Towards Selecting Software Development Team Members using Task Analysis

K. Otokiti[1], S. C. Chiemeke[2] and Kingsley Chiwuike Ukaoha[3*]
Department of Computer Science,
University of Benin, Benin-City,
Nigeria.
Email: [1]kareem.otokiti@uniben.edu, [2]schiemeke@uniben.edu, [3]kingsley.ukaoha@uniben.edu
**corresponding author*

ABSTRACT

The vision of building successful software products require individuals in teams equipped with a wide range of social and technical skills assigned appropriate job roles. Team compositions and the method used in selection affect team process and, ultimately, the different dimensions of team effectiveness. To achieve successful results, these individuals should be selected for appropriate roles based on the skills mentioned, which also usually creates a kind of dependency on their tasks. Wrong team selections may result in wrong process being applied and a wrong software product developed at the end. We prepared and administered 30 questionnaires to selected experts in software development of which 20 were found useful and used in our study which were further analyzed with the SPSS (Statistical Package for Social Sciences) to compute mean and simple percentages of responses. Our result shows that it is best to utilize task analysis as a tool in selecting individuals that best fits a given software developmental task role in a bid to optimizing skills to roles and hence contributing to the overall success of the software development process.

Keywords: *Project management, selecting team members, project analysis, software development*

Reference Format:

Otokiti, K., Chiemeke, S. C. and Ukaoha, Kingsley Chiwuike (2020), Towards Selecting Software Development Team Members using Task Analysis,
Afr. J. MIS, Vol. 2, Issue 1, pp. 55 - 65.

1. INTRODUCTION

Software process is a set of activities, methods, practices and transformations that people use to develop and maintain software and the associated products (Paulk et al., 1995). It encompasses the total set of engineering activities needed to transform a user's requirement into software system (Chiemeke and Egbokhare, 2013). A software project can be termed a failed project if it is over budgeted (cost overrun), delivered late, of poor quality, does not meet user's satisfaction and/or abandoned. Failures are rarely caused by mysterious factors which are usually discovered post-mortem, or only after it's too late to change direction (Lorin, 2000).

The Information Technology Association of Nigeria (ITAN) has estimated Nigeria's IT market at over 150 billion USD with anticipated growth of 10% over the global average (Mohammed and Abraham, 2013). CHAOS report (2009), revealed that only 32% of IT projects were considered successful, 24% of the IT projects were considered to be failed projects while 44% were considered challenged IT projects.

Mohammed and Abraham (2013) pointed out some notable IT projects that have failed despite the huge amount the Nigerian government has invested. Some of these projects are the African Development Bank (ADB) which cancelled 80% of her project in Nigeria due to project failure, the Energo Nigeria Limited transmission substation achieved less than 5% implementation in year 2008 and one of the most epic is the failed multi-million dollars Nigeria Communication Satellite (NigCom Sat1) project that was replaced with NigCom Sat-R1 after the former installed in 2007 failed which was supposed to have 15 years duration but lasted for 18months. Unfortunately, even with experiences from the past, the nation still experiences failed projects.

The primary factors that have led to failed projects in Nigeria are poor planning, lack of top management support, failure to address risks areas, inadequate skill and expertise of IT project manager (Mohammed and Abraham, 2013). Over time, several literatures have revealed that there have been records of software failures and most of these are caused by but not limited to poor user input, stake holder conflicts, vague requirements, poor cost and schedule estimation**,** failure to plan, communication break downs, poor architecture, late failure warning signals and most importantly with respect to this work, skills that do not match the job.

The successes of information system development projects are usually affected by the interaction of various factors. These factors collectively referred to as the three P's: Problem, Process and People have the People factor at the head of the pyramid as depicted in the software development triangle (Chiemeke and Egbokhare, 2013). A critical look at the software development triangle from Egbokhare and Onibere (2008) depicts "people" as the most important factor of the Software Development Environment (SDE) and because of its neglect and rather, focus on process and product by researchers, has led to one of the major causes of failures in software development project. Passova (2005) viewed the software development process as a multi-level human communication channel where the problem originates from people (customers), creative and knowledgeable people (developers) use available tools and technology (process) to build the required software systems and the process is also managed by people (manager).

Researchers also noted that human resource is also incurred the same way as physical and technical resources (Tsai et al, 2003). In software organizations perspective, human resources play a critical role in software project success or failure but over the years, people continue with paying least interest to the human resource selection but rather tend to focus more on the technical side. Non competent or defective (Unskilled) people assignment to software projects leads to the main human factor-related issues affecting software project success. This also sometimes leads to the problem of non-fulfilment of stipulated delivery periods (Linberg, 1999). Effective teamwork allows teams to produce outcomes better than the sum of individual members' contributions. As such, there is a compelling need to use the best method in selecting competent team members (Lacerenza et. al, 2018).

In Nigeria, most of the factors responsible for the high rate of failures in software development projects can be traced to improper management of the interactions and roles of the human resource (Egbokhare and Onibere, 2008). Also, in software engineering, the software development team is the most important resource available and their enthusiastic contribution can determine the success or failure of a software development project (Mohamed et. al, 2013).

Developing high-quality products continues to be a major struggle for software companies to maintain and expand while major contributions to the outcome of software projects depend on decisions taken on the assignment of personnel (Otero et al. 2009).

The aim of this research work is to apply task analysis in aiding the selection of personnel of the software development teams. This study can help serve as guide to selecting the right actors (development team) involved in software development projects in a bid to contributing to the success of such projects and curbing catastrophic, gargantuan or hazardous loss that may arise from its failure.

2. RELATED WORKS

The genesis of modern day task analysis methods and techniques can be traced back to the scientific management movement in the early 1900's (Annett, 2004).

The 1950s gave rise to new theories of human performance in systems and new ways of assessing human activities in system design.

However, the actual birth of task analysis can be tied to the study performed at the Tavistock Institute of Human Relations where they researched on the relationship between the social and technical subsystems of an organization and how these subsystems interact (Trist & Murray, 1990). This led to a more rigorous, systematic and cost effective analytical techniques which influenced the emerging interdisciplinary practice of Human Computer Interaction (HCI). As computing power expanded, HCI came to encompass vast new areas of human behavior, and task analysis took on greater scope and complexity. Task analysis now includes a range of techniques aimed at obtaining descriptions of what people do, representing those descriptions, predicting difficulties, and evaluating systems against functional requirements. These techniques focus on quite different levels of analysis and contribute different insights.

Tsai et al., (2003), in their paper described the selection of human resources using Critical Resource Diagram (CRD) method and Taguchi's parameter design approach. CRD methods focus on resource scheduling rather than the activity scheduling and selection of human resource. Their method also did not focus on selection method of efficient human resources for fulfilment and delivery of software project in timely manner to reduce the cost.

Past literature regarding the assignment of software developers emphasized more on human capabilities in software development by stating that experience of manager, heuristic knowledge and instinct help managers to decide developers (Acuña and Juristo, 2004).

The Grey decision model for selecting contractors was proposed by Norita and Laplante (2006), but the contractors may have uncertain, undefined, fuzzy and vague attributes which may lead to complexity in the selection of contractors. In this scenario Grey model works satisfactorily but it does not meet the human resource selection with fixed criteria.

An algorithm using the Knowledge Modelling and Description Language (KMDL) to process a team composition for specific task by analyzing the knowledge and skills of the employees needed for software development was developed by Gronau et al. (2006) while Strad and Guid, (2010) in their research "A fuzzy-genetic decision support system for project team formation", focused on flexibility of requirements specification using special format that expresses the required team capabilities using fuzzy descriptors. But they also did not focus for the selecting developers based on project specific skill set.

Cognitive task analysis is applied to modern work environments such as supervisory control where little physical work occurs, but the tasks are more related to situation assessment, decision making, and response planning and execution (Hoffman et al., 1998).

Task analysis is also used in education. It is a model that is applied to classroom tasks to discover which curriculum components are well matched to the capabilities of students with learning disabilities and which task modification might be necessary. It discovers which tasks a person hasn't mastered, and the information processing demands of tasks that are easy or problematic. In behavior modification, it is a breakdown of a complex behavioral sequence into steps. This often serves as the basis for chaining.

According to a recent survey conducted by Deloitte across 130 countries and over 7,000 participants, the number one global workforce trend is teamwork (Kaplan et. al, 2016). The concept of task analysis is used in system designs to help break down the various steps and requirements in every stage of the system development by ensuring that the best hands are selected thereby helping to avert failures and ensuring the project success.

Most times, candidates with the required skills to work on specific tasks are not available, and the decision to assigning resources (individuals) to task is not certain and can be very fuzzy hence, taskmakers are forced to assign resources to tasks based on subjective measures.

The typical scenario is that when a software project is to be executed, the project manager picks from available team members the person he/she subjectively deems fit to handle a particular task. This decision may be guided by perceived experience from previous projects and in cases where they lack required skill for a particular task involved in the development; they seek the services of external developers to help with development. Software managers typically assign people to roles based on "their experience, heuristic knowledge, subjective perception and instinct" (Acuna et al., 2006).

Tsai et al. (2003) proposed selecting resources using the Critical Resource Diagram (CRD) method and the

Taguchi’s parameter design approach. The CRD method focuses on resource scheduling rather than activity scheduling to represent human-resource workflow and tasks’ precedence. Their method also did not focus on selection method of efficient human resources for fulfilment and delivery of software project in timely manner to reduce the cost. Deviations from target values resulted in additional costs.

Another methodology used to evaluate staffing alternatives is the Analytical Hierarchy Process (AHP). The AHP is a structured technique for organizing and analyzing complex decisions, based on mathematics and psychology. It represents the most accurate approach for quantifying the weights of criteria. Individual experts’ experiences are utilized to estimate the relative magnitudes of factors through pair-wise comparisons. Each of the respondents has to compare the relative importance between the two items under special designed questionnaire (note that while most of the surveys adopted the five point likert scale, AHP's questionnaire is 9 to 1 to 9, (Li et al, 2019). Ho-Leung (2010) used AHP to tackle the problem of human-resource substitution, considering several organizations, client, and application attributes. The proposed model did not consider relationships between known and required skills. The author mentioned the importance of developing faster methods for human-resource substitution.

Antoniol et al. (2004) used queuing theory and stochastic simulation to study staffing needs for software maintenance problems. The authors proposed modeling maintenance processes as first-in-first-out (FIFO) queues in which incoming requests are passed to first available maintenance teams. Since most of the process was standardized, with some portion of it being automatic, different programming skills and experiences were not included in the decision making process.

Considering the importance of human resource to the success of software development, it is necessary that a formal method be defined, rather than the subjective method used by software managers for team selection. Although some of the existing systems have their advantages, they possess some of the following constraints, most of which revolve around the ability to select suitable team members for a given software development task:

i) Subjective and heuristic measures for human resource (team) selection for software development projects.
ii) Non focus on selection method of efficient human resources for fulfilment and delivery of software project in timely manner to reduce the cost.
iii) Non consideration of the relationships between known and required skills needed to carry out task in the development process.
iv) By passing incoming requests to first available maintenance teams, different programming skills and experiences were not included in the decision making process.

Based on the demerits of the existing systems, we propose a Task Analysis based framework with the aim of handling the aforementioned drawbacks.

3. MATERIALS AND METHODS

Our proposed Task Analysis method is designed in six (6) phases. The first phase is identifying and describing the required role/position for a given software development team. This is followed by describing the associated functions and task to be performed by the role personnel. After which a preliminary list of task and (Knowledge, Skills and Ability) KSA statements is developed. Experts in the software development fields are consulted by way of interviews and questionnaires to discuss, review, revise, and augment the preliminary list of task and KSA statements. These experts also help to identify critical task associated with roles and to establish the relationship between the essential tasks and the important KSAs required upon entry for successful job performance.

The proposed method will ‘attempt’ to resolve the constraints of the existing ones analyzed. With the ability of the task analysis method to break down complex task to its constituent sub tasks with a view to exposing or matching the required Knowledge, Skill and Ability (KSA) required to perform the critical subtasks needed for the successful completion of a give task role. The task analysis technique is able to ensure that the best fit candidate is selected as a team member for a given software developmental project.

4. RESEARCH DESIGN

The main instrument used for the data collection was a questionnaire consisting of three (3) sections. Section A captures the demographic information of the respondents, Section B consists of Task Statements and respondent’s opinion of the criticality and frequency of each task statement and Section C captured respondents view of the Knowledge, Skills, Abilities and their relationship to the successful performance of a selected task role. The population of the study consisted of 30 software development experts and 30 questionnaires were distributed, but 20 were found fit and used for analysis. We relied more on expert opinions from experienced developers who have been in the business of software development for many years and thus, have better experience to share. The experts were located in the ICT units of nearby universities and software training houses. Oral interviews were also conducted to get other relevant information.

Data gathered from the respondents were analyzed based on the observed and expected frequencies. Data entry and analysis was done using SPSS to compute the mean and simple percentages of the data captured.

The task statement for the selected role (Software Architect) for this study was gotten from The Carnegie Mellon University Software Engineering Institute Digital Library (USA) and was vetted by experts in the field of study. The Questionnaire respondents rated each task statement on its criticality and relative time spent performing each task. Independent rating scales were used to evaluate each task statement

Task Scale A Criticality: This scale first identifies the essential functions of the job and then measures the importance of those essential functions to overall successful job performance. Scale responses range from (1) Trivial, (2) Important to (3) Critical.

Task Scale B Frequency of Performance: This scale assesses the relative time spent performing each task of the job. Scale responses range from (1) Not Performed, (2) Rarely Performed, (3) Occasionally Performed to (4) Frequently Performed.

KSA Rating Scales: Questionnaire respondents rated each KSA statement on its importance, the amount of the KSA required upon entry to the job, and the relationship between the amount of the KSA possessed and increased job performance. Independent rating scales were used to evaluate each KSA statement. Scale A was for Criticality, Scale B for When Required, Scale C for Relationship to Job Performance**.**

5. RESULTS AND DISCUSSION

Result of task frequency and task criticality is depicted in Figure 1a and Figure 1b.

The defined task statement of Identifying and interacting with stakeholders to make sure their needs are being met showed that the entire valid sample size (100%) agreed it is a frequent task for the software architect and that the task was important to successful role performance.

18 (90%) of the total 20 respondents were of the opinion that the task of crafting the right architecture to solve a software development problem is a frequent task performed by the Software architect. 25% of the population agreed that this task was important and 75% agreed that this task was critical to job success.

Results analysis also revealed that the task of resolving disputes and making tradeoffs was a rarely performed task with a total of 80% of the respondents in agreement and a total of 12 (60%) of the 20 respondents said it was an important task.

Results also show that the task of managing risk identification and risk mitigation strategies associated with the architecture was an occasional task with 80% of the respondents in agreement and also this task was rated as a critical task by a total of 12 (60%) respondents.

Of the total valid population, results of analysis show that there was an even tie in respondents’ view as to the task of the software architect giving input as needed to issues like tool and environment selection. 50% of the respondents said it was an occasionally performed task while the other 50% said it was a frequently performed task. A total of 16 from the 20 respondents said it was an important task. 90% of the respondents said it was a frequent task for the SA to make sure the software and system architectures are in synchronization and most of the respondents (80%) said it was a critical task required for project success.

The task of making sure that the software architecture is not only the right one for operations, but also for deployment and sustainment was seen by 80% of the respondents as frequently performed and 80% a critical task. 70% of the respondents were of the opinion that the SA occasionally Maintains morale, both within the architecture group, and externally as well. 70% of the respondents also considered this a critical task required for project success.

Making sure the right modeling is being done, to know that qualities like performance are going to be met was considered an occasional task by 50% of the respondents while 70% were in concord it was an important task.

Results also revealed that a high percentage (70%) of the respondents said the task of understanding and representing the various disparate views of information, documenting actual or probable structural issues; reverse-engineering architectures from existing structures is an occasional task and also 12 (60%) of the total respondents said it was a critical task.

Results analysis of when a given knowledge is required and the knowledge criticality to project success is depicted in Figure 2a and Figure 2b.

Inter modular interaction and necessary middleware skills for distributed components was considered by 55% of total respondents as a knowledge required when 60% agreed that it is an important knowledge needed for successful job performance.

The Working knowledge in High-End databases was considered not required by 60% of total respondents and also a total of 60% of the respondents said it was a trivial knowledge required for successful job performance. The knowledge of use of latest tools was considered by 80% of respondents as a knowledge need when hired and it was critical to job performance as pointed out by 75% of the total valid respondents.

From our results, we found out that 90% of respondents were of the view that Computational, technical knowledge and deep process definition knowledge is required when hired as it is vetoed critical by 85% of respondents were of opinion that successful job performance. Computational analysis reveals that 75% of respondents agreed that knowledge on Architectural Patterns and knowledge on Design patterns is required when hired and also 85% of respondents deemed it critical for successful job performance.

Results analysis of when a given skill is required and the skills criticality to project success is depicted in Figure 3a and Figure 3b.

Our results showed that 95% of respondents believed the Software Architect should have good grip on non-functional areas like performance, scalability studies, human interaction, and software ergonomics as 65% said it was an essential skill needed for job performance.

Results reveal 75% of the total respondents are of the view that the software architect requires the developers' capability skill when hired as also a total of 60% are of the opinion that it is an important skill required for successful job performance.

From our results, we found out that about 90% of the total respondents were of the view that excellent communication skill is need when hired and 80% were also of the opinion that it is a critical skill required for successful job performance.

Respondents opinion was about 90% of total sample size in favor as to the SA's endurance in repeating the same idea over and again was a skill required when hired and 75% said it was an essential skill required for successful job performance.

The creativity of finding simple solutions to complex issues was a skill 85% of the respondents said the SA required when hired and 70% said it is an essential skill required for successful job performance.

Results analysis of when a given ability is required and the ability criticality to project success is depicted in Figure 4a and Figure 4b.

The ability to extract the non-functional requirements from the user was considered by 90% of respondent as one required from a Software Architect when hired and also 90% of the respondents said it was a critical ability needed for successful job performance.

Of the total respondents as shown from results, 75% are of the opinion that the ability to adopt and understand new technology to apply in business is a desired ability needed when hired and 80% of respondents said it is important for successful job performance.

The software's architect ability to test and analyze complex reports to ensure that quality goals are reached was seen by 95% of the sample size as a skill required when hired and also, 90% of respondents saw it as a critical ability for project success.

Our results also revealed that 85% of respondents believed in the ability of a Software Architect to assist developers in resolving the technical issues and providing proof-of-concepts on newer technologies when hired and 50% said it was important to project success.

The ability of a Software Architect to assist developers in resolving the technical issues and providing proof-of-concepts on newer technologies was considered by 75% of total respondents as an ability required when hired ad 70% said it was essential. All of the respondents agreed that this ability was needed for successful job performance.

Considering the important role of Task analysis in breaking down a given task into its subtask, analyzing them in terms of their frequency and criticality and also defining the knowledge skills and abilities required to perform such task gives a clear map or guide as to what is needed to successfully perform the said task.

Result from the case study position (Software Architect) analyzed in this work shows that the role of a Software Architect (Using the herein defined task statement) should be assigned to an individual who is able to perform well in the identified critical tasks and has sound bearings in required knowledge, skills and abilities that have been identified as critical with high relationship to successful job performance.

The assignment of the task role to such an individual implies matching the required skill to the defined job which can in turn improve performance and thus successful implementation of the associated task.

6. CONCLUSION

The degree of loss occasioned by failed software projects is gargantuan. It has slowed down achievement of technological advancements and there is the need to proffer solutions to failure factors. Task analysis has been able to serve as a tool that provides guidance to Software Managers and Human Resource persons on qualities and criteria to watch out for when making selection of individuals to form software development team. With this, we recommend that task analysis be adopted as a tool for selecting individuals with the right technical skills, knowledge and ability to well perform their given task roles and thereby improving performance and thus contributing to success of software development projects.

REFERENCES

[1]. Acuña, S. T and Juristo, N. (2004). 'Assigning people to roles in software projects'. *Software-Practice and Experience*, vol. 34, pp. 675 – 696.

[2]. Acuña, S.T., Juristo, N. and Moreno, A.M. (2006). 'Emphasizing human capabilities in software development'. *IEEE Software*, vol. 23, pp. 94–101.

[3]. Annett, J. (2004). *Hierarchical Task Analysis*. In: Diaper, D. and Stanton, N.A. (eds) *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Erlbaum Associates, Mahwah, New Jersey, pp. 67-82.

[4]. Antoniol, G., Cimitile, A., Di Lucca, G. A., and Di Penta, M. (2004). 'Assessing staffing needs for a software maintenance project through queuing simulation'. *IEEE Transactions on Software Engineering*, vol. 30(1), pp. 43–58.

[5]. *Chaos Report*, The Standish Group. (2009). Retrieved November 2015 from: http://www.standishgroup.com/chaos.ht

[6]. Chiemeke, S. C. and Egbokhare, F.A. (2013). 'Towards Improving Software Developers Productivity and Effectiveness'**.** *Computing, Information Systems, Development Informatics & Allied Research Journal*. Vol. 4. No. 3.

[7]. Egbokhare, F.A. and Onibere, E.A. (2008). 'The most neglected P in software development projects'. *Asian Journal of Information Technology* 7 (9): 416-419 Medwell Journals

[8]. Gronau, N., Froming, J., Schmid, S. and Russbuldt, U. (2006). 'Approach for requirement oriented team building in industrial processes'. *Computer in Industry*, Vol. 58, pp. 179- 187.

[9]. Hoffman, R. R., Crandall, B., and Shadbolt, N. (1998). 'Use of the critical decision method to elicit expert knowledge: A case study in the methodology of cognitive task analysis'. *Human Factors, 40*, 254-277

[10]. Ho-Leung, R. (2010). 'Using Analytic Hierarchy Process (AHP) Method to Prioritise Human Resources in Substitution Problem'.

International Journal of the Computer and the Internet.

[11]. Kaplan, M., Dollar, B., Melian, V., Van Durme, Y. and Wong, J. (2016). *Human Capital trends 2016 survey*, Oakland, CA. Deloitte University Press.

[12]. Lacerenza, C., Marlow, S., Tannenbaum, S. and Salas, E. (2018). 'Team Development Interventions: Evidence-based Approaches for Improving Teamwork'. *American Psychologist*, 71(4), 517-531.

[13]. Li, R. Y., Chau, K. W. and Zeng, F. F. (2019). 'Ranking of Risks for Existing and new Building works'. *Sustainability* 2019, 11, 2863; doi:10.3390/su11102863.

[14]. Linberg, K. R. (1999). 'Software Developer Perceptions About Software Project Failure: A Study'. *The Journal of Systems and Software*, Vol. 49, Issue 2/3 pp. 177-192.

[15]. Lorin, J. (2000). 'Major Causes of Software Project Failures'. *The Journal of Defense Software Engineering* Crosstalk Associate Editor

[16]. Mohammed A. O., Sherif, A. M. and Ehab, E. (2013*). '*Survey: Problems Related to Human in Software Projects'. *IOSR Journal of Computer Engineering*. Volume 10, Issue 1.

[17]. Mohammed, A. E. and Abraham, O**.** (2013). **'**The Role of Enterprise Architecture in Aligning Business and Information Technology in Organizations: Nigerian Government Investment on Information Technology'. *International Journal of Engineering and Technology*, Volume 3, No 2.

[18]. Norita, A. and Laplante, P. A. (2006). 'Software Project Management Tools: Making a Practical Decision Using AHP'. *Software Engineering Workshop SEW-30* (SEW'06), IEEE.

[19]. Otero, L. D., Grisselle, C., Alex, J. R. and Otero, C. E., (2009). *A Multi-Attribute Decision Making Approach for Resource Allocation in Software Projects.* Computers & Industrial Engineering 56: 1333–1339.

[20]. Passova, S. (2005). *The need for Customer-oriented Software Development*. Sofea Inc. e-Biz Development.

[21]. Paulk, M. C., Weber, C. V., Curtis, B. and Chrissis, M. B. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley.

[22]. Strad, D. and Guid, N. (2010). 'A fuzzy-genetic decision support system for project team formation'. *Applied Soft Computing* 10. 1178–1187.

[23]. Trist, E. and Murray, H. (1990). *Historical Overview, in The Social Engagement of Social Science: A Tavistock Anthology*. Philadelphia: University of Pennsylvania Press, 1-34.

[24]. Tsai, H., Moskowitz, H. and Lee, H. (2003). 'Human resource selection for software development projects using Taguchi's parameter design'. *European Journal of Operational Research*, vol. 153, issue 1, pp. 167-180.

Table 4.1: The key for abbreviations

F = Frequency
-NP = Not Performed
-RP = Rarely Performed
-OP = Occasionally Performed
-FP = Frequently Performed
C = Criticality
-T = Trivial
-I = Important
-Cr = Crucial
WR = When Required
KC = Knowledge Criticality
SC = Skill Criticality
AC = Ability Criticality
JP = Relationship to Job Performance

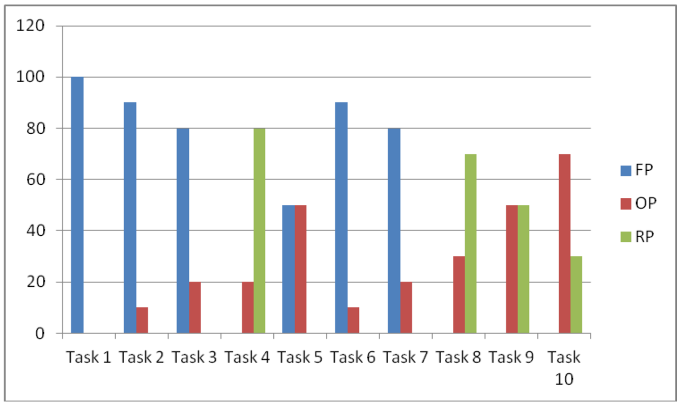


Figure 1a: Task FrequencyAnalysis

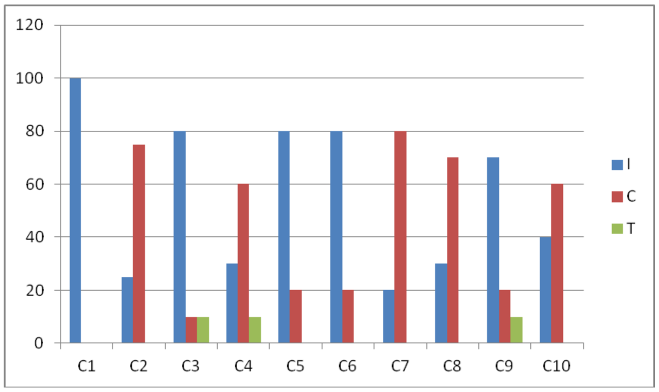
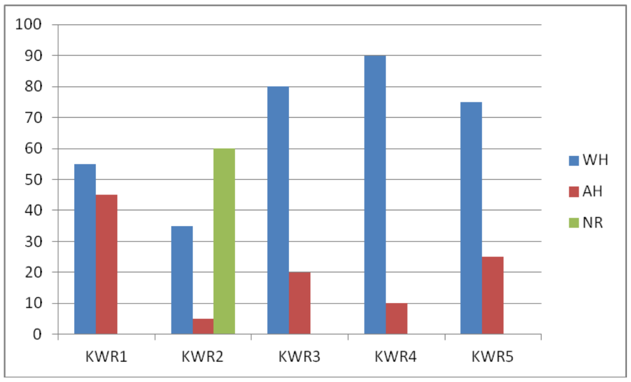


Figure 1b: Task Criticality Analysis



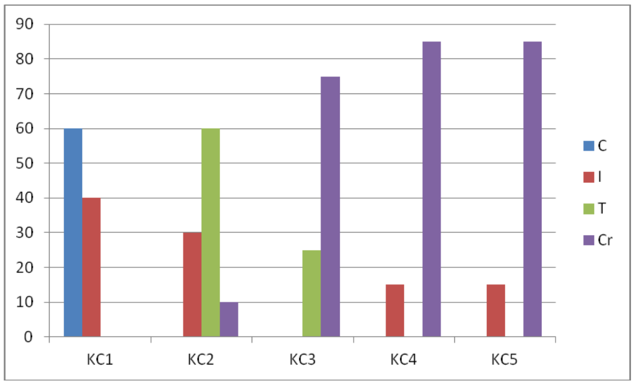
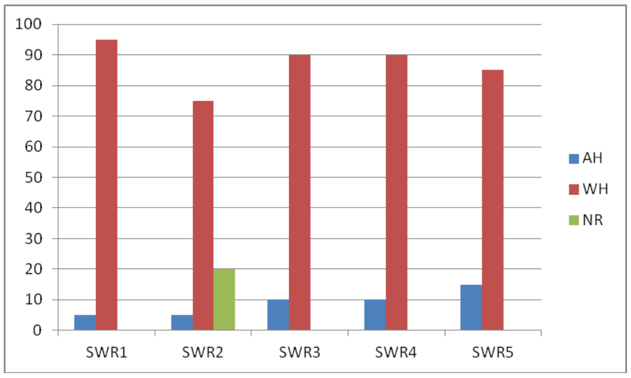


Figure 2a: Knowledge Analysis (when required)

Figure 2b: Knowledge Criticality



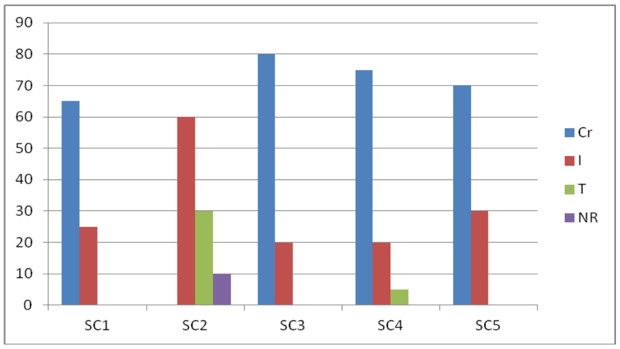
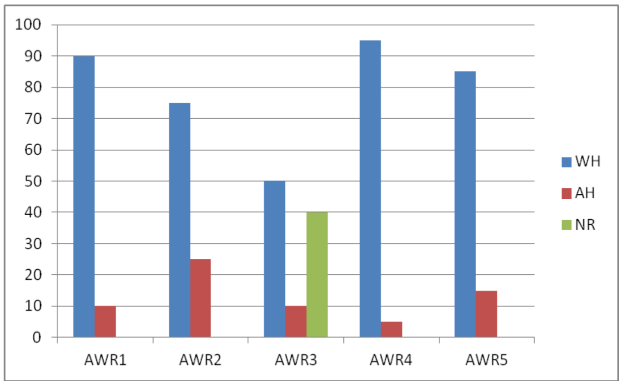


Figure 3a: Skills Analysis (when required)

Figure 3b: Skills Criticality



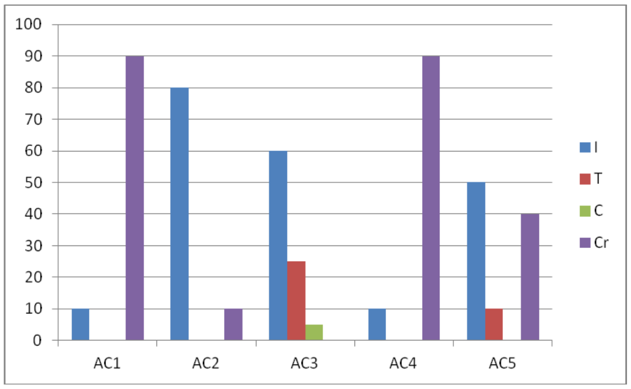


Figure 4a: Ability Analysis (when required)

Figure 4b: Ability Criticality