An Investigation on Application Domains for Software Effort Distribution Patterns

Thomas Tan, Barry Boehm, Brad Clark
Center for Systems and Software Engineering
University of Southern California
California, United States
{thomast, boehm}@usc.edu, brad@software-metrics.com

Abstract − **Many software cost estimation models suggest a one-size-fit-all effort distribution patterns as the basis of their guidelines for resource allocation on software development activities. This set of distribution percentages is generally developed from experiences with limited data support. In this paper, we will examine a set of real world data points and share our findings that uncover the significant variations of effort distribution patterns when categorized by application domains. Furthermore, we will discuss the possibility of extracting active factors from application domains that may help us enhance the overall effort distribution patterns as in COCOMO II model.**

Keywords: Cost Estimation, Effort Distribution, Application Domain, Effort Allocation

1. INTRODUCTION

Effort distribution is often a tough job for project managers as many estimation models and guidelines provide limited instructions on resource allocations. Boehm [1] introduced the Detail COCOMO model that explains the complicated steps to calculate detail costs based on the module-subsystem-system hierarchy and phase sensitive cost drivers, which are different by phases and activities. This estimation process is somewhat difficult to be executed most of the time; therefore, many practitioners would rather use a one-size-fit-all or a rule-of-thumb distribution percentages table to estimate the effort allocation across most common activities such as requirements, design, and coding, etc. This approach, many times, would finish the allocation task, but usually suffers from inaccurate distribution of resources and therefore creates a threat to the complete the project.

In the Detail COCOMO model, Boehm discussed the possibility of domain information as a software cost driver that may influence the effort. However, due to lack of data support and possible overlap with other cost drivers, it was eliminated from the model. In our recent research project, we've found that when we plot data from different application domains, we can get different effort distribution percentages; this interesting observation redirects our attention to investigate more about application domain on its role in affecting effort distribution patterns and hopefully ways to enhance the effort distribution percentages table similar to the one used in the COCOMO II model.

In the remainder of this paper, we will review research works that are closely related to effort distribution patterns in section 2; then, we will describe our investigation purpose and approach in section 3 follows by analysis results and discussion in section 4; at last, we will wrap up our discussion and conclusion in section 5.

2. RELATED WORK

Many practitioners use a conventional industry rule-of-thumb for distributing software development efforts across a generalized software development life cycle [2]: 15-20 percent for requirements, 15-20 percent for analysis and design, 25-30 percent for construction (coding and unit testing), 15-20 percent for system-level testing and integration, and 5-10 percent for transition. This approach is adapted by many mainstream software cost estimation models in forming effort distribution percentage tables by different activities or phases in software development process.

In the COCOMO 81 model [1], Boehm defined four software development phases, namely plan and requirements, product design, programming (consists of detailed design, coding, and unit testing), and integration and test phases. Using the Rayleigh curves approximation method, as discussed by Norden [3] and Putnam [4], Boehm estimated the effort allocation ratio for each of the four phases in the COCOMO 81 model. Furthermore, the effort allocation ratio is categorized by software size and COCOMO 81's three modes: organic, semidetached, and embedded, and a complete effort distribution percentages table [1, Chapter 23] is suggested for practitioners for spreading the total effort across the waterfall-like software process. In addition, Boehm argued that these effort distributions were independent of other cost drivers only in its Intermediate level of the model: in the Detailed level of the model, where cost driver may have different effort multiplier values, may use a different set that were based on extensive behavioral analysis [1, Chapters 24-27] and limited data [1, Chapter 29].

The COCOMO II model [5] inherits the approach from COCOMO 81 model with modification that unifies the three modes into one and replaces the original phase definitions with two activities schemes that are commonly used: Waterfall and MBASE/RUP, which was co-evolved with RUP[6]. Both schemes accommodate plan and requirements and transition phases if not already have and both schemes are provided with effort distribution percentages tables. There is no detailed effort distribution model as in the Detailed level of COCOMO 81 model due to lack of

calibrated data. The waterfall-like scheme is derived from the COCOMO 81 model's phase definitions thus similar to what we have for our research; and for that reason, we will use its effort distribution percentages to compare against our results in this analysis. The detail effort distribution percentages table for this waterfall-like scheme is shown in Table 1.

TABLE 1: COCOMO II WATERFALL EFFORT DISTRIBUTION PERCENTAGES [2]

Phase/Activities	Effort %
Plan and Requirement	7 (2-15)
Product Design	17
Detailed Design	27-23
Code and Unit Test	37-29
Integration and Test	19-31
Transition	12 (0-20)

Similar to the COCOMO models, many mainstream cost estimation models such as SLIM [7] and SEER-SEM [8] also define software development activities and effort distribution guidelines to assist management to allocating resources. RUP [6] also add detailed distribution pattern for its process using RUP's hump chart. To confirm the RUP distribution pattern, Heijstek and Chaudron [9] conducted a study in producing visualization of software development effort mapping to the RUP disciplines and finding that their results are very similar to that illustrated by the RUP hump chart.

In addition to mainstream models' proposed effort distribution guidelines, some of the recent studies also focus on investigating contributing factors that affect effort distribution patterns. For instance, Milicic and Wholin [10] investigated how experience level, work package size, and team size affects the outcome of estimation accuracy; and out of the three variables, experience level presents the most significant correlation to estimation accuracy deviation. Yang et al. [11] conducted a research study on 75 Chinese projects to investigate affecting factors of variant phase effort distribution. Their results show factors such as development type, software size, development life cycle, and team size may all play important role to change phase effort distribution pattern. Similarly, Kultur and et al. [12] discussed their approach of analyzing 395 ISBSG data points which they added application domain information in addition to development type and software size. Their results suggest that domain information can be used to improve estimation accuracy while confirming correlation between effort distribution patterns and development type and software size. However, unlike Yang's analysis, their study does not show detail definitions of the domains and disciplines (especially by phases) of the subject projects.

3. PURPOSE AND APPROACH

Our goal for this study is to find solid evidence to support our hypothesis that application domain is an important piece of information that can be exploited to find influential cost drivers on effort distribution patterns, and we can use these cost drivers to enhance existing one-size-fit-all software cost

estimation guideline on resource allocations as we have currently with COCOMO II model. To achieve this goal, we need to answer the following questions:

1. Does application domain matters? In another word, will effort distribution patterns look different in different application domains?
2. If application domain information matters, does it make enough differences in effort distributions averages compares against percentages table from the COCOMO II model?
3. If the answer to both of the above questions is yes, are there evidences to proof that we can extract cost drivers from application domains to improve software cost estimation guidelines on resource allocations?

We would love to answer all these questions "yes", so that we can build a strong foundation to continue our research in this direction, and ultimately, we can find influential drivers to improve the software cost estimation guideline on resource allocations. In the rest of this section, we outline our technical definitions, data processing steps, and data analysis procedures in order to provide enough detail about our investigation.

A. Definitions

Our first and foremost task is to have definitions for both software development activities and application domains that we use for our analysis.

Our data source, the source-sanitized subset of the Software Resource Data Report [13], provides a data dictionary that defines development activities, which are used as part of the SRDR form to collect effort in person hours. These development activities are very similar to those defined in ISO12207 and that used in COCOMO 81 and COCOMO II model. We map these definitions against the COCOMO II's waterfall-like phases as shown below in Table 2. Using the mapping between our data's activities definitions and the COCOMO II phases, we are able to connect the activities' effort percentages and then make comparisons in our analysis. Note that because our data does not cover any transition activities, we exclude the transition phase from the COCOMO II model for any comparison in our analysis.

TABLE 2: MAPPING OF SRDR ACTIVITIES TO COCOM II PHASES

COCOMO II Phase	SRDR Activities
Plan and Requirement	Software requirements analysis
Product Design and Detail Design	Software architecture and detailed design
Coding and Unit Testing	Coding, unit testing
Integration and testing	Software integration and system/software integration; Qualification/Acceptance testing

For the domain definitions, we use one of our previous studies [14] that defines 21 application domains based on industry and system functionality and 8 operating environments based on system platforms. For this set of application domain definitions, we compared several existing

taxonomies namely NAICS [15], IBM's work-group taxonomy and business functions [16], Digital's Industry and Application Taxonomy [17], Reifer's Application Types [18], and application types from US Air Force Estimation Handbook [19] and Putnam's study on software productivity [7, 20]. Base on the comparisons, we are able to select a list of application types from Reifer's Application Types, the US Air Force Cost Estimation Handbook, and some software domains discussed in Putnam's study on domain productivities. We merge these application types and supply each domain with examples to solidify the definitions. While we are selecting and merging application types, we also find that many application types are actually characteristics of system platforms and not exactly representing functional capabilities, thus, we decide to add 8 operating environments, summarize from those platform-based application types, to cover system platforms. Although we are not using the operating environments for this particular analysis on effort distribution, it is worth mentioning that the operating environments may be used as additional dimension to evaluate the results from analyzing application domains for effort distribution patterns.

Table 3 lists the eight domains and their definitions that we will use for this paper. There are 13 other domains that are not used because we can not find enough data points for these domains to have conclusive results. We may add these domains into our analysis if we find more data points either by backfilling missing data with feasible technique or getting more data points from the source.

TABLE 3: DOMAIN DEFINITIONS [14]

Domain	Definitions
Business	Software that automates business functions, stores and retrieves data, processes orders, manages/tracks the flow of materials, combines data from different sources, or uses logic and rules to process information.
Command and Control	Software that enables decision makers to manage dynamic situations and respond in real time. Software provides timely and accurate information for use in planning, directing, coordinating and controlling resources during operations. Software is highly interactive with a high degree of multi-tasking.
Communications	Software that controls the transmission and receipt of voice, data, digital and video information. The software operates in real-time or in pseudo real-time in noisy environments.
Mission Management	Software that enables and assists the operator in performing mission management activities including scheduling activities based on vehicle, operational and environmental priorities.
Mission Planning	Software used for scenario generation, feasibility analysis, route planning, and image/map manipulation. This software considers the many alternatives that go into making a plan and captures the many options that lead to mission success.
Simulation	Software used to evaluate scenarios and assess empirical relationships that exist between models of physical processes, complex systems or other phenomena. The software typically involves running models using a simulated clock in order to mimic real world events.
Sensor Control	Software used to control and manage sensor

and Processing	transmitting and receiving devices. This software enhances, transforms, filters, converts or compresses sensor data typically in real-time. This software uses a variety of algorithms to filter noise, process data concurrently in real-time and discriminate between targets.
Weapon Delivery and Control	Software used to select, target, and guide weapons. Software is typically complex because it involves sophisticated algorithms, fail-safe functions and must operate in real-time.

B. Data Processing

The second step is processing the data points so it is ready for our data analysis. The data we are using is from a collection of source-sanitized SRDR data [13]: each record in this data collection represents a software project; each project includes the following information: activity effort data, from requirements activities to qualification tests activities; sizing parameters such as new size, modified size, unmodified size, etc.; and project specific values such as application types, maturity level, staffing information, and requirement volatility, etc. In addition, we have also added in domains and environments categorization for our research works.

Due to the fact that we don't know the source and detailed background information about all projects, we assume normal distribution for all data patterns including effort distribution patterns. We query out data points with full information for this study, which means a record must have all of its activity effort data and we must know its application domain. Because the SRDR effort data separated integration and qualification testing effort into two fields, we have to add them together to match the COCOMO II model's integration and testing phase.

Once we have all the raw data prepared, we perform a series of calculations and checks before we make the set ready for the data analysis. First, we need to convert all the actual person-hours to effort distribution percentages. We then group the data points by application domains and checked its normality. Initially, we can look at the descriptive statistics, produced from Microsoft Excel, to see whether the means and medians are far apart or if the skewness and kurtosis look odd; if something is strange, we can also look at histograms and Q-Q [21] plot to visualize the data distribution. In order to have a final call on normal distribution, we conduct the Shapiro-Wilk test [22], Kolmogorov-Smirnov test [23], and Pearson's Chi-square test [24]. If three out of two are positive, we can use the mean value produced to represent the domain average for that particular activity. In situation where these normality tests reject normal distribution for the subset, we do one more round of data processing by eliminating possible outliers that may be twisting the data distribution for the data set. Using the inter-quartile range method [25] and the visualization plots, we can catch and eliminate some obvious outliers and then repeat the normality tests to produce the final data set. If we still get rejection on normality after the outlier elimination, we have no choice but drop the data set for now.

The outcome of the data processing procedure give us sets of data grouped by application domains with basic information such as average effort percentages for each activity and additional statistical parameters such as record counts, standard deviations, medians, and variances (all from descriptive statistics).

C. Data Analysis

After the data preparation step, we get 8 domains with complete effort data: Business, Command & Control, Communications, Mission Management, Mission Planning, Sensor Control and Processing, Simulation, and Weapons Delivery and Control. Each domain has a set of effort data divided into four activity groups: Plan and Requirements, Architecture and Design, Coding and Unit Testing, and Integration and Qualification Testing.

We run a single factor ANOVA analysis on all domains to answer our first question: whether domain information matters? The null hypothesis for this test is that all domains would have the same mean percentages, whereas the alternative hypothesis is that domains may not have the same mean percentages. We are hoping that the ANOVA results indicate rejection of the null hypothesis, which means we can declare that effort distributions are different between at least two domains, thus answer the first question with a "yes" and then able to move on to the second question.

For our second question, we use the independent one-sample t-test [26] that would test if the domain percentages agree with the COCOMO II model's effort distribution percentages. The formula for the t-test is shown as follows:

$$t = \frac{\bar{x} - \mu_0}{\frac{s}{\sqrt{n}}} \quad (1)$$

Statistic t in the above formula is to test the null hypothesis that sample average $\bar{x}$ is equal to a specific value μ_0, where s is the standard deviation and n is the sample size. In our case, μ_0 would be the COCOMO II model's effort distribution averages and $\bar{x}$ is a domain's distribution percentage average for each activity group. In this part of the data analysis, we are hoping to get as many rejections as possible against the null hypothesis, which means the domain average does not agree with the COCOMO II model's effort distribution averages. The more disagreements we get, the more confident we are to say that there are enough differences between the COCOMO II model, which represents one-size-fit-all effort distribution, and application domains' distribution. Such ideal result indicates that the current COCOMO II model's effort distribution percentages are not sufficient for accurate estimation on effort allocation and thus necessary to find improvement.

For both ANOVA and t-tests, we use 90% significance level to accept or reject the null hypothesis because our data is from real world projects and the noise level is rather high.

If we are able to answer the first two questions with "yes", then we can continue to look for evidence to answer the third question. In answering this question, if we are able to extract specific factors from application domain information and these factors have clear relationship with effort distribution pattern, then we may find proof that using these factors, we can construct better effort distribution percentage table that is more accurate than the current one-size-fit-all effort distribution patterns.

4. ANALYSIS RESULTS AND DISCUSSION

From the data analysis, as described in the last section, we collect results that we can use to answer our questions. In the following paragraphs, we will review these results and discuss their implications.

In Table 4, we list out the average effort percentages for each domain by activities groups. The average effort percentages are closer for the requirement activities between domains: a little bit more than 10% difference between the high (Business domain) and the low (Sensors Control and Processing domain). This difference is a lot smaller than those from other activity groups: almost 25% difference find in architecture & design and code & unit testing activities; 15% for integration and qualification testing activities. We can also see this in Figure 1, which shows a plot of effort percentages across all activities and for all eight domains. Our first explanation of this trend is that people perform the same set of tasks initially to collect requirements regardless of the system domain. However, we can see that some domains need more time than the others in collecting requirements, so maybe there is something that we can dig into but we stick with the results for now. Another interesting observation is that between Mission Planning and Sensor Control and Processing domains, their plan & requirement and integration & qualification testing activities have similar average effort percentages, but huge differences in the other two activity groups; we believe that this difference may be influenced by some particular characteristic that is sensitive to the architecture & design and coding & unit testing activities or vice versa. Finding this characteristic may be good to answer our third question.

TABLE 4: EFFORT DISTRIBUTION AVERAGE BY ACTIVITY GROUPS FOR DOMAINS

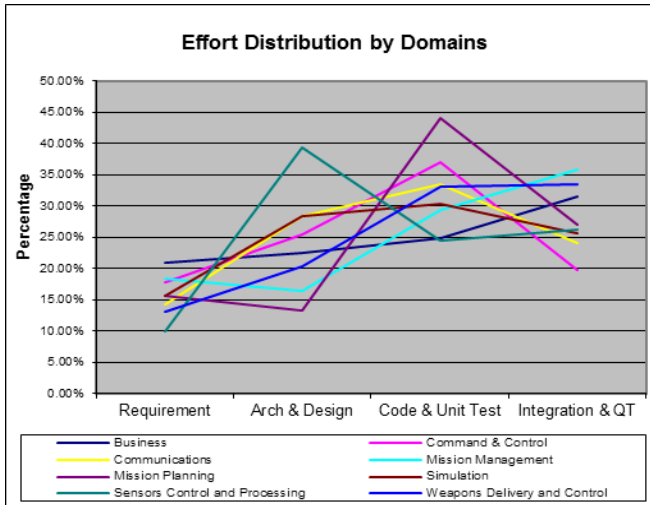| Domain	Requirement	Arch & Design	Code & Unit Test	Integration & QT
Business	20.98%	22.55%	24.96%	31.51%
Command & Control	17.82%	25.47%	36.98%	19.74%
Communications	14.21%	28.32%	33.42%	24.05%
Mission Management	18.33%	16.51%	29.39%	35.77%
Mission Planning	15.66%	13.30%	44.08%	26.95%
Simulation	15.69%	28.31%	30.43%	25.56%
Sensors Control and Processing	9.98%	39.42%	24.43%	26.17%
Weapons Delivery and Control	13.06%	20.37%	33.05%	33.52%

FIGURE 1: Effort Distributions by Domains

Although Figure 1 gives us a good glance of the effort distributions and shows some obvious variations, we can't declare that these variations are actually meaningful unless we have statistical proof. In order to determine the significance, we need to examine the ANOVA and t-tests results. In Table 5, we can see that the ANOVA results support to reject the hypothesis that all domains produce the same effort distribution for all activity groups except plan & requirement activities, which would reject the hypothesis at about 50% confidence level and far from our standard at 90%. This confirms what we've seen in Figure 1 where most domains have similar average effort percentages for requirement activities. In contrast, architecture & design and integration & qualification testing show strong rejection of the hypothesis, that p-values are much less than 0.1 for 90% confidence level. However, we are a little surprise that we can barely reject null hypothesis for code & unit testing activities which show a much larger variation in the graph than indicated by the ANOVA results; perhaps the average effort distribution percentage for coding & unit testing is much closer between domains than we believed. Overall, different domains produce different effort distribution percentages in three out of four activity groups, which mean that the majority of activities will have different effort percentages between domains, and that should be enough to confirm our guess to answer the first question, which the answer is "yes".

TABLE 5: ANOVA RESULTS

Activity Groups	F	P-value	Results
Plan & Requirements	0.9165	0.4995	Can't reject
Architecture & Design	3.7064	0.0019	Reject
Code & Unit Testing	1.8000	0.1020	Barely reject
Integration & Qualification Testing	2.1125	0.0542	Reject

With "yes" to the first question, we can answer the second question by using the t-test described in section 3.3 to test individual domain vs. COCOMO II model's average effort percentage for each activity group. The following four tables show the results of the t-tests. In the tables, if the t-

stat is greater than the t-critical value, we can reject the null hypothesis, otherwise, the result is inconclusive as we can't neither reject nor accept the null hypothesis.

TABLE 6: DOMAIN VS. COCOMO II – PLAN & REQUIREMENT ACTIVITIES

Domains	Average	t-stat	t-critical	Results
Business	0.2098	3.0535	2.132	Reject
Command & Control	0.1782	4.2287	1.782	Reject
Communications	0.1421	5.4045	1.729	Reject
Mission Management	0.1833	3.4513	1.796	Reject
Mission Planning	0.1566	2.3763	1.943	Reject
Sensor Control & Processing	0.0998	0.6120	2.132	Can't reject
Simulation	0.1569	2.1004	1.943	Reject
Weapon Delivery and Control	0.1306	3.2134	2.132	Reject

Note: COCOMO II Average: 7%

TABLE 7: DOMAIN VS. COCOMO II – ARCHITECTURE & DESIGN ACTIVITIES

Domains	Average	t-stat	t-critical	Results
Business	0.2255	10.2678	2.132	Reject
Command & Control	0.2547	3.9674	1.782	Reject
Communications	0.2832	7.5065	1.729	Reject
Mission Management	0.1651	6.8012	1.796	Reject
Mission Planning	0.1330	11.5336	1.943	Reject
Sensor Control & Processing	0.3942	0.3744	2.132	Can't reject
Simulation	0.2831	2.6657	1.943	Reject
Weapon Delivery and Control	0.2037	10.1673	2.132	Reject

Note: COCOMO II Average: 42%

TABLE 8: DOMAIN VS. COCOMO II – CODING & UNIT TESTING ACTIVITIES

Domains	Average	t-stat	t-critical	Results
Business	0.2496	2.4822	2.132	Reject
Command & Control	0.3698	1.2184	1.782	Can't reject
Communications	0.3342	0.1670	1.729	Can't reject
Mission Management	0.2939	1.2077	1.796	Can't reject
Mission Planning	0.4408	1.2854	1.943	Can't reject
Sensor Control & Processing	0.2443	1.6346	2.132	Can't reject
Simulation	0.3043	0.5257	1.943	Can't reject
Weapon Delivery and Control	0.3305	0.0188	2.132	Can't reject

Note: COCOMO II Average: 33%

TABLE 9: DOMAIN VS. COCOMO II – INTEGRATION & QUALIFICATION TESTING ACTIVITIES

Domains	Average	t-stat	t-critical	Results
Business	0.3151	1.6347	2.132	Can't reject
Command & Control	0.1974	0.7998	1.782	Can't reject
Communications	0.2405	0.3897	1.729	Can't reject
Mission Management	0.3577	2.9630	1.796	Reject
Mission Planning	0.2695	0.2940	1.943	Can't reject
Sensor Control &	0.2617	0.4070	2.132	Can't reject

Processing				
Simulation	0.2556	0.1067	1.943	Can't reject
Weapon Delivery and Control	0.3352	3.2346	2.132	Reject

Note: COCOMO II Average: 25%

As we can see from Table 6 and 7, the majority of the domains rejects the null hypothesis and indicates that our calculated averages are significantly different from the COCOMO II model's averages for requirements and architecture & design activities. On the other hand, for coding & unit testing and integration & qualification testing, we can't reject the null hypothesis and have to say that we are not 90% confident to claim the domains' averages are different from the COCOMO II model's averages. One interesting remark is that t-test results for Sensors Control & Processing domain seem to not reject any of the COCOMO II effort averages as most others one would reject for requirement and architecture & design activity groups. For this reason, we may need to look deeper into this domain and perhaps it will help us get more information on why other domains produce inconclusive results. In summary, the inconclusive results from the tables, particularly table 8 and 9, may suggest lack of data points and call for further investigation or deeper analysis. However, as we mentioned early, we want to get as many rejections as possible from the t-tests, and if we get more than half, we can make a bold call that we have enough evidence to answer "yes" to our second questions, thus give ourselves a chance to move forward. We understand that this call may be overturned later when we have more data points, but we are optimistic that we are on the right track for now.

With answer to both questions is "yes", we need to look for evidences that enable us to extract cost drivers from application domain information. Recalling the COCOMO 81 model [1, Chapter 28], we understand that application domains may carry information such as complexity, execution time, and database; therefore, if we find a specific factor that all application domains have but with different value, we can test whether this factor can help enhancing the estimation guideline for resource allocation.

One of the suggested factors is human-intensiveness that most likely has some influence over requirement and architecture & design activities. We get this idea from observing the requirement effort percentages: there are two groups of domains; one group tends to use more time, and the other tends to use less. We figure that this trend has something to do with how human is involved from the client side in requirements and architecture & design activities: for instance, for a system in the Business domain, the specifications usually come from human users, but for a system in Sensor Control & Processing, the specifications are most likely coming from the hardware requirements of the sensor. Categorizing by this factor, the 8 domains we have can be divided into two subsets – human-intensive domains and hardware-driven domains. In the human-intensive group, call it Group A, we have Business, Command & Control, Mission Management, and Mission Planning; in the hardware-driven group, Group B, we have

Communications, Sensor Processing & Control, and Weapons Delivery and Control; we can't find a clear cut for Simulation domain, thus we dropped it from this small experiment. Combining the data points in each group, we will get two sets of data points that can be used in a t-test to check the significance of their difference. Similar to the ANOVA tests we ran earlier for answering question 1, we get results that show significant difference between the two groups for requirement and architecture & design activities, but inconclusive for coding & unit testing and integration & qualification testing. This result is as expected since we group the domains by a factor that we believe will influence system specification and design. In addition, we see that drivers may be active for some activities yet inactive for others; similar to what Boehm suggested in the Detailed level of the COCOMO 81 model: there are phase-sensitive cost drivers that its driver value differs from activities to activities. This is important that if we can get these values from application domain information, we can definitely provide a better estimation guideline, especially at the early stage of the software development lifecycle, when limited information is available.

In addition to the t-tests checking the significance between groups, we also run the t-test on both groups to test their averages against the COCOMO II model's effort percentages, and the results are similar to what we find using all eight domains against the COCOMO II model's effort percentages. This is not surprising because we believe this factor is only active for requirement and architecture & design activities, thus changes, if any, will only be seen in these two activity groups and not the other two.

From this simple experiment, we learn that we can repeat the combining and tests for drivers that we can extract from application domain information and check whether the drivers apply for any specific activity group or all of them. This helps answering our third question, which asks for evidence of extracting drivers to improve estimation guideline on resource allocation; moreover, encourages us to continue our research in the direction to find these factors.

5. CONCLUSION

In summary, we see differences between domains and against the COCOMO II model's effort percentages and we use various statistical tests to prove the significance of these differences. Although we get some inconclusive results that may affect the overall creditability of the research outcome, this study provides the step stones that we need in order to complete the research in the right direction. In addition to continuously getting more data points from our data source, we are also improving our data analysis by designing an effective scheme to backfilling the missing data to increase our sample size; we believe we will have enough evidence to solidify the research findings.

As for now, based on the current analysis results, we are convinced that application domain, which is usually available early in a software development lifecycle, is an important piece of information that we can exploit in finding enhancement to the existing effort distribution guideline in most mainstream cost estimation models, particularly the

COCOMO II model. With this groundbreaking investigation, we are positive to continue our research with this direction toward our goal on improving cost estimation guidelines on resource allocation.

REFERENCES

[1] Boehm, B. Software Engineering Economics. Prentice Hall, New Jersey. 1981.

[2] Borysowich, C. "Observations from a Tech Architect: Enterprise Implementation Issues & Solutions – Effort Distribution Across the Software Lifecycle". Enterprise Architecture and EAI Blog. http://it.toolbox.com/blogs/enterprise-solutions/effort-distribution-across-the-software-lifecycle-6304. October 2005.

[3] Norden, P.V. "Curve Fitting for a Model of Applied Research and Development Scheduling". IBM J. Research and Development. 1958. Vol. 3, No. 2, Page 232-248.

[4] Putnam, L.H. "A Macro-Estimating Methodology for Software Development". IEEE COMPCON 76 Proceedings. September 1976. Page 138-143.

[5] Boehm, B., et al. Software Cost Estimation with COCOMO II. Prentice Hall, NY. 2000.

[6] Kruchten, P. The Rational Unified Process: An Introduction. Addison-Wesley Longman Publishing Co., Inc. Boston. 2003.

[7] Putnam, L. and Myers. W. Measures for Excellence. Yourdon Press Computing Series. 1992.

[8] SEER-SEM. http://www.galorath.com.

[9] Heijstek, W., Chaudron, M.R.V. "Evaluating RUP Software Development Process Through Visualization of Effort Distribution". EUROMICRO Conference Software Engineering and Advanced Application Proceedings. 2008. Page 266.

[10] Milicic, D., Wholin, C. "Distribution patterns of Effort Estimation". EUROMICRO Conference Proceedings. September 2004. Page 422.

[11] Yang, Y., et al. "Phase Distribution of Software Development Effort". Empirical Software Engineering and Measurement. October 2008. Page 61.

[12] Kultur, Y., Kocaguneli, E., Bener, A.B. "Domain Specific Phase By Phase Effort Estimation in Software Projects". International Symposium on Computer and Information Sciences. September 2009. Page 498.

[13] Defense Cost and Resource Center. "The DoD Software Resource Data Report – An Update." Practical Software Measurement (PSM) Users' Group Conference Proceedings. July 2005.

[14] Tan, T. Clark, B. "Technical Report of a New Taxonomy for Software Application Domains and Operating Environments." USC CSSE Technical Reports. 2011.

[15] North American Industry Classification System, http://www.census.gov/eos/www/naics/, 2007.

[16] IBM Corporation. Industry Applications and Abstracts. IBM. White Plains, N.Y., 1988.

[17] Digital Equipment. VAX PWS Software Source Book. Digital Equipment Corp., Maynard, Mass., 1991.

[18] Reifer Consultants. Software Productivity and Quality Survey Report. El Segundo, Calif., 1990.

[19] US Air Force. *Software Develoopment Cost Estimating Handbook*, Software Technology Support Center, Vol 1, Sep 2008

[20] McConnell, S. *Software Estimation Demystifying the Black Art*, Microsoft Press, 2006, page 62.

[21] Blom, G. Statistical estimates and transformed beta variables. John Wiley and Sons. New York. 1958.

[22] Shapiro, S. S.; Wilk, M. B. "An analysis of variance test for normality (complete samples)." *Biometrika* 52 (3-4), 1965: page 591–611.

[23] Stephens, M. A. "EDF Statistics for Goodness of Fit and Some Comparisons". *Journal of the American Statistical Association*. Vol. 69, No. 347 (Sep., 1974). Page 730-737.

[24] Pearson, K. "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". *Philosophical Magazine*, Series 5 50 (302), 1901. Page 157–175.

[25] Upton, G., Cook, I. Understanding Statistics. Oxford University Press. Page 55. 1996.

[26] O'Connor, J. Robertson, E. "Student's t-test", *MacTutor History of Mathematics archive*, University of St Andrews, http://www-history.mcs.st-andrews.ac.uk/Biographies/Gosset.html