Team Size Optimization – a Management Panacea: Fact or Fantasy

Waqar Ahmad Gulzar

Department of Industrial Engineering, King Abdul Aziz University, Jeddah 21589, Kingdom of Saudi Arabia
00966-503632416 (m); wahmed@kau.edu.sa
Stream: Information System Engineering

Abstract: This paper discusses recent research findings in software engineering concerning the optimum size of teams to be employed during project execution. The unique advantages of software development using water-fall life-cycle method are explored. Optimizing the team sizes for successful completion of projects is discussed with a case example from the armed services taken as a general rule. It has been considered from many years that the optimum team size for several military tasks and similar complex projects, is four. Even so, research efforts on team size optimization is still lively in software engineering area. The author examines the recent research efforts into optimization of team size for testing, defect detection and other stages of waterfall life cycle model. It is concluded that the findings of the author on the likelihood of life-cycle optimal team sizes may exist at some stages, but for other stages the evidence is still inconclusive.

[Waqar Ahmad Gulzar. **Team Size Optimization – a Management Panacea: Fact or Fantasy.** *Life Sci J* 2014;11(6):469-478]. (ISSN:1097-8135). http://www.lifesciencesite.com. 66

Keywords: Waterfall Model, COCOMO, System Design, Metrics, Software, Team Building, Productivity, Optimization,

1. Introduction

The main aim of Software Engineering (SE) is to develop reliable and technologically and economically viable software products. SE techniques attempt to improve the functionality of software and the competence of software developers. A systematic software process should be followed when initiating a software development project in order to achieve the desired quality within the specified time schedule and estimated budget. Researches indicate that 60%-75% of IT projects are failures due to low productivity, exceeding budget expenditure, delayed deliveries, high defects rates and huge maintenance costs indicating that the software product and the development process are deficient and of poor quality. During the '70s, efforts were made to increase software quality focusing on the coding stage of the software development process with no attention driven to better understanding of the system requirements or the better management of the software project. Since early 80s, more attention was given to the areas of specification, design, testing, measurement and management. Consequently various software development methodologies were designed concentrating on the early stages of software development such as structured design methodologies, Rapid Application Development (RAD) methodologies and Agile Development methodologies.

Most important elements that affect software development are management of the project teams who build the system and clear understanding of system objectives. Therefore, this paper investigates the impact of team factors on the software quality specially the effect of team size on the management of software development project that uses the waterfall model as a structured design method. Section two defines the waterfall life cycle and its phase. It also points out the advantages and disadvantages of using this model in software development. Section three addresses the software development project team highlighting its importance and deals with the major concepts about team building and team structure. Moreover, it touches upon the most effective personality attributes of team members, emphasizes the most important skills and knowledge that each team member should acquire depending on his responsibilities in the team and it outlines the regulations and rules used to determine the most advantageous team size for each phase of the waterfall life cycle using Boehm's Cost Constructive Model. Furthermore, this section points out some factors that influence the success of project team management. Finally conclusions are presented regarding the difficulties associated with the methods determining the team size.

2. Materials and Methods

The Waterfall Software Development Life Cycle is the simplest process model and widely used.

"Waterfall model establishes a sequence of stage requirements, specifications, design, coding, testing and maintenance to guide the development process." (Kang, Levy, 1989)

System engineering follows waterfall model because of the need for the parallel development of different part of the system. Its simplicity makes it easy and useful for the developer to know what they need to do. In this model one phase has to be complete before moving on to the next and this cascade where output of one is input to next gives the waterfall model it's name. Two important points have to be considered, verification and validation means that the output of the phase should be consistent with the input and it consistent with the requirements of the system (Jalote, 1991, p. 17). This model is based on two assumptions, software development proceeds linearly from analysis down to coding and the result of each phase are frozen before continue to the next one (Ghezzi et al, 2003, p.407).

2.1. The Advantages and Disadvantages of Waterfall Model

There are many advantages and disadvantages for the Waterfall model system development. A few of them are listed here:

2.1.1 Waterfall Model Advantages

- Good progress tracing due to clear development stages, milestones and deliverables can be clearly identified (Encyclopedia.thefreedictionary.com).
- Follows orderly prioritized stages where the output of each phase is the input of the next phase. This sequential nature enforces organized procedure, which facilitate software construction process, (Als, A. and Greenidge, C., 2003, p.4).
- Documentation is produced after each phase, which will result in a well documented system (Als, A. and Greenidge, C., 2003, p.4).

2.1.2 Waterfall Model Disadvantages

- The entire system requirement should be gathered during the Requirement Gathering and Analysis phase in order to produce a properly designed system. Unfortunately in real life customers keeps on adding requirements even after the end of the Requirement Gathering and Analysis phase. (Parekh, 2005, p.2).
- The problems of each phase are not completely solved during the relevant phase and many others may occur after the phase is ended which will result in badly structured system (Parekh, 2005, p.2).
- It forces developers to make large jumps in the system state during developments, which is not necessarily (Plant, 1991).
- It doesn't reflect the way code is really developed (Pfleeger, 2001, p.50).

2.2 Water-Fall Model Stages

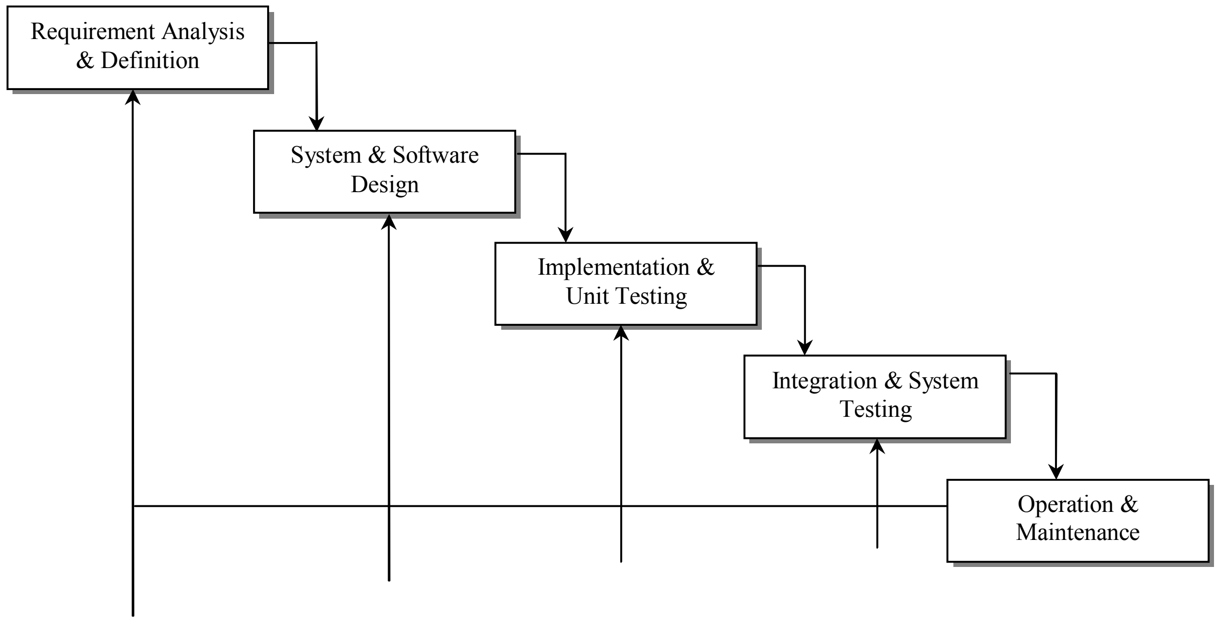
Requirement Analysis & Definition
System & Software Design
Implementation & Unit Testing
Integration & System Testing
Operation & Maintenance

Figure 2.1: The Waterfall Model

2.2.1 Requirement Analysis & Definition: The functionality and constrains which required from the system by the end-user are gathered and analyzed at this stage in order to create the Requirement Specification document which will be the guideline for the next phase of the model (Parekh, 2005, p.1).

2.2.2 System & Software Design: the system design is prepared in this phase though studding the requirement specification from previous phase in order to define the system architecture and specifying the hardware and software requirement (Parekh, 2005, p.1). The system design specification will be the input of the next phase (Parekh, 2005, p.1).

2.2.3 Implementation & Unite Testing: the actual coding is started in this phase after the system design is received and the project is divided to unites, the system development starts by developing the units which will be integrated in the next phase (Parekh, 2005, p.1).

Each unit is tested in order to inshore the requirement verifications (Parekh, 2005, p.1).

2.2.4 Integration & System Testing: unites are integrated into a complete system and the coordination between different unites is tested and the whole system behavior is tested too during this phase to inshore that the pre specified requirement are met in order to deliver the system to the customer (Parekh, 2005, p.1).

2.2.5 Operation & Maintenance: this is the longest and never ending phase since it represent the system life time, after deployment the problems of the system development are exposed during the actual system use and are solved in order to maintain the system functionality and reliability (Parekh, 2005, p.1).

3. Software Development Project Team definition

Software development is a complicated and sophisticated process where long procedural steps are involved, which can't be accomplished by one person effort. Depending on the forgoing the effort of many people is needed to develop a software system which will lead to a team. According to Verma, V., (1997, p.37) team is a group of people working interdependently and that they are generally committed to certain common goals to produce high quality results. Thus team is group of people with special characteristics, in which they have to work in mutually supporting bases in order to achieve a common goal with high quality results. Some basic features team are given in Table 3.1.

Table 3.1: The Basics of Team Building (teamtechnology.co.uk, 1995, p.1)

A group of people	Synergy	Having one aim
Whole > Sum	Co-operation	Flexibility
Working together	Reporting to one boss	Serving one customer

A team may also be defined based on the fundamental features given in Table 3.1 as a synergy group of people working together committed to a single goal and customer at a time, cooperating with each other and reporting to one boss.

3.1. Team Building

Building a team is a process of selecting group of various individuals with different skills, backgrounds and expertise to work together effectively as a team unit to plan work, face challenges, deal with problems, find solutions and deliver results. The knowledge and effort of the team members are merged and directed to achieve the team common goal (Verma, 1997). Project success relies heavily on how the team members work together to achieve the organizational objectives therefore forming and managing a project team effectively is vital for accomplishing the project. The principal members in a software development project team are; the project manager, system analysts, programmers and testers.

3.2. Team Structure

One of Appleton's eleven criteria for successful software project is that a single individual must have responsibility and authority for the project success (DeGrace, and Stahl, 1993). That is why the project manager has a significant role in the software development team. He is responsible for guaranteeing that the project is completed in time, within budget and the system's functions meet the client's requirements. He also has to supervise the quality of the software produced (Dennis, Wixom, 2003). Therefore, one of the main roles of the project manager is to select and manage the project team members. The project manager is responsible for identifying and organizing the tasks, roles, responsibilities and assigning human resources appropriately to carry out these tasks because correct task allocation improves the team productivity and performance. In addition, his role includes developing the project plan and monitoring work schedule. The second team member is the system analyst whose role is to design the new business processes with the assistance of the business analyst who has a business experience and represents the interests of the project sponsors. The business analyst analyzes the business requirements, identifies the business values provided by the new system and helps the system analyst in planning the new policies and processes. The system analyst main interest is to design the information system and makes sure that the new system adheres to the information system principles (Verma, 1997). The tasks of system developers or programmers are to write the source codes and develop the system modules. Testers or defect inspectors set test plan and perform testing (unit testing and system testing) using the software testing metrics and testing techniques. Testers are responsible for the software quality because quality is measured by the defects found in the software therefore most organizations devote money and time on testing to prevent failure caused by the software bugs after the system is installed (Marri, 2010).

3.3. Team Cultural and Personal Diversity

Software projects suffer from poor performance despite the fact that they are provided with the technological tools. Therefore, studies initiate focusing on the human aspect of software development project rather then the technological aspects to enhance the project performance. Studies indicated that Size, compatibility, adaptability, homogeneity are determinants of team effectiveness (Dafoulas and Macaulay, 2001). Hence, more importance is given to personality composition of team members within a software development project since it affects team performance and productivity. Selecting team members of appropriate personality types became a demanding issue. One of the studies finding is that IS managers should consider selecting human resources so there is staff heterogeneity between team members and the project manager. On the other hand, it is better to have homogeneous personality among the team members because some of the team members work together in common tasks like system developers (Gorla and Lam, 2004). Cultural and personal differences affect communication and team stability. Consequently, such differences should be investigated carefully to find out the most effective way to bring the right people together in the same team because members should be able to communicate effectively, cooperate, have same work habits and have collaborative behaviors during problem solving situations. Personality characteristics affect role allocation. When selecting the team members, the project manager should consider the personal characteristics suitable for the allocated role to select the most suitable candidates for effective team performance.

3.4. Team Skills

"In information system development team, skill is defined as the breadth of abilities team members provide a group" (Guinan et.al., 1998)

Team skills are one of the factors that have an effect on the cost, productivity and quality of the software produces. Researches made on the importance of the human factors in addressing the problems of software development showed significant improvement in team effectiveness when using developers with advanced abilities (Guinan, et.al., 1998). Therefore, IS managers are giving more attention to the process of selecting highly skilled team members since it is an element that affects project success. Team skills influence internal processes that directly influence performance. Hence, each team member should possess specific skills depending on his role in the team and the kind of tasks he performs.

3.4.1. Skills of Team Project Manager

Since the project manager is chiefly responsible for managing the project and the team, he should have the following skills and knowledge; leadership, organizational structure, organizational behavior, project planning, project tracking, cost management, human resource management, schedule management, change management, supplier/ subcontract management, communication skills, meeting management skills, negotiation skills, clear understanding of the organization objectives, culture and mission. Project managers usually work as system analysts for many years before being assigned to supervise a project thus they usually have experience of alternative software lifecycles, software metrics, measurement theories and Goal-Question-Metric paradigm (Basili, V. et.al; 1994; Tockey, 2005).

3.4.2. Skills of System Analyst

This moves us to the skills and knowledge to the system analyst who is assigned to system specifications and requirements definition. Consequently, he should possess high analytical skills and the ability to adopt a scientific approach to make decisions based on facts finding and logical methods (Gorla and Lam, 2004). System analyst must have the knowledge of; analysis, requirements engineering, system design, human-computer interaction, usability engineering, software-software/ software-hardware integration, reuse techniques, system analysis and design CASE tools. Moreover, acquiring the knowledge of code optimization, semantics preserving transformations, specific programming languages and debugging techniques help the system analyst performing multiple tasks such as designing programming specifications and programming activities especially in small teams.

3.4.3. Skills of System Programmers

This leads us to the programmer skills and knowledge that he is supposed to acquire. Programming language concepts, data structure concepts, database system concepts, relational algebra, operating system concepts, software architectures, Petri nets, complexity theory, computer graphics, linguistics/ parsing theory, computability theory, set theory, predicate logic, formal proofs and Turing machine theory. In addition to the technical and theoretical programming knowledge, programmers need to have communication skills, experience in specific programming language and domain. Research findings indicate that when programmers can effective communication skills, the team performance is significantly enhanced because they need to communicate with project manager, system analysts, system designers, IT department, and data entry operators and sometimes with system users when they do system analysis tasks in small teams (Grola and Lam, 2004). The programmers' experience in specific software domain and programming language have potential improvements in the quality and cost of the software produced. The programmers' experience in these areas is related to the decrease of the software defects and increase of their ability to detect and debug the errors in less time. Thus, higher level of the programmers' familiarity of the application domain and programming language lead to reduction in rework and cost of software development and maintenance and increase in development productivity (Krishnan, 1998).

3.4.4. Skills of System Testers

The software testers' knowledge and skills are strongly associated with the software quality, too; because they are the last group that reviews the product to ensure it is defect free and meets the required functionalities before it is delivered to the client. Consequently, we can strongly argue that they are responsible for the software quality assurance. Testers should manifest high skills in previews, readiness reviews, walkthroughs, inspections, software project audits, requirements tracing, quality function deployment, software testing techniques, software testing CASE tools, proofs of correctness, process definition/ improvement techniques, statistical process control and technology innovation.

4.0 Results

4.1 Team Size Optimization

One of the factors affecting team productivity, quality of the software produced, software cost estimation and the software development project success is team size. Selecting the optimum team size for each stage of the waterfall software development model is essential for the success of that stage that leads to the success of the next stage because in the waterfall model each phase is dependent on the preceding one. Special programs were developed to estimate the most suitable team size for each phase in the waterfall model such as Costar 7.0. There are three variables that affect the estimation of team size; project size, application complexity and the degree of cost constraints (Putnam, 1997). There are two methods of measuring the project size; Function Points (FP) and Lines of Codes (LOC). Function Points is a way of measuring the system size through counting the amount of functions from the requirements. This method was developed by Albrecht to estimate the software size in an early stage before development and coding stage by identifying five logical components; internal logical files (ILF), external interface files (EIF), external inputs (EI), external outputs (EO) and external queries (EQ) and applying a complexity scale on each component (Flitman, 2003). The Lines of Code method is another method measuring system size. There are two points to be considered when counting the line of codes; identifying the programming language used and identifying and adhering to a counting rule like counting only the Executable Lines of Code (ELOC).

4.1.1 Software Effort and Schedule Estimation Method

In order to specify the team size needed for the project, it is essential to estimate the effort and time needed for developing the project. Therefore, to solve the effort estimation problem, in 1970s, Boehm developed a Constructive Cost Model (COCOMO) using information on cost database of software projects built by an American company considering the economics and engineering perspectives. In the last few years, Boehm developed COCOMO ll model but this study tackles the original COCOMO model only. COCOMO is an algorithmic effort estimation method using the project size measured in thousands of delivered lines of code (KDL) (Pfleeger, 1998). The COCOMO model exists in three stages; basic, intermediate and advanced. In this study, the intermediate stage is used for the effort estimation. The first step is to calculate the initial effort in terms of person-months (PM) using the equation: $E_i = a*(KDL)^b$ where a and b are constants for the different project types. In the intermediate stage of the COCOMO model, there are three categories for software projects; organic mode, semi-detached mode and embedded mode. An organic project is developed in the organization, uses a small team, who is familiar with the development environment, has an experience of the applications developed and there is no constraints on the requirements. Organic project are characterized of being easy. Semi-detached project is harder, the organization uses some outsourcing, the team has less familiarity with the applications, has mixture of experience of the software developed and the requirements have more constraints than the organic project. In embedded projects, there are tight requirements constraints and the team has little experience of the applications developed. Embedded projects are very hard to develop in the organization. The constants ‘a’ and ‘b’ for the different project types are provided in Table 1.

Table 1. Constants for Project Types

System Type	A	B
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

The second step is to calculate the effort adjustment factors (EAF). EAF is a multiplication of fifteen different attributes called cost drivers. Cost driver variables influence the project cost estimation and they vary from project to project. Cost drivers are included in estimating the development effort because projects of the same size do not require the same effort because of other non-technical factors that depend on product, computer, personnel and project attributes. Each attribute has a rating scale and a multiplying factor is provided for each cost driver. Cost drivers and its multiplying factors are usually assumed by the project manager. A list of the cost drivers and its multipliers is provided in Table 2 (Faghih F. 2003).

Table 2. Multipliers for Different Cost Drivers

Cost Drivers	**Rating**					
	Very low	Low	Nominal	High	Very high	Extra high
ACAP Analyst capability	1.46	1.19	1.00	0.86	0.71	*
AEXP Application experience	1.29	1.13	1.00	0.91	0.82	*
CPLX Product complexity	0.79	0.85	1.00	1.15	1.30	1.65
DATA Database size	*	.94	1.00	1.08	1.16	*
LEXP language experience	1.14	1.07	1.00	0.95	*	*
MODP Modern programming Practice	1.24	1.10	1.00	0.91	.82	*
PCAP Programmer capability	1.42	1.17	1.00	0.86	.70	*
RELY Required software reliability	0.75	0.88	1.00	1.15	1.4	*
SCED Required development schedule	1.23	1.08	1.00	1.04	1.10	*
STOR Main storage constraint	*	*	1.00	1.06	1.21	1.56
TIME Execution time constraint	*	*	1.00	1.11	1.30	1.66
TOOL Use of Software tools	1.24	1.10	1.00	0.91	0.83	*
TURN Computer turnaround time	*	0.87	1.00	1.07	1.15	*
VEXP Virtual machine experience	1.21	1.10	1.00	0.90	*	*
VIRT Virtual machine volatility	*	0.87	1.00	1.15	1.30	*

The third step is to calculate the total effort by multiplying the initial effort by effort adjustment factor: E= EAF * E_i. Estimating the total effort needed for system development enables the project manager to estimate the project overall cost but managers usually need more detailed estimation for planning and checking purposes. COCOMO provides a distribution of the percentage of total effort for each phase in the waterfall model depending on the type and size of the system. The phase effort is estimated by multiplying the total effort by the phase effort percentage. Table 3 shows the phase-wise distribution percentage of effort of an organic mode, Table 4 shows effort distribution of semidetached mode and Table 5 shows effort distribution of embedded mode (Faghih F. 2003).

Table 3. Phase Distribution of Effort: Organic Mode

	Size			
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plan & requirements	6%	6%	6%	6%
Product design	16	16	16	16
Detailed design	26	25	24	23
Code & unit test	42	40	38	36
Integration & test	16	19	22	25
Total	100	100	100	100

Table 4. Phase Distribution of Effort: Semidetached Mode

	Size			
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plan & requirements	7%	7%	7%	7%
Product design	17	17	17	17
Detailed design	27	26	25	24
Code & unit test	37	35	33	31
Integration & test	19	22	25	28
Total	100	100	100	100

Table 5. Phase Distribution of Effort: Embedded Mode

	Size			
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plan & requirements	8%	8%	8%	8%
Product design	18	18	18	18
Detailed design	28	27	26	25
Code & unit test	32	30	28	26
Integration & test	22	25	28	31
Total	100	100	100	100

In order to calculate the number of team members needed for each phase, we need to estimate the duration for each phase. The first step is to estimate the total project duration or time of development in terms of months using COCOMO schedule equations. For organic mode, the equation is $D=2.50(E)^{0..38}$, for semidetached mode, the equation is $D=2.50(E)^{0..35}$ and for embedded mode, the equation is $D=2.50(E)^{0..32}$. Tables 6, 7 and 8 display the phase distribution of schedule under these three modes respectively. The second step is to calculate each phase duration by using the percentages for the different phases of the waterfall model provided in Table 4. In this table, the detailed design, coding and unit test are combined into the programming phase because these activities are done by the programmers. The final step is to estimate the number of members in each phase by dividing the phase effort by the phase duration (Faghih F. 2003).

Table 6. Phase distribution of Schedule: Organic Mode

Size				
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plans & requirements	10%	11%	12%	13%
Product design	19	19	19	19
Programming	63	59	55	51
Integration & test	18	22	26	30
Total	100	100	100	100

Table 7. Phase distribution of Schedule: Semidetached Mode

	Size			
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plans & requirements	16%	18%	20%	22%
Product design	24	25	26	27
Programming	56	52	48	44
Integration & test	20	23	26	29
Total	100	100	100	100

Table 8. Phase distribution of Schedule: Embedded Mode

	Size			
Phase	Small 2KDL	Intermediate 8KDL	Medium 32KDL	Large 128KDL
Plans & requirements	24%	28%	32%	36%
Product design	30	32	34	36
Programming	48	44	40	36
Integration & test	22	24	26	28
Total	100	100	100	100

4.2 Example of Using COCOMO Metrics

An example to apply the COCOMO model is to estimate the effort and time needed to design and develop a registration system in a college. According to the requirements analysis, the system will consist of seven modules: data input, data output, data update, query, schedule, transcript and report. Depending on the requirements, the system is regarded to be organic. The size of the system is estimated according to the sizes of the modules: data input.75 KDL, data output 3.5 KDL, data update 1.6 KDL, query 3.4 KDL, schedule 2.5, transcript 10 KDL, report 5.5 KDL and the total size is 27.25 KDL. So it is a medium size system. The project manager assessed the cost drivers attributes and their ratings based on the requirements as the following: database size is high 1.16, software reliability requirement is high 1.15, analyst capability is very high.71, use of SW tools is low 1.10 and the rest of the cost drivers are set to nominal. The effort adjustment factor (EAF) is EAF= 1.16 * 1.15 *.71 * 1.10 = 1.04. Using the previous data, we can estimate the initial effort: $E_i = 2.4 * (27.25)^{1.05} = 77.15$ PM. Using the two values, we can estimate the total effort: E= 1.04 * 102.86= 80.24 PM. Using the total effort estimate, we can get the effort in each phase by using values in table 3. Since the project size is 27.25 KDL, interpolation is used to estimate the percentage for this project size and the end two values for interpolation are the percentages for 8 KDL and 32 KDL. The percentages for this project phases: plans & requirements 6%product design 16%, detailed design 24.197%, code & unit test 38.396%, integration & test 16.59%. The effort estimates for the different phases: Plans & requirements.06 * 80.24 = 4.82 PM, Product design.16 * 80.24 = 12.84 PM, Detailed design.24197

* 80.24 = 19.42 PM, Code & unit test.38396 * 80.24 = 30.81 PM, Integration & test.1659 * 80.24 = 13.31 PM.

For schedule and staff requirement estimation, we have to obtain the total duration in terms of months and the duration for each phase. Using the total effort, we can estimate the total duration: $D = 2.50(80.24)^{0.38} = 13.23$ M. To get the duration for each phase, we will use the values in table 4 and again we will use interpolation to get the percentage for this project size. The percentages for this project phases: plans & requirements 11.80%, product design 19%, programming (includes both detailed design and code phases) 55.79%, integration and test 25.21%. The duration estimation for each phase: plans & requirements.1180 * 13.23 = 1.56 M, Product design.19 * 13.23 = 2.51 M, Programming.5579 * 13.23 = 7.38 M, Integration & test.2521 * 13.23 = 3.34 M.

Now we can estimate the staff needed for each phase: plans & requirements 4.82/1.56 = 3 P, Product design 12.84/2.51 = 5.1 P, Programming (19.42 + 30.81) /7.38 = 6.8 P, Integration 13.31/3.34 = 3.9 P.

4.3 Advantages of Optimal Team Size

A research study was conducted on medium-sized information system projects to find the optimum team size. The outcomes of the research were that small teams consisting of 5 to 7 persons were more productive, had best schedule performance, used fewer person-months and of course required less cost than larger teams consisting of more than 8 people (Putnam and Mayers, 1998). One of small team advantages is communication simplicity. As the size of team grows, the communication paths increase and there are more chances for communication errors. Communication paths increase multiplicatively and the formula for computing the number of possible interactions (I) is: $I = K (K-1)/2$ where K is the number of people in the team. For example, using this formula, a team of 10 persons has 45 paths; $I= 10(10-1)/2= 45$ (Fried, 1991). When communication interactions increase, communication time increases and team productivity decreases. Large teams use massive quantity of formal documents as a means of streamlining communication and at the same time to keep records on the project developed whereas for small teams e-mail records is sufficient alternative for formal documentation (McConnell, 1997).

5. Discussion

Project Team Management: In order to complete a software development project successfully, the project team must be managed effectively. Hence the project manager administering technique has an influence on the team performance and productivity. Listed below a number of the most important success factors that the project manager need to consider when managing the project team:

Role Assignment: A project manager must assign specific role to each team member and clearly define tasks and responsibilities. Role assignment allows each team member to focus on his work and identify his share of responsibility in the developed product. It also allows the manager to monitor and measure each member's performance and interfere in the right time to make the suitable corrective decisions to improve the performance of the whole team (Fried, 1991).

Schedule Development and Assigning Realistic Deadlines: In order to avoid missing deadlines and milestones, the project manager should set realistic objectives, expectations and deadlines. He or she must make sure that all the team members are clearly aware of the commitments made to users and stakeholders (Reel, 1999). To protect the project from schedule slips and overruns, a project manager must plan and control schedule activities by estimating how long each activity will take and strictly abide to the allocated time frame.

Communication: Effective communication among team members is an important factor of project success. A project manager should make sure that his team members communicate with each other to resolve their mutual problems. Therefore, he must establish appropriate communication channels to distribute information and make it available in a timely manner. Despite the usefulness of information technology communication tools such as e-mail, a project manager should not rely on them. Moreover, he must work on removing communication obstacles and barriers to build open and effective communication by frequently organizing meetings among team members themselves and between the project team and users. Efficient communication promotes cooperation and better understanding of the users' requirements hence producing software that actually meets their needs (Verma, 1997).

Workload Distribution: A project manager should be able to divide the work load into small manageable tasks and distribute them on the team members in a way not to overload or stress them giving each member a space to be creative.

Motivation: The project manger ability to motivate his team members throughout the project is vital to sustain high level of team performance. A good motivation act is to make people feel that their work is appreciated and recognized. A project manager should involve his team members in making decisions that affect them. Also the project manager should create comfortable work environment for his team providing them with lighting, desk space and any technological equipments they need to finish their work (Dennis and Wixom, 2003).

Teamwork: In order to avoid having team members working as separate individuals, the project manager must encourage teamwork. Through good communication and interpersonal skills, the tam manager should be able to understand the feelings and sentiments of each individual in his team to develop the appropriate strategies so that he could create effective teamwork to accomplish project objectives successfully.

6. Conclusions

Software Engineers realized the advantages of using small teams in software development projects therefore there is a growing tendency of developing methods and software to estimate the most optimum team size. This paper investigated the relationship between team size and successful project management exploring the factors that can contribute to diminishing team size such as team skills, selecting team members with the right personality attributes and team experience. It suggested that highly skilled team shows more production and relationship oriented processes. Advanced personnel capability of the team is associated with increased productivity and quality and decreased development and maintenance cost and time and effort. Experienced team stabilizes project quickly with less coordination. It was found out that increasing team size diminishes productivity whereas a small team requires less time schedule, less communication complexity, uses less development effort, requires less management overhead and uses less formal documentation. In addition, this paper demonstrated one of the software metrics COCOMO used to estimate software effort and schedule as a means to estimate team size and software cost for the waterfall model. Selecting the optimal tam size for each stage of the waterfall model is a key element to a successful software project (on time, on budget, with good quality) using waterfall as a software development method. Team size affects team dynamics, team productivity, software quality and cost estimation. In short team size optimization is in fact a management solution. Software development is a human endeavor so it is the efforts of the software development team that make it possible to deliver a reliable quality system. Hence, further research studies need to be done to find out methods that help in deciding the optimal team size and better software project team management.

7. Acknowledgements

The author expresses his deep sense of gratitude to the Department of Industrial Engineering and the Faculty of Engineering, King Abdulaziz University, Jeddah for extending their whole hearted support to this research.

References

1. Als A, Greenidge C. Waterfall Model (2003); Available at: http://scitec.uwichill.edu.bb/cmp/online/cs22l/waterfall_model.htm. Accessed on 28 August 2005.
2. Basili V., Caldiera G., Rombach H.D. The Goal Question Metric Approach (1994); Encyclopedia of Software Engineering; John Wiley & Sons, Inc. pp. 528-532. ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf. Accessed on 12 November 2008. Also available at http://www.cs.umd.edu/~basili/publications/technical/T87.pdf
3. Dafoulas G, Macaulay L. Facilitating Group Formation and Role Allocation in Software Engineering Groups. ACS/IEEE International Conference on Computer Systems and Applications (2001); pp.25-29 June, IEEE Computer Society Beirut, Lebanon (TCSE & TCCA), ISBN 0 7695 1165 1.
4. DeGrace P, Stahl L. The Olduvai Imperative The Olduvai Imperative: Case and the State of Software Engineering Practice (1993); Yourdon Press Computing Series Englewood Cliffs, N.J. ISBN-10: 0131611003; ISBN-13: 978-0131611009
5. Dennis A, Wixom B. System Analysis & Design (2003); John Wiley & Sons, Inc. United States of America.
6. Faghih F. Software Effort and Schedule Estimation. (2003) http://www.enel.ucalgary.ca/People/Smith/619.94/prev689/1997.94/reports/farshad.htm. Accessed on 28 March 2005.
7. Flitman, A. Towards meaningful benchmarking of software development team productivity. Benchmarking: An International Journal (2003); 10(4), pp. 382-399.
8. Fried, L. Team size and productivity in systems development. Information Systems Management (1991); 8(3), pp. 9-27.
9. Ghezzi C, Jazayeri M, Mandridi D. Fundamentals of Software Engineering. 2nd Ed. (2003); Prentice Hall/Pearson Education, International, Upper Saddle River, New Jersey. ISBN-10: 0133056996; ISBN-13: 978-0133056990
10. http://encyclopedia.thefreedictionry.com/waterfall+model. Accessed on 24 March 2005.
11. Jalote P. An Integrated Approach to Software Engineering with 86 Illustrations. (1991); New Delhi: Narosa Publishing House.
12. Kang K.C, Levy L.S. Software Methodology in the Harsh Light of economics. Information and Software Technology (1989); 31(5), June Pp.

239-250. http://www.sciencedirect.com/scince. Accessed on 21 March 2005.

13. Krishnan M. The role of team factors in software cost and quality: An empirical analysis. Information Technology & People, (1998); 11(1), pp. 20-35.
14. Marri K.K. Models for evaluating reviewz effectiveness (2010); Proceedings, The Third International Conference on Software Testing, Verification and Validation, ICST 2010, Paris, France, April 7-9, 2010; IEEE Computer Society. ISBN 978-0-7695-3990-4.
15. McConnel S. Less is More: Jump-Start Productivity with Small Teams (1997); Software Development, October 1997, pp. 28-34. Also available at: www.stevemcconnell.com/articles/art06.htm
16. Parekh, N. The Waterfall Model Explained. (2005); www.buzzle.com/editorials/1-5-2005-63768.asp Accessed on 30June, 2010.
17. Putnam D. Team Size Can Be the Key to a Successful Project: Perspectives. Quantitative Software Management (1997); 20(1), pp. 1- 4.
18. Reel J.S. Critical Success Factors in Software Projects. (1999); IEEE Software Volume 16 Issue 3, May-June 1999 Page 18-23. IEEE Computer Society Press Los Alamitos, CA, USA DOI: 10.1109/52.765782
19. Gorla, Narasimhaiah., Lam, Yan. Building Effective Software Project Teams (2004); Communications of the ACM, 47(6), pp. 1-8.
20. Guinan, Patricia, Cooprider, Jay, Faraj, Samer. Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach. (1998); Information Systems Research, 9(2), pp. 101-125.
21. Pfleeger, S.L. Software Engineering: Theory and Practice 2nd Edition. (2001); Prentice Hall, Upper Saddle River, New Jersey. ISBN:0130290491
22. Pfleeger, S. L. Software Engineering: Theory and Practice 1st Edition. (1998); Prentice Hall, Inc. Prentice Hall, Upper Saddle River, New Jersey. ISBN:0-13-624842-X
23. Plant R.T, Rigorous Approach to the development of Knowledge-Based System. Knowledge-Based Systems (1991); 4(4), December, pp. 186-196. http://www.scincedierct.com/science. Accessed 24 March 2005.
24. Putnam, L, Mayers W. Selecting the Right Team Size: Small is Beautiful. (1998); http://www.cutter.com/research/1998/crb981222.html visited 2006-05-05. Accessed on 11 July 2005.
25. Team Technology. The Basics of Team Building (2005); http://www.teamtechnology.co.uk/tt/h-articl/tb-basic.htm Accessed on 13April 2005
26. Team Technology. Team Building - A Complete Guide (1995); http://www.teamtechnology.co.uk/tt/h-articl/tb-basic.htm Accessed on 24 March 2014.
27. Tockey, S. Recommended Skills and Knowledge for Software Engineers. 3rd Edition. (2005); Software Engineering. Volume 1. The Development Process.
28. Thayer R.H. Christensen M.J. Software Engineering, The Development Process, Vol. 1. 3rd Edition. (2005); Wiley, John & Sons ISBN: 0471684171, ISBN-13: 9780471684176
29. Verma, V. Managing the Project Team: The Human Aspect of Project Management, Vol. 3. (1997); Project Management Institute, Inc., United States of America. ISBN-10: 1880410427; ISBN-13: 978-1880410424.

5/10/2014