models such as the ones described above. A popular belief that if Phase Pj follows Phase Pi in the development lifecycle then it was quite possible to move from Pi to Pj. The question was how it will react to a situation where we tried to traverse backwards from Pj to Pi. Thus D (Pi) = D (Pj), but D (Pj) ≠D (Pi) [REE 3]

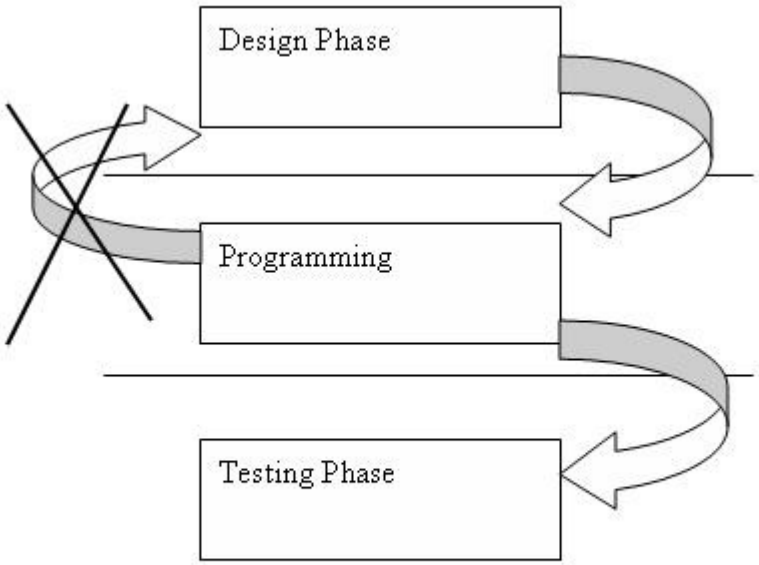

Figure 2: Unidirectional Flow of Project Development

This means in a waterfall model, it is possible to traverse forwards but not so backwards. There were a lot of implications to this which further lead to models such as Spiral and Prototypes.

Let us now have a look at some of the current software project planning concepts.

Shari Lawrence Pfleeger in her textbook on Software Engineering has mentioned some interesting workflows for Software Project Planning. The sequence of activities for a project plan according to her are as follows,

- Understand the requirements.
- List all the deliverables (Including documents like online help tools)
- Analysis of the above mentioned list and to designate the key milestones. Key milestones are generally % completion of the deliverables.
- Analyze the milestones and designate detailed activities.
- A phase wise schedule using the Work Breakdown Structure (WBS).

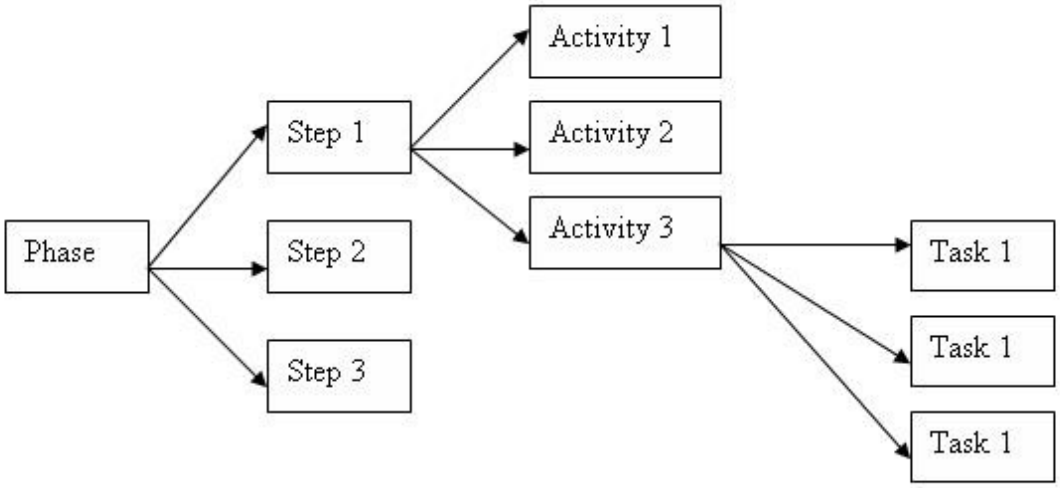

Figure 3: Work Break-Down Structure for a Software Project

- Assign Time and Effort for each task
- Assign Project Roles
- Estimation for the project costs
- Estimation for effort and phase wise effort calculations
- Finally Project Plan [FLE 5]

It is interesting to know that most of the estimations mentioned here, in reality, are not based on the capability and knowledge available but on instincts and previous project experiences. Such estimation, although helpful, will not analyze the current situation and will take the skill of the developers for granted. What we will be looking at is not to introduce another method of estimation or planning but a serious consideration to the knowledge and skill factors.

Ian Sommerville on the other hand has a similar approach, but somewhat conservative, towards Software Project Planning. He has outlined the following guidelines for project planning.

- Establish Project constraints.
- Initial assessment of Project Parameters. (Structure, Size and Distribution of functions)
- Define Milestones and Deliverables.
- LOOP TILL END
  a- Draw up a project schedule
  b- Commence initial activities and tasks
  c- Review the progress
  d- Revise Estimates and Schedules
  e- Update Project parameters
  f- Re-negotiate project constraints and deliverables
  g- END IF [SOM 6]

This can be looked at as a more reserved and a low-risk approach towards project planning. The important point to note here would be the revision of project constraints and re-negotiation of the deliverables and the schedule. This emphasizes the fact that project planning needs to be done promptly on the basis of available knowledge and skills. It also needs to be dynamic and flexible for changes.

It is quite conspicuous that most of the activities in this planning program are dependant on knowledge acquisition and availability of skills. The loop follows a spiral model and

keeps on updating the status of the deliverables. This helps in staying close to the target schedule and achieving the deadlines.

We also need to look at some pointers provided by Pressman for Project planning.

- Understand the scope (Max, Min) for the project. This will ensure that unwanted requirements are not fulfilled
- Involve the customer in planning activities to have a better understanding of the requirements and giving the customer some confidence
- Recognize that planning itself is an iterative process. It cannot be a static process
- Use the experience of employees you would know best about the product. Choose those employees who would have a great deal of application domain knowledge.
- Consider risk as a factor in planning
- Estimate based on what you know. This is very important for creating a realistic schedule
- Be realistic with the feasibility
- Adjust granularity as you proceed. Do not plan for details too early. They may have to be changed
- Define clear plans for ensuring quality
- Define clear procedures for SCM (Software Configuration management)
- Keep monitoring and revise plans.

It is quite clear by now that most of the emphasis in project planning principles is laid on keeping the procedure dynamic. This allows a room for revising the plans and also ensures that the project runs on track. We will look at this issue later in the paper when we deal with concepts like Extreme Programming[i] which deal with this issue with forced knowledge acquisition.

There has been immense research done in the field of non-rigid process engineering.

[i] Please refer to http:\\www.extremeprogramming.org\

A major shift from rigid process can be seen to using frameworks which help in project planning. Since most of the scenarios and developing techniques have changed over the period, we can see a level of incompatibility introduced in the process followed and process on paper [PRE 7].

Chapter 3

Use of Knowledge acquisition Traditional Engineering Domains

Project planning has been existent since centuries. Disciplines like Construction Engineering, Mechanical Engineering and Chemical Engineering have caught the interest of mankind since ages. Over the years of development of marvelous architectures and machines, these disciplines have matured to a stage where the craft and art of developing the final product is quite rigid. Compared to such immense history of success and failures, Software Engineering seems like a newborn. It has not been more than thirty years that the concept of engineering software has caught up. We intend to study many such cross domain examples and see how they relate to the software project planning. We shall go through some examples of successful project planning activities and some problems faced by this industry. We shall thus look at various issues relating to the knowledge factor.

Some of the basic fundamentals followed during any construction project as laid down by Richard Clough and Glenn Sears are as follows.

1- Design-then-Construct – Actual construction work should not be started unless the Chief Architect approves the design. Many big projects see a team of designers work on the blueprint of the building and then the Chief Architect needs to approve only after checking all the dependencies. This requires a large amount of domain knowledge and foresight. It can be gained with experience on live construction projects.

2- Planning activity along with the maintenance includes the "Useful Life" of the system. This factor will be dependant on the durability of the materials used and the overall structure.
3- There are three basic competencies/qualities that a construction site manager should posses. A) Technical Expertise & Experience. B) Availability of expertise in terms of people in specific management techniques, (Plan, Schedule, Computer Support). C) Personality Issues, (should be able to handle workers with lesser expertise or training)
4- Project planning must be done by people with experience and who are thorough with the field work
5- Success of a plan is dependant on the involvement of people who will implement the plan.
6- The plan should include personnel from each department of the organization to ensure that each and every restraint is satisfied. Estimators, Project Managers, Site Superintendents, and Field Engineers are some of the people who are a must while devising the plan.
7- Plan represents the best thinking available when it was conceived. Thus with changing conditions, plan should be dynamic.

One of the major techniques used for project planning is the Job Logic. A series of job activities are identified and these activities are prioritized. A table is generally used for sorting out dependencies and resource smoothing. Techniques such as Precedence diagram or Critical Path Method is used for the same.

Categorization of Job Activities

- by area of responsibility (Contracts)
- by work as distinguished by craft or crew
- by equipment required and usage
- by raw materials used
- by structural elements
- by location
- by payment methods

The following table will clearly define all the dependencies between various operations based on the requirement of human resource. Such tables have been used for ages to ensure that the manpower is available at the right time and ready for deployment on the job.

Competency\ Operation	Excavation	Pour Footing	Rub Concrete	Guard Rails	Saw Joint	Paint
Carpenters				√	√	
Laborers	√	√	√		√	
Equipment operators	√		√			
Oilers				√	√	
Cement Masons		√	√			
Truck Drivers	√	√				

Table 1: Availability Matrix

Such tables help in managing People, Material and Money. [RIC 8]

Another example that can be quoted for understanding the defining role of knowledge for Project Planning is the Automation of the Irrigation Water Delivery System project. This project was undertaken by the Irrigation and Drainage division of the American Society of Civil Engineers. The basic aim of the project was to devise an automation system which can be configured for various characteristics of water flow. The need and value of dependable and flexible water supply was the driving force for such an irrigation project.

We shall have a look at an algorithm which considers the duration and rate of water supply. These two factors are the decision points in the following algorithm.

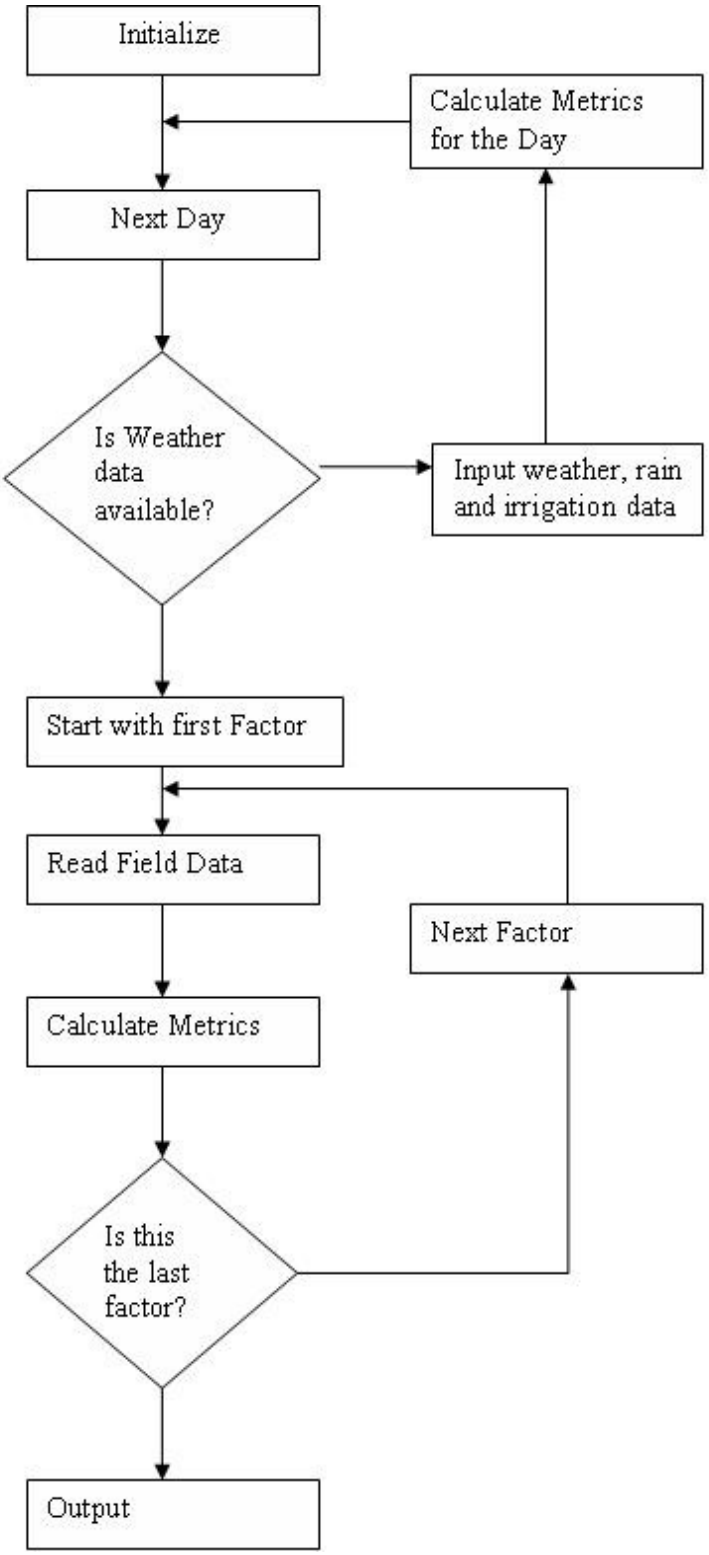

Figure 4: Algorithm for Calculating Weather Metrics

It is important to not that at each decision juncture; some action is being taken in the background on a daily basis like an increase or decrease in the water pressure levels or rate at which the water is discharged. [ZIM 9]

This shows the dynamicity in the simplest of the algorithms in the Traditional Engineering domain. It is high time that we also deduce some techniques from these domains.

Research

Chapter 4

Pervasiveness of KABASPP Domains

Let us look at some major research works which have acknowledged the importance of knowledge and are related to the acquisition process. We shall follow a disciplined approach of looking at various issues in order to find implicit or explicit implications towards the importance of knowledge acquisition and usage of the Knowledge domains as defined in KABASPP. As stated earlier, a positive outcome for the presence of KABASPP will reinforce the importance of recognition of knowledge acquisition as a important factor in Software Project Planning.

The major target in this section is also to investigate the presence of various domains defined in KABASPP which have been implicitly or explicitly used in various researches. This will help us in understanding the importance of classification of knowledge and its importance while planning any software project.

There are immense numbers of projects which we can grade as medium sized or “small” projects. Such projects have little or no documentation supporting their development and the process followed. These projects are generally in the range of 100 to 500 function points. E.g. a plug-in developed for major software companies.
So the only way to test such programs (without any specifications),where code itself is the complete documentation, is to study the patterns in which the programmers are more prone to making errors and to study the domain information to find modules which are more error prone. This clearly emphasizes the importance of knowing the application domain before any testing starts.

Let us take an example where we have to find the misuses of data types in a particular application which solves algebraic equation. Thus it is imperative to know how these equations work. This shows the need for application domain knowledge. If we were to insert some type checking using a small algorithm which solves the equation then we are in need to the Application Solution domain knowledge and skill before starting it.
Absence of such skills or knowledge at the time of development will result in a delay and the schedule would be disturbed. [HOW 10]

A large amount of research has been dedicated towards identifying the number of factors influencing how processes were tailored. There are factors which affect the decision of managers in which they choose a particular instantiation technique on the following factors. Let us see how they map to the requirement for various domain skills.
Thus before using any kind of process model, it was imperative to tailor it according to our needs as each project is unique in its own way. Thus we should consider the following before tailoring

- Culture
- geography
- Customer population
- Politics
- Size of system
- Safety issues
- Requirement analysis

[NAZ 11]

Most of the above mentioned issues have been included in the Management Domain in KABASPP. To have information on Culture, geography, Customer population, politics will be considered to be managerial issues wherein the prior knowledge of these issues will greatly facilitate the smooth progress of the project. So it is understood that acquiring such knowledge before starting the project is imperative.

Another survey was conducted by the Datamation group on the similar lines.

57 software projects were studied and a survey was carried out with the project managers. Most of these projects ranged from 20 staff members in size to a large of 3250 members. An average standing was a project with around “392” project members.

Most of the questions focused on the amount of rework required during the complete development cycle.

It was found out that 26% of specification rewrite was required on an average. The reasons that were listed were,

a- Errors, ambiguities (38%)

b- Changes in the requirements (38%)

c- Better understanding of the *Application Domain* (22%).[LEH 12]

This is an explicit proof that had the application domain knowledge was absolutely necessary for writing the specifications to avoid re-writing the specs.

Another question that can be raised is about the changes in the requirements. Changes generally occur in the requirements from the client side and late in the project. One of the major reasons cited for this is that the customer or client is himself unsure of what his requirements are and only after the realization of some percentage of the system would he start making up his mind. This situation can be avoided if the manager with some previous experience understands the real requirements of the user apart from what he has stated and tries to establish them early in the lifecycle.

We can find many such cases where lack of experience of the manager alone has resulted in poor outcomes. Inexperienced software managers (in terms of knowledge and skills to handle *application domain* data and *management domain* principles) often fail to recognize and expose early software problems.

They fail to recognize the hardware or specification instability and the impact of inexperienced personnel and mistakenly think that they can resolve them over time. But the Cleanroom Process forces early problem into the open by studying them and the reasons behind them. This gives the management an opportunity to solve such problems. [MIL 13]

Cleanroom Process has been specially devised to tackle projects which have very low margin for errors and follow a very strict quality control procedure.

It is noticed that they have reiterated the importance of knowledge acquisition in the application and management domains before even starting the process. This calls for an interesting discussion which can lead to serious implications. Is it possible to re-arrange the phases in software development in such a manner that project management activities can be started before their scheduled time? We have seen time and again that activities like software testing need a lot of groundwork before actually starting them. But the question lies that can testing be done even before the designing or coding is done. We can actually study projects where test cases can be devised even before the design or development has materialized. Many system level test cases can be devised on the basis of requirements and specifications alone. This could open various options for different approaches towards Software Project Planning. We would like to emphasize on the fact that a lot of work can be done in making possible the performance of two or more activities in parallel which are dependant on the same knowledge base.

This issue is further discussed in the proposal section.

If the project management is weak and relies heavily on the technical team to evaluate all the technical outputs, it must rely on the same unit which is introducing the errors.
If this unit itself is technically weak then so will be its judgment. Assigning these to different teams would mean a learning curve introduction in the team. Thus if we are to improve the quality of the reviews conducted by the same unit then we have to enable them to review impartially by and introducing professionalism in the process. They should thus be equipped with enough Development domain and application solution domain expertise. One common fault is to produce too much detail at the initial planning stage. You should be stop when you have a sufficient description of the activity to provide a clear instruction for the person who will actually do the work, and to have a reasonable estimate for the total time/effort involved. You need the former to allocate (or delegate) the task; you need the latter to finish the planning. [FRE 14]

Knowledge is the raw material of software design teams. For complex projects, knowledge from multiple technical and functional domains is a necessity**.** Ideally, a software design team is staffed so that both the levels and the distribution of knowledge within the team match those required for the successful completion of the project. Because of knowledge shortfalls such as the thin spread of application domain knowledge in most organizations, however, this is seldom the case**.** In general, individual team members do not have all of the knowledge required for the project and must acquire additional information before accomplishing productive work. The sources of this information can be relevant documentation, formal training sessions, the results of trial-and-error behavior and other team members.

Group meetings are an important environment for learning, since they allow team members to share information and learn about other domains relevant to their work. Productive design activities need to revolve around the integration of the various knowledge domains. This integration leads to shared models of the problem under consideration and potential solutions [DIA 15].

We should analyze various ways suggested to handle the shortfall of domain knowledge by Diane B. Walz. It is duly noted that actions such as documentation, training, and group meetings should be taken well in advance to ensure the smooth running of the project.

According to a project conducted by Diane Walz, a project was closely watched to understand the way it was managed and in one of the meetings the following points were noted. Few of the designers were familiar with Prolog as the implementation language and were reluctant to commit to implementing the object server in Prolog. Thus, the issue of whether the object server should be written in Prolog or some other language remained unresolved. A great deal of technical information on Ingres as the implementation language was shared so that the group could evaluate its potential as a tool for building the object server.

Technical knowledge was introduced, exchanged, and evaluated according to its ability to meet requirements in the context of one or more specific design approaches. New information about requirements was evaluated in the context of design approaches framed in terms of technical and application knowledge. Presentations about new technology were discussed in light of various design approaches and whether or not such approaches met requirements. Thus, new information was sought, filtered, and integrated in context. [DIA 15]

This supports the theory mentioned above which claimed that managers should be well aware of all the domains of knowledge and have special expertise in the management domain. If some decision needs to be taken for a technical problem, right kind of information and data should be demanded by the management.

An interesting discussion by A.T. Berztins can be mentioned from the 6th international Software Process Workshop at Virginia. Commenting on Component based development,

it has been noted that while components are brought in an institution during the development phase, a large amount of knowledge and skill is also brought in. It is also emphasized that we are equipped with some amount of skill to develop our product further due to acquisition of such components and thus highlights the importance of having knowledge and skill at the right moment to develop any artifact. [BER 16]

It is important to note that the introduction of any component can be perceived as introduction of new capabilities to the team as well as the reduction in the capability to develop the component. But on the other hand, it has already enabled us to take a step further and concentrate on the issues lying ahead. Some reverse engineering can be done on the components in case the code is also available.

We can see that a lot of emphasis has been laid on gaining enough application solution domain expertise and knowledge before starting with the project.
Consider the design of editor software. If the analyst knows that the editor will have to be ported to a different operating system within a given period of time, provision can be made for that in the analysis model. This kind of flexibility need not be included if the platform is not going to change. This means that the effort required in incorporating the factors leading to a flexible design can be avoided if the system does not need that quality. [REQ 17]

Looking at these findings we can confidently say that there is a definite existence of the KABASPP domains throughout the ongoing research and practices in the Current Software Project Planning Scene. It is still conspicuous that although the domain knowledge plays an important role in defining the success of the project, it still is not considered as an independent factor while undertaking planning activities. A lot of people will still recognize the importance of presence the right knowledge and skill at the right time; we still fall short of categorizing such knowledge and defining an organized way of looking at this knowledge shortfall. This factor is further defined as the “Knowledge Gap”.

Chapter 5

Pervasiveness of the KABASPP Process

A lot of emphasis has been laid on the pervasive nature of a process in human life. We shall now specifically look for the pervasiveness of the KABASPP process throughout the Planning scene. Please refer to Pg. 7, Para. 1.

Leon Osterweil has studied such instances and also noted a key difference between a process and process description. A manual how to drive a car would qualify as description of the process while driving the car actually would be a process. Thus we end up creating generalized solutions and archive them as process descriptions. Instantiation of these process descriptions is done every time a new process model is introduced. Is there anything particular that we should be taking note of in such a scenario?
Process Descriptions are said to be static while processes are dynamic as explained by Dijkstra in his insistence to the minimum usage of the GOTO statement. So when we generalize, we may include some examples which did not suit well and when out of context would give absolutely variant results. This calls for a framework to be defined which can be utilized in defining the project plan which not only understands the changing nature of the development process but also encapsulates the necessary ingredients. [LEO 18]

What then forms the base for such project plans? As seen earlier, knowledge and skill are the most important inputs for a software development process. We will be looking at various other engineering domains to see how the availability of the raw materials and mechanism plays a vital role in project planning. We can translate the same need here and claim that much of software development planning should rotate around the required knowledge and skill acquisition.

Based on the belief that superior product is a result of a superior design whereas a superior design is a result of superior design process, we shall look at some interesting process issues which will re-iterate the importance of knowledge acquisition. Software Design is one such activity. Much work done by Leon Osterweil shows that design has been used as both a verb and a noun in the software industry. This means that the variation in the final design, noun, are due to the variations during design (verb). The reactions and feedbacks of users to variations in the design process seem to be potentially useful tools in determining what aids human designers want and need. In particular, resistance to certain forms of assistance and more ready acceptance of others is noted. That, in turn, seems to promise to provide insights into the nature of design, and the ways in which humans perform designing. [LEE 19]

Apart from the successful and timely completion of a software design, what really matters is the quality of the outcome. It is imperative to hold high standards of design and development plans in order to march ahead in a competitive market.

"Estimating quality of software systems has always been a good practice in software engineering. Presently, quality evaluation techniques are applied only as an afterthought to software design process. However, quality of a software system should be stated based on the end-user's requirement for quality. Based on this observation, a paper was proposed in 2005 for an estimation model called ReQuEst (Requirements-driven Quality Estimator). ReQuEst is an attempt to quantitatively estimate the quality of a system being designed from its analysis model. The quality is estimated in terms of adaptability and extendibility which are also important parameters in system design. During requirements analysis, evolving requirements are also analyzed to capture a few quality indicators from them. These indicators are used to compute the requirements for the above parameters from the analysis model. This is an extension to the thought discussed earlier where we suggest some tweaks in terms of rearrangement of development phases. Thus, the analyst can quantitatively specify the quality demands of the system to be designed along with the functional requirements. These quality specifications enable the system designer to precisely design systems meeting the values specified. Further, the model can be used to estimate the maintainability of the system in terms of the above parameters." [JAN 20]

This gives us a much clearer idea about the impact of knowledge acquisition in every aspect of software project development. What needs to be studied in m much detail is the impact of such factors on the project plan. The managerial domain is the most involved in such issues and a lot of work has been done in order to categorize employees in such a manner that their competencies can be put to optimum use.

NASA faced quite a bit of failures lately in their space shuttle development plans and thus this triggered a re-look at their process by GAO [Government Accountability Office].
At NASA major Review is done before moving from design to Implementation phase to ensure the knowledge maturity level. NASA has defined certain knowledge points to review knowledge and to check future feasibility of every project.

This review provides them with certain sound information for investment decisions. It has to be noted that an investment up to the scale that NASA deals have a large stakes and dependencies.

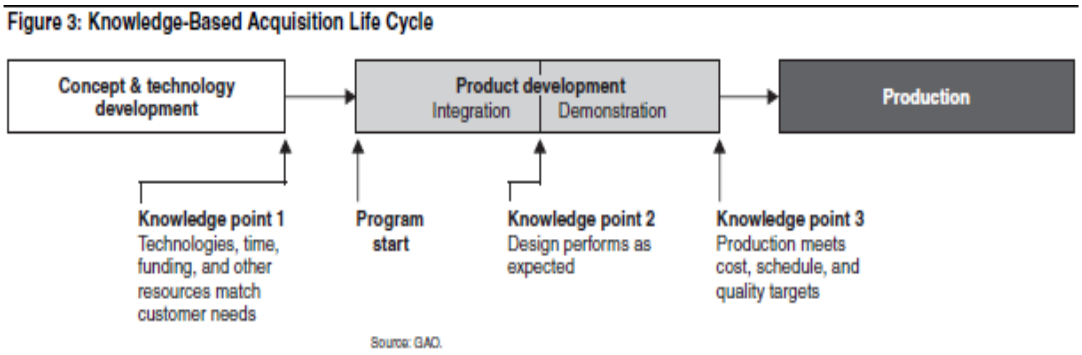

Figure 5: Knowledge Points [GAO]

But for developing their Flight control and most of the Ground projects, they do not look at the technology and "Skill" maturity levels. This caused some initial jitters in the projects and so it was suggested that they use Technology Readiness levels at each of

these points. [TRL] was defined by NASA itself and now should be used as a contributing factor to prove the future feasibility. [GAO 21]

TRL as defined by NASA was mostly used to gauge the readiness of the technology levels. This helped in minimizing the risk factors as higher TRL levels indicated lower risks. One of the major factors for critical projects such as the Flight control is the minimization of risk. This was easily achieved using the TRL.

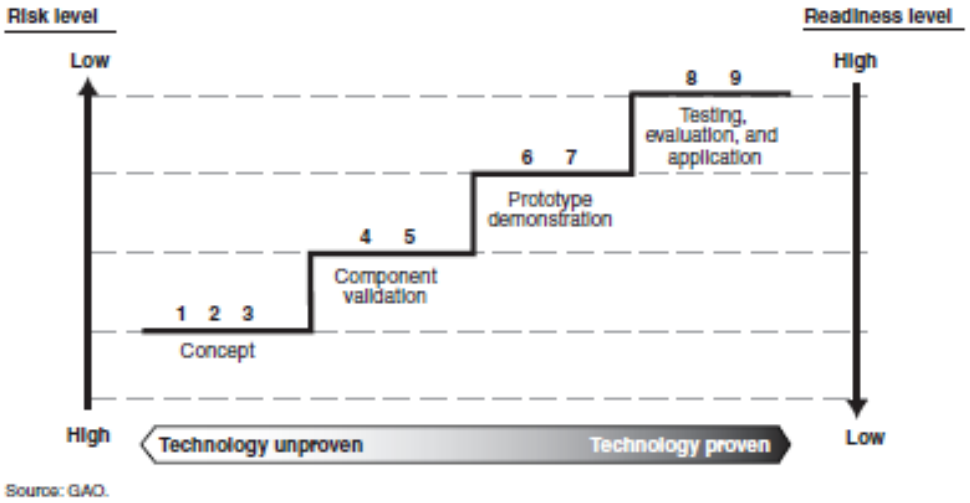

Figure 6: TRL by NASA [GAO]

Another contribution towards the non-rigid process model research was done by Leon Osterweil and Aaron G. Cass in the following discussion. A possibility was discussed of having a collection of models which collectively are consistent with the requirements.
This requires a series of activities to complete the models. Thus the planning would be changed according to the latest context and status. This would be an excellent example of an opportunistic approach. It also is mentioned that it becomes rather difficult when it comes to formalizing such a model. As part of this ongoing effort, they have developed a process-programming language called Little-JIL and an interpreter for it called
Juliette and have used both to encode and execute various complex processes, in software engineering as well as in such other domains as medicine and government.
The method followed here is as explained above and a large amount of TWBS [Task Work break down Structure] has been used and every task has been considered as a

model and completed. A stress has been laid on acquisition of knowledge for the development of the artifacts. [AAR 22]

This can be seen as a major shift from traditional thinking of rigid and static processes. A large amount of dynamicity has to be introduced in order to incorporate the changing conditions and project delays.

The idea behind looking at various software process cases was to study the explicit or implicit pervasiveness and presence of knowledge acquisition in the software development process. We can clearly state that although the timely acquisition of knowledge and skill has proved to be critical, not much effort or formalization has gone into recognizing this situation.

KABASPP was introduced with a similar motive and intends to fill this gap by introducing the 5 domains of knowledge. We shall see later on in the proposal how these can be utilized to make full use of knowledge based planning.

Chapter 6

Presence of KABASPP in Traditional Software Engineering Issues

We shall now look at the presence and implication of knowledge acquisition in Traditional Software Engineering practices such as Testing, Component Based Design, and Planning.

A reliable way of achieve a desired output of the available resource is to control the conduct of those executing the process throughout the complete process.

This clarifies the need for Management domain knowledge for a successful project completion when there is a constraint on the resources.

Although this control could be negative, various techniques and plans can be introduced to sugarcoat the discipline. [DEW 23]

In such a situation, it becomes mandatory to focus on providing knowledge and skills. Thus in terms of a process we should provide them with some instrumentation. Thus it is again important to have the right information about the right tools at the right time. It can thus be clearly seen that it has been emphasized time and again to ensure the presence of knowledge and skill required as claimed in KABASPP [HUM 24]

A lot has been written on context switching for programmers and it is interesting to study the excerpts. Context Switching is difficult and thus it is tough for a person to easily adapt to the new phase. Thus if a person needs to switch from the requirement phase to the design/coding phase then we can conclude that he also needs to switch from the Application domain to the Development Environment domain. Thus such a context switch would be successful if that person is equipped with the knowledge and skill to handle the new environment. [OST 25]

Some disagree to this and claims that since we do so many things concurrently we can do it easily. Thus it is claimed that we have a % of knowledge of both the domains and thus makes it even more important to know about these domains and study the shortfalls so that we can handle both the processes simultaneously. [KAP 26]

Thus we should provide the people with sufficient support doing it so that they can switch easily thus, implicitly, making the people ready for the Application solution domain.

Chapter 7

Use of Knowledge acquisition in Traditional Engineering Domains

Project planning has been existent since centuries. Disciplines like Construction Engineering, Mechanical Engineering and Chemical Engineering have caught the interest of mankind since ages. Over the years of development of marvelous architectures and machines, these disciplines have matured to a stage where the craft and art of developing the final product is quite rigid. Compared to such immense history of success and failures, Software Engineering seems like a newborn. It has not been more than thirty years that the concept of engineering software has caught up. We intend to study many such cross domain examples and see how they relate to the software project planning. We shall go through some examples of successful project planning activities and some problems faced by this industry.

As discussed earlier, Construction Project Management is not seen as much as an internal affair but is largely accomplished through management of personnel of different domains and employers working together.

Could this be so in the Software Development Team? In the coming years, with highly outsourced and component based development, this could very well be the case.
Mostly planning activities are carried out using a simple approach of Task Work Breakdown Structure. This also calls for prioritizing these activities by using techniques such as Precedence diagram Critical Path Method.
Activities in a Construction Project are highly dependant on the completion of previous activity. An example of this can be seen in the construction of a pile supported footing.
This activity includes the following sub-activities. Excavation, Building of footing forms, Procurement of Piles.
All these activities can be said to be independent in terms of their raw material and craftsmen required. But in a process of constructing a pile supported footing, all these

activities are interdependent. Each activity is followed by some activity except the last activity.

This puts every activity in an important position. This is the reason why not even a single activity is put on hold for more than a threshold amount of time on Construction Engineering. Although contrary to the dependency, the raw materials required for the remaining activities can be procured in advance or on schedule to avoid any further delay.

Can this be said for the Software Engineering Activities? Can we procure the raw material required for building software? The answer is yes. This can be achieved by careful planning and recognition of the fact that delays can be caused most of the times due to lack of knowledge or expertise and not due to the lack of facilities.

We should also study some of the major constraints faced during project planning and try to map them to the software engineering domain. Planning for a Construction project is mostly restraint based. This means that the shortcomings in the requirements are used as the basis for design and development decisions. This may help us in exploring new approaches towards software project planning.

Construction Project	Software Project
Raw Material Constraint	Knowledge Constraint
Equipment Constraint	Skills/Tools Constraint
Functional Constraint	Requirement Deadlines Constraint
Safety Constraints	Quality Issues

Table 2: Restraint Matching

This clearly signifies that restraints as shown above play a major role in any project planning. These should then be shown as activities on the job logic or plan. The process of satisfying these constraints in a software development project can be viewed as the process of knowledge acquisition or skill development, i.e. KABASPP.

Another example where project restraints play a vital role is when restraints act as dependency between two activities. Let us take an example of a Carpentry Job where the Drill is been required by two Carpenters for two different jobs at a time.
Similar example can be seen in the parallel development of software where tool like ClearCase prove to be handy. When two or more teams are working on the same artifact then managing the skeleton of the system could prove to be a real tough job.

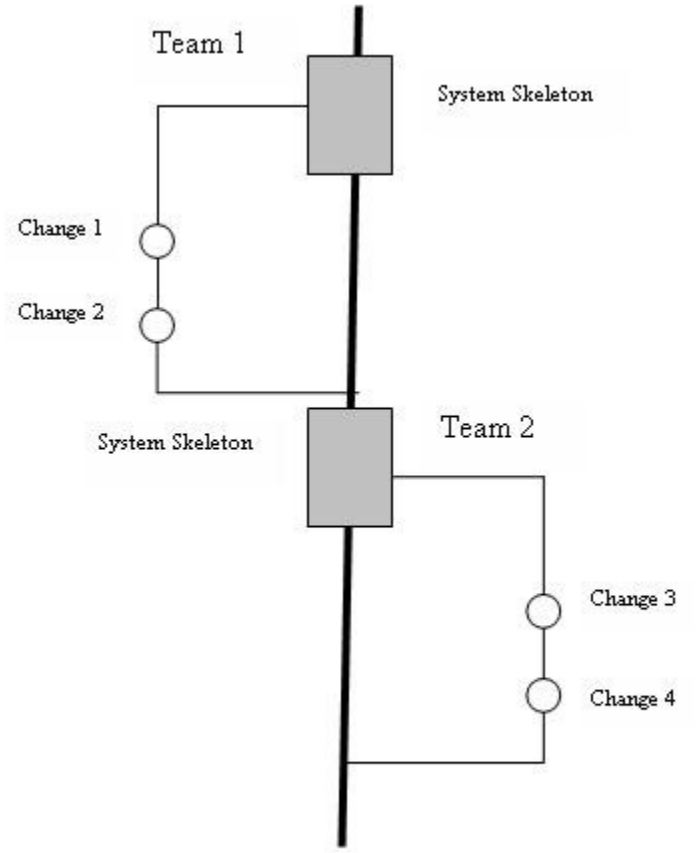

Figure 7: Parallel Development

This would need a lot of expertise in managing the integrity of the basic skeleton when more than one team is working on it. In Fig. 9, a sequential approached is accepted but there could be artifacts which are used in common. Thus it is really important to understand the dependencies and the integration tricks for a parallel development.
[RIC 8]

Let us now look at some of the major problems faced in Construction Projects and how they can be translated for similar situations in Software Projects.

a- We cannot expedite physically beyond a certain level. *Is this so in Software Development? Can the process of coding be expedited by adding personnel?*

It is a known fact that adding people late in the project will force the team into the learning phase and further delay the project. What if we anticipate such crunch situations and start preparing the personnel for late introduction?

b- If we do expedite some of the critical paths, other dependant less critical path may now become critical. *If we finish the design early, what effect will this have on training time for developers?*

c- Long Range and Short Range (30 days) planning is done separately. For short range plans, necessary workers and technicians can be interchanged. *Can this be a case for Software Teams?*

d- Construction Projects need different manpower throughout the project. Thus the Hiring, Training, Firing technique will not work or else you will loose your best craftsmen. *Similar can be said about the software development teams. Can this be overcome by proper training and knowledge maintenance?*

Let us consider a project for 10 people. We would be requiring the following skills throughout the project. UML modelling for Design, C++ coding for development, MS Project for Documentation, Jtest for Automated Testing.

The project is to be completed in 3 phases. Each phase requires a different combination of skills and the task is to manage the 10 people for the development. What could be the easy way out for such situations?

Requirements	Phase 1	Phase 2	Phase 3
1	7 UML	6 C++	7 Jtest
2	3 MS Word	3 MS Word	3 MS Word
Total	10	10	10

Table 3: Competency Requirements

A simple example where each phase requires strength of 10 people is explained. Such situations are generally tackled in the software domain by finding ways to acquire those skills in the following ways.

1- Internal Transfers

2- Sub Contracting

3- Training Schedules.

This process is known as Resource Smoothing. [RIC 8] It can be thus claimed that managers should adapt to a Knowledge based project planning such as KABASPP which will ensure timely availability of the required skill and knowledge.

Lastly we can look at some interesting research which will come very close to recognizing the need for special attention for knowledge acquisition. Working on similar lines, we can deduct from a report published by the European Foundation for the Improvement of Living and Working Conditions that an ergonomic approach needs to be adapted to improve the working conditions and productivity.

It is important to bring together all specific areas of knowledge relating to human workers, design, tools, Machinery and safety. This is called as an ergonomic approach. Today most of the trades are multi-disciplinary and worker is at crossroads of a number of constraints such as noise, heat, workload, achievement.

The actual work may not match the design and participatory goals unless every stakeholder is involved in the evaluation process. Thus an issue like safety needs to be handled at all levels of knowledge and position such as the designers shall take care of design faults and will be the top brass. The managers will ensure that the rules and precautions are followed strictly to avoid any deviance from the safety guidelines and the design. Workers follow the procedures individually and take care of their own safety. Finally the owner takes care of the budget issues regarding safety.

[DRA 27]

Thus it is highly imperative to ensure the presence of knowledge from every sector and domain as defined in KABASPP. We have now seen clearly that without using

KABASPP or any knowledge base for the planning activity proves disadvantageous for any project.

On a similar note, we shall now look at Team formation issues which can be solved using the KABASPP approach.

Chapter 8

Manpower Buildup and Team Formation

Having had a pretty good look at the pervasiveness of KABASPP and knowledge acquisition in the Software Engineering field, let us now concentrate on the application of these factors for activities such as Team formation and Manpower buildup throughout the development process.

Cognitive Psychology deals with many such issues related with the logic behind performing the development tasks. It is the study of the mechanism by which mental processes are carried out and the study of type of knowledge required for each process. [HOC 28]

We wish to use such approaches to tackle the ongoing problem for the lack of acknowledgement of knowledge as an input in the software development process.

An interesting categorization has been done by Silvia et al, for grouping employees according to personality factors and competencies. This forms a managerial point of view about employees and a lot of emphasis has been laid upon selecting the right people for the right places. These 16 Personality factors have been grouped into 4 major competency brackets and are as follows.

A) Intrapersonal
    1- Analysis
    2- Decision-Making
    3- Independence
    4- Innovation and Creativity

5- Judgment
6- Tenacity
7- Stress Tolerance

B) Organizational
1- Self Organization
2- Risk Management
3- Environmental Knowledge
4- Discipline
5- Environmental Orientation

C) Interpersonal
1- Customer Service
2- Negotiation Skills
3- Empathy
4- Sociability
5- Teamwork and Co-operation

D) Management
1- Co-Worker Evaluation
2- Group Leadership
3- Planning and Organization

Following table has been used from the referenced paper to exemplify the exact nature of categorization of employees. This is a suitability table used for matching the competencies and personality factors to that required for a particular role.

Capabilities required to fulfill a role in the organizations we studied*

	Capabilities																				
	Intrapersonal							Organizational					Interpersonal					Management		
Software roles	Analysis	Decision-making	Independence	Innovation and creativity	Judgment	Tenacity	Stress tolerance	Self organization	Risk management	Environmental knowledge	Discipline	Environmental orientation	Customer service	Negotiation skills	Empathy	Sociability	Teamwork and cooperation	Coworker evaluation	Group leadership	Planning and organization
Team leader	✓	✓		✓			✓		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
Quality manager	✓	✓	✓	✓			✓	✓				✓			✓		✓	✓	✓	✓
Requirements engineer	✓				✓		✓					✓	✓		✓	✓	✓			
Designer	✓	✓	✓			✓	✓	✓			✓	✓			✓		✓			
Programmer	✓	✓	✓			✓	✓	✓			✓	✓			✓		✓			
Maintenance and support specialist						✓	✓	✓			✓	✓	✓		✓		✓			
Tester			✓		✓		✓	✓			✓	✓			✓		✓			
Configuration manager			✓		✓		✓	✓			✓	✓			✓		✓			

Table 4: Competency Matching for Managers [SIL 29]

The basic idea behind using such techniques is to identify the requirements for each task and satisfying it. It can be clearly seen that the 16 factors mentioned here act as input for the final product. The process followed for achieving this is simple.

a- Categorize people according to the 16PF.

b- Define the Roles

c- Match the Individuals. [SIL 29]

KABASPP is on similar lines although with a major difference in the strategy for competency matching. What is important to note here is that such solutions can be applied to situations where new members are not to be recruited. But when a new team is to be formed, it is imperative to define the task and subtasks first. The requirements for these tasks are to be clearly identified based on the 5 domain structure defined in KABASPP. Thereafter the possible recruits are matched to the requirements and those who fall short are deployed on a respective training session. We shall see more of this in the proposal.

A large amount of research has thus been pointed towards competency development. One such paper has been proposed by Judy Hallstrom. She has organized the skills and competencies of the mangers into 3 different compartments. Each of these competencies is interdependent and the weakness and strength of each depends on others.

The Three Competencies -Business, Technical and Management

Paper identifies three knowledge/skill domains that were common to most successful PMO groups. These are

Business Acumen - what you need to be able to effectively manage projects at an Enterprise.

Technical Acumen - the knowledge and skills that are specific to Project Management.

Management Acumen -the knowledge and skills needed by anyone supervising others but specific to the project management arena where one may be managing people on a project that are not direct reports.

The process followed in this is as follows

- to create a plan based on the shortfalls of knowledge
- to asses the skills shortfall
- adjusting the plan accordingly

[JUD 30]

This is a very conservative approach and can be very helpful in guaranteeing the completion of the project.

Carrying on the discussion over manpower classification, Niederman in his paper has clearly mention three categories of competencies for E-Commerce Project teams.

[NED 35]

Knowledge- Area	High-Level Skill
Technical	Web Programming
	Web Networking
	Web Databases
	Web security
	Web Management
	Web Site Design
Human	Interpersonal Communication
	Problem Solving
	Conflict Resolution
	Collaboration
	Dealing with change
Organizational	Organizational Goals and Objectives
	Organizational Policies and Procedures
	Organizational functions and processes
	Organizational culture
	Organizational Constraints

Table 5: Manpower Classification

It is thus quite evident that all these solutions have been proposed with a clear idea of knowledge and skill as sole inputs for the final product. Other factors such as facilities and funds are taken for granted.

Now that we have recognized that training is absolutely necessary to hire and retain good professionals, we shall look at a training model proposed by Blanton et al. This proposal justifies the use of Requirements as the basis for planning the training programs. They propose that training efficiency can be measured by the amount of time spent in training whereas the training effectiveness can be measured by examining three different training outcomes: learning, task performance and organizational results.
Following methodology is used for this model.

- Task Analysis for each project
- Personal Analysis of the potential recruits
- Analysis of total Competency Deficiency
- Training the deficiencies
- Measuring Training Effectiveness
- Measuring Training Efficiency.

The research model in Figure is based on Blanchard and Thacker's five phases of the training process.

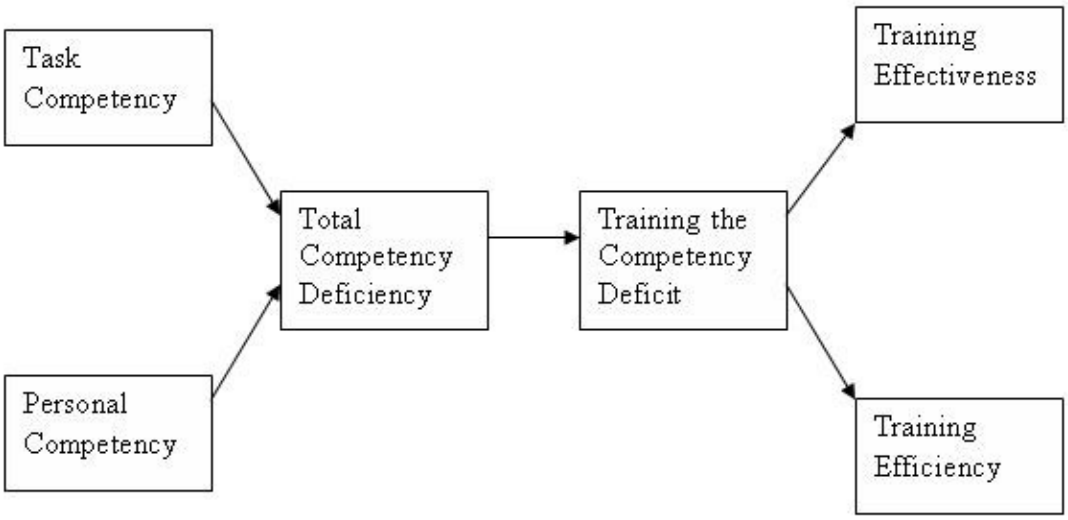

Figure 8: Task Deficiencies

It is thus very important to note that such a framework can be utilized in applying KABASPP for planning. This is very similar to the process we wish to propose but needs to be using more of KABASPP for tasks such as Personal Analysis and Task analysis. Proposal will throw more light on how to actually find the Competency Deficiency and some methods to gauge accurately the lack of knowledge. [BLA 31]

Finally before going to the proposal we shall look at some work done by the Software Engineering Institute (SEI) on Team processes and P-CMM. TSP (Team Software

Process) and P-CMM (Person Capability Maturity Model) can be viewed as models introduced at the SEI to address the issue of optimum Human Resource Management.
The documentation can be downloaded from the SEI site.

The People CMM describes an evolutionary improvement path from ad hoc, inconsistently performed workforce practices, to a mature infrastructure of practices for continuously elevating workforce capability. We shall not study the details of P-CMM or TSP but it is important to understand the logic behind introduction of such frameworks. Following principles have been quoted to be the basis for P-CMM.

1. In mature organizations, workforce capability is directly related to business performance.
2. Workforce capability is a competitive issue and a source of strategic advantage.
3. Workforce capability must be defined in relation to the organization's strategic business objectives.
4. Knowledge-intense work shifts the focus from job elements to workforce competencies.
5. Capability can be measured and improved at multiple levels, including individuals, workgroups, workforce competencies, and the organization.
6. An organization should invest in improving the capability of those workforce competencies that are critical to its core competency as a business.
7. The improvement of workforce capability can be pursued as a process composed from proven practices and procedures.
8. Since technologies and organizational forms evolve rapidly, organizations must continually evolve their workforce practices and develop new workforce competencies.

Following table quoted from the documentation can be used as a basic guide.

Levels\	Highlights	Process Improvement
Initial	Inconsistent Management	NA
Managed	People Management	Repeatable Practices
Defined	Competency Management	Competency based Practices
Predictable	Capability Management	Measured practices
Optimizing	Change Management	Continuously improving practices

Table 6: PCMM levels [PCM 32]

P-CMM clearly identifies the need for an agile workforce and emphasizes the subject of the discussion here that the human resource needs to be trained adequately according the project requirements. It thus has suggested a 5 level maturity framework for the entire workforce starting from the initial to the optimized which are all based on the maturity of Process areas defined for the individuals. It thus concentrates on the Process areas and Process area goals independently. It accumulates the common practices and tries to achieve the Process area Goals. Finally achieving the maturity levels is thought of.

Chapter 9

Proposal

We have looked at various domains in Software Engineering and in other Engineering Disciplines to understand the logic behind planning decisions and strategies. We are enabled to conclude that KABASPP as a process and KABASPP as a Project Planning Framework is the need of the hour. We have seen how KABSPP can be incorporated into existing project planning models such as the Waterfall model and how can activities such as Component based Design and Testing can be done more efficiently using KABASPP. We have also looked at some of the major differences seen in traditional engineering disciplines like Construction Engineering and that in Software Engineering. This study has leaded us to some important questions which need answering.

As seen in the work by Putnam and Myers, some principles applied to resource management in traditional disciplines can be reapplied in the software development project. It is important to study the life-cycle manpower model [NOR 33] before analyzing the work done by Putnam and Myers. Peter V. Norden developed a life-cycle manpower model in 1963 at the IBM development laboratory, New York. This model was specifically designed to look at the impact of investment done in human resource in a hardware project. It was duly noted that considerable amount of the total expenditure was spent on human resource. He then applied the Rayleigh Curves, well known after the physicist Lord Rayleigh, to this manpower buildup. The Rayleigh equation also applied to the manpower curve proposed by Peter V. Norden.

$$y = 2Kate^{-at2}$$

Here,

y = manpower rate at each point on curve
k = effort (area under the curve)

t = development time
a = a constant governing the time to peak manpower

The graph can be seen in Fig. 10.

We can clearly see a pattern of growing strength in terms of people in the beginning of the project and gradual settling down through the middle phases of design and development and finally a drop towards the completion of the project. This is a fairly acceptable pattern for staffing in most of the hardware projects.

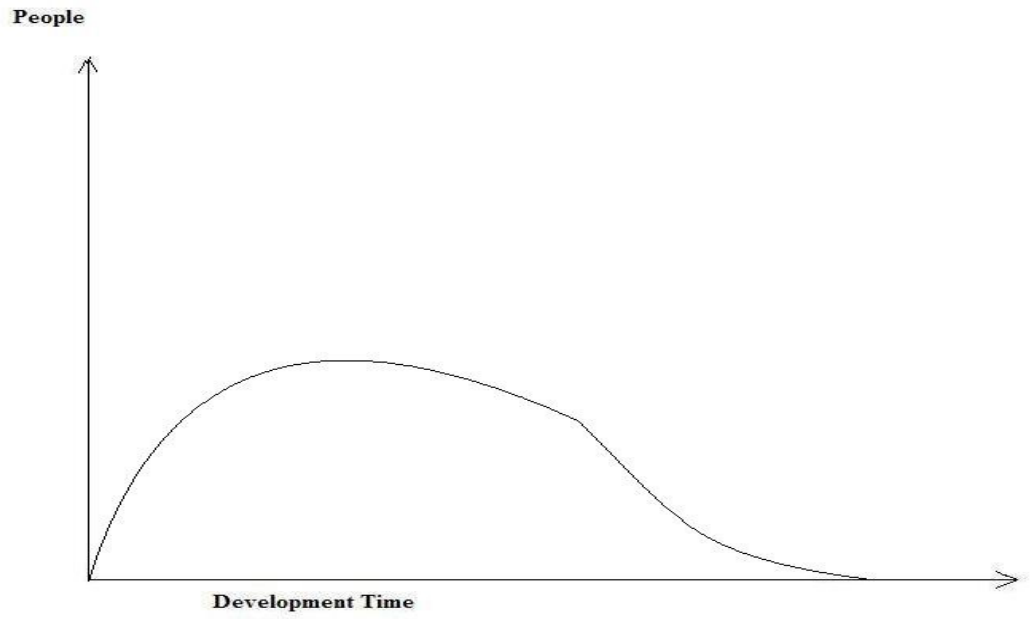

Figure 9: Staffing pattern for a Hardware Project

There is a considerable amount of relation between the Rayleigh curves for hardware and software. Implications for these now point towards the project planning process and the sequence of activities taking place during the development life-cycle.

Despite the fact that there is disagreement about the practicality of actually applying the Rayleigh Curve for software project planning, the follow up done by Putnam and Myers and many other research institutes increase it's credibility.

We can see a widespread usage of the Rayleigh curves for project planning purposes ranging from government organizations to private research institutes and educational bodies. This life-cycle manpower model was further applied to the software engineering projects in their work by Putnam and Myers.

A life-cycle manpower buildup graph was introduced for the software projects. This can be seen in Fig. 11.
[PUT 34]

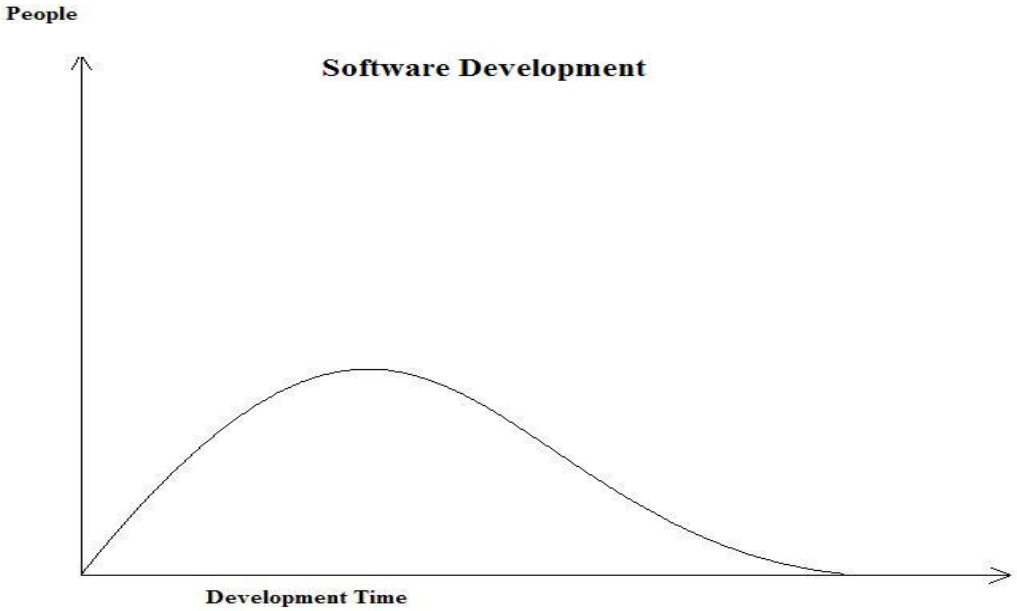

Figure 10: Staffing pattern for a Software Project

Let us analyze this situation further by introducing the concept of instantiation here. Let us say that at time t1, we need p1 number of people for the task that is currently ongoing at time t1.

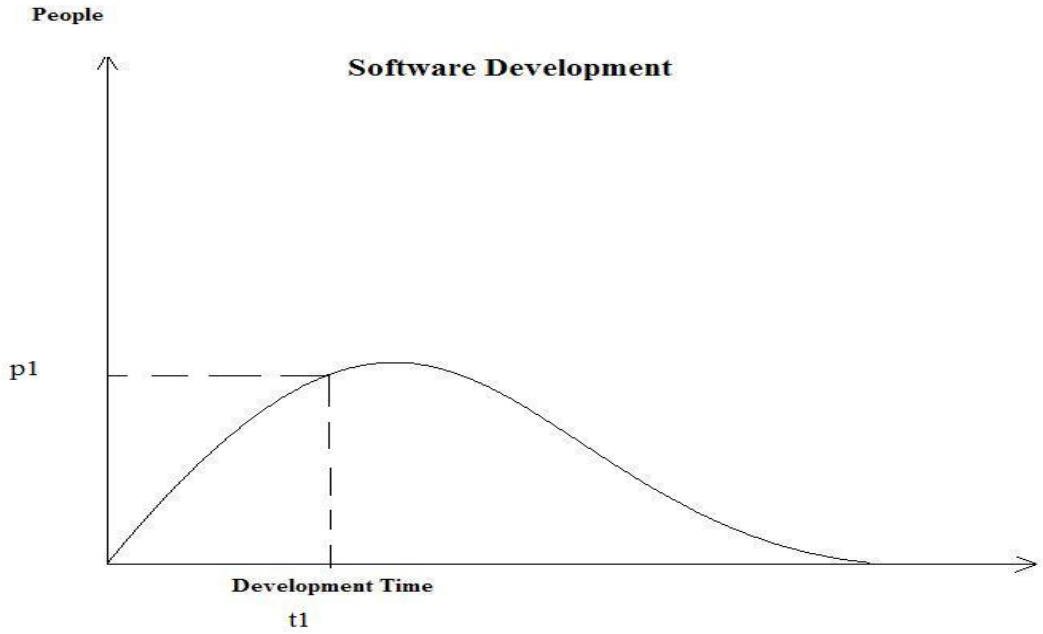

Figure 11: Instantiation of the Staffing Curve

Now, most of the phases in traditional software development plans like the waterfall-model are said to be dependant on the previous phase. Thus the completion of one phase leads to the initiation of the next.

Every phase can be seen as a miniature repetition of the overall process. Thus each phase can be represented in the Rayleigh curve fashion.

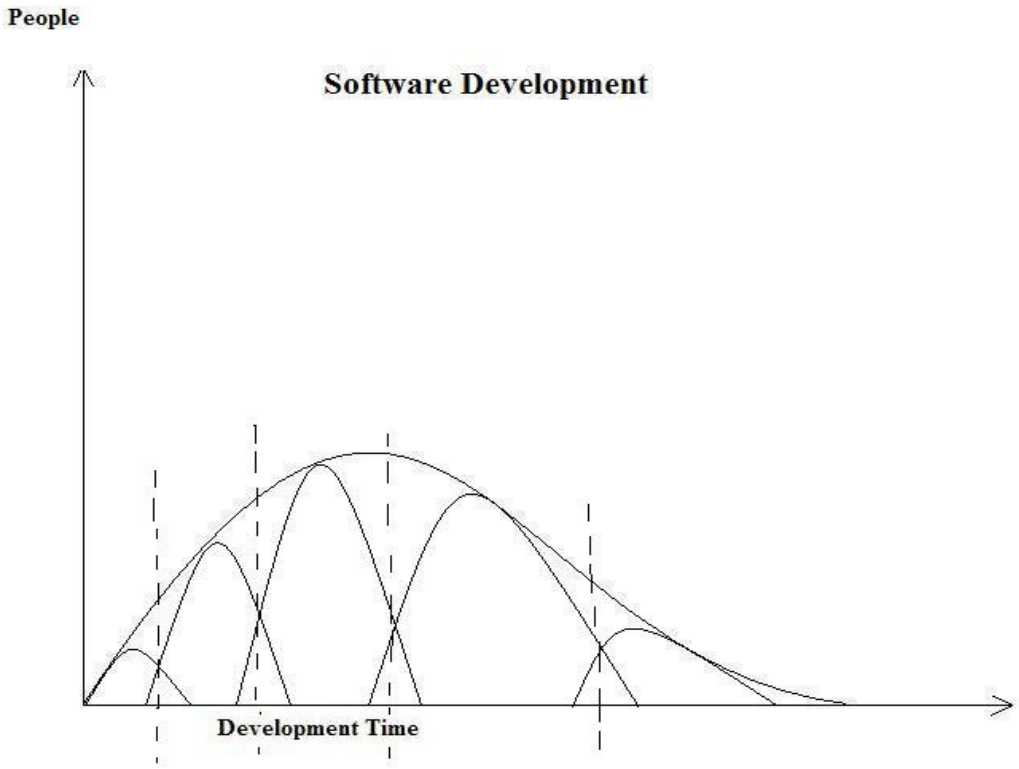

Figure 12: Staffing pattern for each phase of Software Development

How will this situation unfold in case the project runs overtime or lagging behind?

A lot has been said and done on Software Project Planning and utilization of resources like the staff and developers to achieve the deadlines. Many models have been introduced to study the staff loading patterns and the amount of effort required during the process.

The underlying assumption in most of these studies has been that the developer or team member is at the maximum knowledge or skill maturity level. General project management wisdom contradicts this assumption and compels us to have a re-look at the factors affecting delays in the planned schedule. The “Knowledge/Skill-factor” needs to be considered while analyzing such situations.

Thus considering the case in Fig. 12, when we expect p1 number of people to be deployed on the project, we are actually implying for the deployment of 12 completely trained and experienced individuals on the project. Is this actually possible in all situations? As mentioned earlier, general project management wisdom shall contradict this and lead us to completely new angle of looking at this staffing issue.

As we have mentioned earlier, Knowledge/Skill forms the foundational investment in any software development project. Thus we can conclude that the raw material for any software development is the knowledge and skill of the development team. It has been thus highlighted throughout the paper that the satisfaction of these inputs at the required junctures will ensure timely completion of the project.

In order to ensure this we should introduce some kind of a metric system which will enable us to gauge the current status of Knowledge/Skill levels and the required Knowledge/Skill levels.

Let us consider a Knowledge/Skill grading for the software developers on this project and call it "Knowledge Points". [ii]

Consider p1 = 12.

If the developers are to be graded on a range of ten based on their knowledge and skill levels, a few suggestions for which are mentioned later, then we should be looking forward to deployment of 120 knowledge points at point t1. We shall call this as "Knowledge Loading".

"Knowledge Loading" can be defined as the amount of knowledge /skill required at a particular time t to ensure the timely completion of the task adhering to the requirements that is ongoing at point t.

[ii] This is not to be confused with knowledge points introduced by GAO, USA which is actually a phase whereas the knowledge points mentioned here shall act as metrics for measuring the knowledge and skill levels of the developers.

This finally changes the complete outlook towards Figure 11. This now forms as the Requirement Curve rather than the actual loading curve. If this is satisfied then we can vouch for successful schedule estimation. In reality, we can see that the actual loading is not as we desired.

So finally when we compare the required amount of Knowledge/Skill required for the completion of the task at point t1 to the knowledge/skill available in p1 team members, we can conclude that both are not in conformance with each other.

Thus we shall look at another Rayleigh curve to explain the knowledge loading concept.

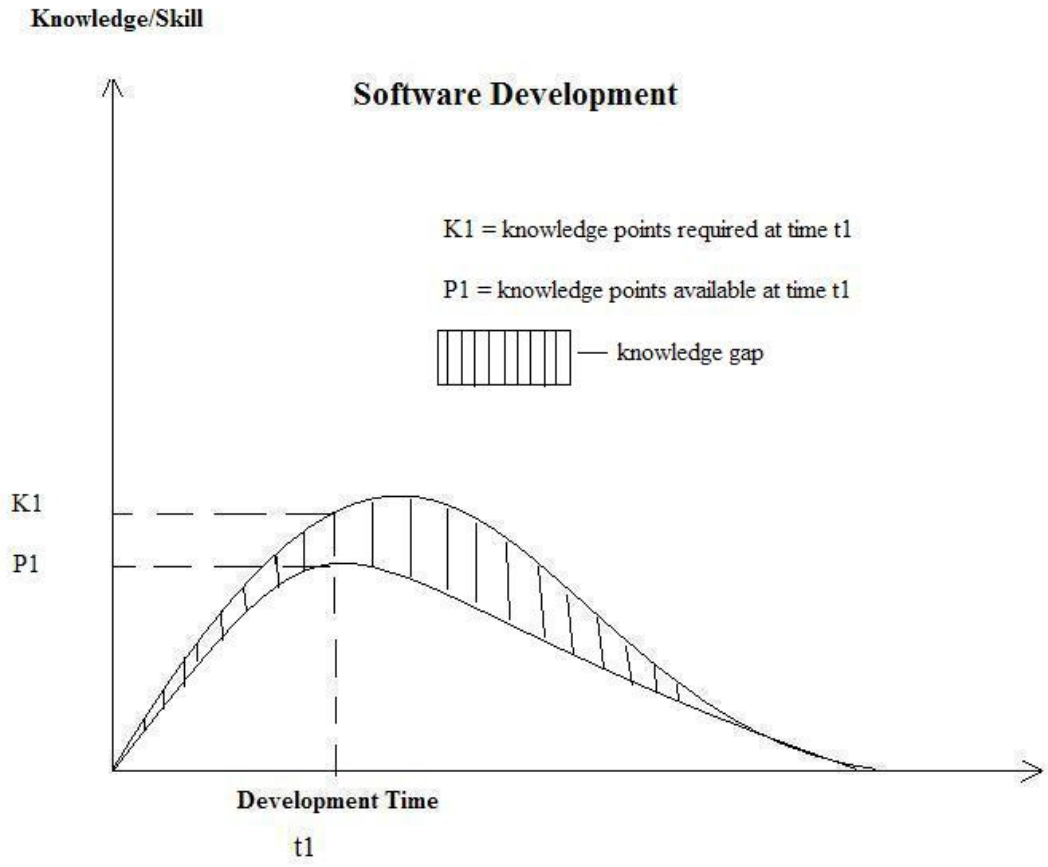

Figure 13: Knowledge Gap

The answer to this problem lies in introduction of KABASPP as an underlying framework for software project planning. We will be looking at various strategies in dealing with this knowledge gap problem.

Let us consider the following hypothetical situation for solving this problem and looking at the various options available for solving it.

Gyan Inc. is supposed to be a startup firm which develops location based solutions for home users. It deals with external telecom providers and has just bagged a project for developing a geospatial tracking and reporting.

Project managers at Gyan Inc. have studied previous projects and sketched a schedule and development plan. They have also decided to deploy 12 members including 4 testers, 3 developers, 3 designers, an accountant and a project manager on the team based on the function point estimation done by the managers.
During the development phase, Gyan Inc. faces a shortage of J2EE developers and monitoring clearly mentions that the module cannot be developed in time.
Apart from the 3 developers, the 3 designers can be used as developers. It is a well known fact that adding personnel late in the project shall further delay the process as the complete development team goes into a learning phase.

There are methods in which such a situation can be handled but before we look at it let us step back see if we could have avoided getting to this point all together.
It is proposed to use the KABASPP process for planning the project. This would lead us to follow these steps.

1- Analysis of the skills\knowledge necessary and when.
2- Analysis of the skills \knowledge available.
3- Acquisition of the skills\knowledge necessary before the task is initiated.
4- Actual performance of the task.

A primitive guidance is provided here to understand the application of KABASPP concepts to projects. These can be changed or instantiated as per the specific requirements of the project group.

Step 1 requires a study of previous projects by experienced managers and a categorization of the technologies\skills required on the basis of KABASPP knowledge domains. [REE 2]
A detailed analysis of the knowledge\skill requirements has to be done for each and every phase of the development. They should be categorized into the KABASPP domains namely Application domain, Application solution domain, Development environment domain, Run time environment domain, and Managerial domain.
This is followed by scheduling and estimation of the project and a detailed workflow is devised. To obtain the knowledge points, we can multiply the No. of people required (For one particular skill) by 10. This shall give us the total knowledge points for that skill required at time t1. Adding up the knowledge points for each skill required at time t1 shall deliver us the total number of knowledge points required at time t1.

Step2 is one of the most important stages of software planning. Skill\knowledge available can be calculated by grading each individual who is assigned to the project or is a probable.
Based on the 5 domains, these individuals are graded from 1 to 10 with an increasing order of knowledge\skill.
To ensure that these calculations are not affected by bias, time or location, following methodology is used.

The current knowledge and skill of a person can be calculated in the following ways and each answer has a different weight depending on the project. An informal survey concluded the following methods and their weights. These are guidelines for grading an individual for knowledge/skills.

No.	Method of Grading	Weight
1	Work Breakdown Structure (WBS checklist) [To be filled by the individual who is graded]	20
2	Interview/Chat with the person conducted by the project manager	20
3	Observation [Monitoring regular work done by the person]	10
4	Meeting of the team members and discussion groups	10
5	Previous Work experience on the technologies required	30
6	Work inspections [Formal inspection done by peers or testers]	10

Table 7: Method of Grading an Individual for Knowledge

Step3 as discussed earlier would comprise of various strategies to fill up the knowledge requirements. This can thus be achieved by training the required personnel or hiring new personnel with the required knowledge or acquiring components for that module. Outsourcing the module can also be an option. How these options are mapped to the knowledge acquisition has been explained earlier.

Step4 concludes with the actual performance of the task.
This marks the end of a procedure which can be used for gauging the knowledge levels for an individual and the task to be performed using KABASPP.

As mentioned earlier, we have looked at the possibility of rearranging the phases of software development in such a manner that many activities can be performed in parallel which use the same kind of knowledge base. Thus in case of the design and testing of a search engine, we can match the design specification and test case requirements.

Let us consider a team working on a search engine which will search all the people and their telephone numbers living in your vicinity having a car. This search engine can be said to posses the following capabilities and this could form the major part of the design. The search engine should be able to perform the following.

a- to be able to search on the basis of name of the car
b- to be able to search on the basis of nearness
c- to be able to search on the basis of brand of the car

The conformance of these design goals will also ensure the passing of test cases for conformance of the requirements.

The test cases can be viewed as

a- Does the system search person x depending on the name of the car
b- Does the system search person x who lives nearest to the user
c- Does the system search person x who owns a similar brand of a car and lives in the vicinity

Bothe these exercises can be completed with the help of the same knowledge base and such kind of activities can be done in parallel.

We thus propose KABASPP as a solution to a variety of problems in the Current Software Engineering Scenario. We are sure that with the right application of KABASPP classification and process, Project planning would be carried out much more confidently and would reduce the overall risk in the process.

Conclusion & Future Work

The initial study of the current Software Project Planning techniques gives some pointers towards the shortcomings or adjustments required in some basic assumptions. It is high time that we recognize that the real concern is right knowledge and skill acquisition at the right time. We have thus proposed KABASPP as a solution to this knowledge deficit and suggested ways to apply this theory to the existing planning techniques. The pervasiveness of KABASPP as a theory is now quite evident and reinforces its use. Manpower buildup and Team formation were considered specifically to demonstrate the Knowledge Gap and application of KABASPP approach. To conclude we can claim that some major re-thinking needs to be done while approaching Project Plans and Knowledge should be considered as in Input Factor while devising them.

Although this thesis has suggested a preliminary approach towards the application of KABASPP some major areas where there is immense scope for research are as follows,

- The Skills and Knowledge Levels required for a successful completion of a particular task needs to be accurately gauged and measured. This can be in terms of "Knowledge Points" as suggested in the thesis.
- The skills and Knowledge levels of an individual who is a potential recruit for the project needs to be gauged to understand where we stand. The difference between this measurement and the previous one will clearly quantify the "Knowledge Gap".
- Apart from quantification, a major project could be to formalize the technique of application of KABASPP process. This will also include the first two enhancements.

Chapter 10

Appendices

Appendix A: Bibliography

1. P.F. Drucker, *Management: Tasks, Responsibilities, Practices*, Harper &Row, New York, 1973[PETE 1]

2. K. Reed, *"Knowledge an acquisition based approach to software project planning".* Position Paper for CASE90, Amdahl Australia Intelligence Tool Program and La Trobe University, Melbourne, 1991 [REE 2]

3. K. Reed, *CSE41 SPM Lecture Notes Summary, Rev 2.0, Set 1*, La Trobe University, Melbourne, 2005 [REE 3]

4. W.W. Royce*, "Managing development of Large Software Systems: Concepts and Techniques"*, Proceedings of WESTCON, San Francisco, August 1970 [ROY 4]

5. S.L. Pfleeger, *Software Engineering: Theory and Practice*, Prentice Hall, 1998. [FLE 5]

6. Ian Sommerville, *Software Engineering*. Addison-Wesley, 7th edition, 2004 [SOM 6]
7. R. Pressman, *Software Engineering a Practitioner's Approach*, 6th Ed., New York: McGraw Hill, 2005 [PRE 7]

8. R.H. Clough, G.A. Sears, *Construction Project Management*, John Wiley & Sons Publications, Third Ed., 1991, [RIC 8]

9. D.D. Zimbelman, *Proceedings of a Symposium* by Irrigation and Drainage Division of the American Society of Civil Engineers, Portland, July 1987, [ZIM 9]

10. W.E. Howden, *"Validating Programs without Specifications"*, University of California at San Diego, Proc. TAV-3, Key West, December, 1989, [HOW 10]

11. K. Sherdil, N. Madhavji, *"Personal progress function" in software process*, Ninth International Software Process Workshop, 117-121, Airlie, Virginia, USA, 1994, [NAZ 11]

12. J.H. Lehman, "*How Software Projects are really managed"*, Datamation, ACM International Conference Proceeding, 1979 [LEH 12]

13. H.D. Mills, M. Dyer, and R.C. Linger *"Cleanroom Software Engineering",* IEEE Software, September 1987 [MIL 13]

14. G.M. Weinberg, D.P. Freedman, *"Reviews, walkthroughs, and inspections"*, IEEE Transactions on Software Engineering, 1984 [FRE 14]

15. D. B. Walz, J.J. Elam, and Bill Curtis, *"Inside software Design Team: Knowledge Acquisition, sharing and integration"*, Comm. of the ACM, 1993 [DIA 15]

16. A.T. Berztins, "Component based planning", *6th International Software Process Workshop,* Virginia, 1994 [BER 16]

17. D. Janakiram, M. S. Rajasree, "ReQuEst: Requirements-driven Quality Estimator", *Distributed & Object Systems Lab, Department of Computer Science and Engineering Indian Institute of Technology*, Chennai, India, 2004 [REQ 17]

18. L. Osterweil, "Software Processes are Softwares too", Revisited, *University of Colorado Boulder*, Colorado, 1997 [LEO 18]

19. Leon J. Osterweil, "Process Issues are Integral to the Science of Design", *University of Massachusetts*, Amherst, 1994 [LEE 19]

20. D. Janakiraman, M.S. Rajasree, "Requirement driven quality Estimator", Jan 2005[JAN 20]

21. "Implementing a knowledge-Based Acquisition Framework could lead to better Investment Decisions and Project Outcomes". *A Project Analysis Report by the U.S. Government Accountability Office for NASA*, 2005 [GAO 21]

22. Aaron G. Cass, L. J. Osterweil, "Process Support to help Novices Design Software Faster and Better", Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, 2005, [AAR 22]

23. D.E. Perry, A. L. Wolf (1994), "People Processes and Practices", 9th international Software Process Workshop at Virginia [DEW 23]

24. W.S. Humphrey, *"Using a defined Measure and Measured Personal Software Process",* IEEE Software, 1996 [HUM 24].

25. L. Osterweil (1994), *Proceedings of the 9th international Software Process Workshop,* [OST 25]

26. Kaplan, *N. Madhavji,* (1994), *Proceedings of the 9th international Software Process Workshop,* [KAP 26]

27. From *Drawing Board to Building Site*, Working Conditions Quality Economic Performance Report by the European Foundation for the Improvement of Living and Working Conditions, London, 1991 [DRA 27]

28. J.M. Hoc, T.R. Green, R. Samurcay, D.J. Gilmore, *"Psychology of Programming"*, A Publication of European Association of Cognitive Ergonomics. [HOC 28]

29. S.T. Acuna, Natalia Juristo, A.M. Moreno, Emphasizing Human Capabilities in Software Development, IEEE Computer Society, 2006. [SIL 29]

30. J. S. Hallstrom, "PM Knowledge - The Three Competencies", *Notes at Franklin Templeton Investments*, 2003 [JUD 30]

31. C. Gjestland, J.E. Blanton, R. Will, R. Collins, Assessing the Need for Training in IT Professionals: A Research Model, University of South Florida, 2006. [BLA 31]

32. B. Curtis, W.E. Hefley, S.A. Miller, People Capability Maturity Model (P-CMM), Version 2.0, Carnegie Mellon, Software Engineering Institute, July 2001, [PCM 32]

33. P.V. Norden, "Useful tools for Project Management", in *Operations Research in Research Development*, B.V. Dean, ed., John Wiley & Sons, New York, 1963,[NOR 33]

34. L.H. Putnam, "A general Empirical solution to the Marco Software Sizing and Estimating Problem", *IEEE Trans. Software Engineering*, Vol.SE-4, No. 4, July 1978 [PUT 34]

35. F. Niederman, Staffing and Management of E-Commerce Programs and Projects, Saint Louis University, 2005. [NED 35]

Appendix B: Acronyms & Glossary

Glossary

Non-Rigid Processes – A Software Development process which is highly dynamic in terms of the sequencing of its subtasks. The job logic for the process is defined run time according to the conditions prevalent at that stage.

"Knowledge Loading" can be defined as the amount of knowledge /skill required at a particular time t to ensure the timely completion of the task adhering to the requirements that is ongoing at point t.

"Knowledge Points" can be defined as the metric used for measuring the Skill/Knowledge levels of an individual. These metrics are currently in primordial form and need to be defined once enough data collection is done.

Acronyms

KABASPP – Knowledge an Acquisition Based Approach to Software Project Planning
TRL – Technology Readiness Levels
NASA – National Aeronautics & Space Administration
GAO – United States Government Accountability Office
RAD – Rapid Application Development
DOS – Disk Operating System
SD – Structured Design
CASE – Computer Aided Software Engineering
WBS – Work Break-Down Structure
TWBS – Task Work Break-Down Structure
SCM – Software Configuration Management

Appendix C: Unused References for future Study

O. Bray, M. Prietula, "*The Concept of Skill Management"*, Honeywell Inc., August 1979.

P.J. Denning, *"Educating a new Engineer"*, Communications of the ACM, December 1992, Vol.35, No. 12.

H. Munoz, K. Gupta, D. Aha, D. Nau, *"Knowledge-Based Project Planning"*, IJCAI Workshop on Knowledge Management and Organizational Memories, Seattle, August 2001.

P. Giorgini, S. Rizzi, M. Garzetti, *"Goal-Oriented Requirement Analysis for Data Warehouse Design"*, University of Trento, Italy, 2005.

L. Morgenstern, "*A First Order theory of Planning, Knowledge and Action"*, New York University, NYC, 1986.

A.B. Schwarzkopf, R.J. Mejias, J. Jasperson, C.S. Saunders, H. gruenwald, *"Effective Practices for IT Skill Staffing"*, Communications of the ACM, January 2004, Vol. 47, No. 1.

A.C. Nelson, K. Joshi*, "Effectively Utilizing Systems Developers on Projects"*, School of Business Administration, University of Missouri, 1993.

S. Poltrock et al, "*Information Seeking and Sharing in Design Teams"*, Florida, USA, 2003.

G. Tidhar, A. Rao, M. Ljungberg, D. Kinny, E. Sonnenberg, *"Skills and Capabilities in Real-Time Team Formation"*, Technical Note 27, Australian Artificial Intelligence Institute, July 1992.

S. Henninger, *"Developing Domain Knowledge Through the reuse of Project Experiences"*, University of Nebraska-Lincoln, 1995.