Journal of International Technology and Information Management

Volume 18
Issue 3 *Double Issue 3/4*

Article 14

2009

Cultural Issues in Software Estimation: From Intuition to Model Based Estimation in Upgrade Projects

Sanjay Mohapatra
Xavier Institute of Management

Cultural Issues in Software Estimation: From Intuition to Model Based Estimation in Upgrade Projects

Sanjay Mohapatra
Xavier Institute of Management
INDIA

ABSTRACT

This paper discusses issues related to change in culture that happens when software estimation becomes scientific, methodical and predictive. From a person based intuitive approach to model based estimation techniques, the approach necessitates a change in mindset and culture. This change in culture has been noticed in upgrade projects. Upgrades play an important role in product life cycle. A product is released with required features to meet immediate requirements of the customers. Any additional or modifications to features are carried through upgrade projects. These upgrades planned well in advance by the product developing organizations are made available to customers through published road map for the products. And hence it is important to be able to predict the effort required for these upgrade projects accurately and consistently. The objective of this research was to study cultural issues while developing an estimation model that would increase accuracy, predictability of estimation in upgrade projects. The methodology adopted in this research work was to use primary sources of data to develop an estimation model for upgrade projects and test this model in live upgrade projects. The results from the research showed that the level of accuracy for estimation increased, predictability in delivery was higher without comprising quality of final deliverables. The culture also went through a change through training and mentoring.

WHAT IS AN UPGRADE PROJECT

Common types of software upgrades include changing the version of an operating system, office suite, anti-virus program, or various other tools. Most of the cases software upgrades are often downloaded from internet in the form of a patch. A patch does not contain the software in entirety, but changes that are required to be made. These patches usually address small additional functionalities and also address concerns related to software security. A software upgrade can be minor or major depending on the amount changes incorporated in the released software. When a major upgrade happens, there would be a change in version number. The common nomenclature adopted for minor release usually follows with a ".01", ".02", ".03", etc. For example, version 10.03 means that that is the third minor upgrade of version 10. The vendor organization generally does not charge for minor upgrades, but insists that major upgrades be purchased.

An upgrade happens when an existing product is replaced with a new version of the product or already installed application (Box, 1983; Day, 1981; Levitt, 1965; Dhalla & Yuspeh, 1976). Upgrade projects start after newly developed application is installed at the customer site. During this installation the application would perform required functionalities required to meet business needs. However, these functionalities always need to be modified and new functionalities need to be added to meet changing business needs (Liu, Adkins, Yao, & Williams, 2007; Conde, 2002; Dver, 2003). Sometimes upgrade projects take care of non functional requirements such as

performance improvement of the installed application. This is done so that more number of users can access the installed application and can use them simultaneously. Such types of additional requirements are handled by upgrade projects by integrating new functionalities (Raymond, 2009). Through upgrade projects not only software, but also hardware is also modified or replaced so that system can be up-to date and take advantage of new technology. For example installing additional memory (RAM) or graphics card or additional hard disk is also carried through upgrade projects.

BACKGROUND OF STUDY

J & B Software Incorporation (www.jbsoftware.com) is a leader in providing solution in remittance and payment processing industry and been in existence for last 25 years. It had revolutionized the image based payment processing industry by providing simple-to-install and easy-to-maintain core applications that are capable of meeting specific customer needs. These solutions are developed through domain expertise, using the best-of-breed software development model which allows the organization to meet change in requirements with optimized cost. While all the functionality requirements are met, this model helps to meet delivery schedule. The customer profile includes organizations from regional and money-centre banks, insurance companies and mutual funds, credit card and student loan processors, telecom, utilities, government, non-profit and commercial organizations.

Upgrade plays a major role in product development life cycle in J&B Software. After product is installed at the customer site as part of its solution delivery process (www.jbsoftware.com), any change in functionalities or change in performance factors are carried through upgrade projects. In this upgrade projects, addition, modification of delivered functionalities, enhancement of existing performance and migrating from existing hardware to new hardware are executed. These upgrades to existing installed products are carried through upgrade project lifecycle (Box, 1983; Day, 1981; Levitt, 1965; Conde, 2002; Dver, 2003). It is expected that these requirements are correctly understood so that estimation can be done accurately. It's also important the estimation process is consistent so that customers can manage their budget apportioned for upgrading installed products.

The researcher was requested to study and propose an estimation model for these upgrade projects.

OBJECTIVES OF THE STUDY

The objectives of the study were:

1. To study existing estimation models and find their suitability for upgrade projects.
2. To develop an estimation model for upgrade projects.
3. To test this estimation model in a live projects in a software organization and find its usefulness.
4. To formulate an approach so that transition to systematic model based estimation culture can be smooth.

METHODOLOGY

The methodology adopted for the study were:

1. For objective one
    a. To use secondary sources to study available literature on known estimation models for software projects.
    b. To find their usefulness for upgrade projects.

2. For objective two
    a. Study estimation process in a software development projects. The selection of the software organization was dependent on number of upgrade projects carried out by the said organization. The organization was also chosen based on their geographical presence. This ensured that the estimation model, so developed, can be generalized and can be used by other software development organizations building upgrade projects.

    Sample selection and size

    a. The period of data collection was from Jan 2005 to Feb 2007, which ensured that it was being practiced recently.
    b. The sample size was for 24 different features, which was a fair representation of many upgrade projects that are being developed.
    c. The organization selected was in Banking payment domain, which is expected to grow many folds in next five years (NASSCOM, 2004).

3. For objective three
    The model was applied in live projects to validate the benefits.

    Sample selection and size

    a. Five live upgrade projects were selected that had development life cycle of twenty five man days to one man month (thirty man days).
    b. The upgrade projects were started in May 2007 and completed in Dec 2007, which used many of the features studied in objective two.

4. For objective four
    Study of different literature to understand IT culture in different organizations and then propose a framework for implementation for upgrade projects in J&B Software.

LITERATURE REVIEW FOR FINDING SUITABILITY OF USING DIFFERENT ESTIMATION METHODOLOGIES

Several methods are available for estimating software projects. These estimations methodologies have been examined to find suitability of using them for upgrade projects. This section of

literature review examines different methods that are followed for estimation and their suitability for usage in upgrade projects.

Suitability of using functional size measurement

Functional Size Estimation uses number of functionalities to be made available in the application and is a top down technique used for estimating summary level activity first and then breaking it into lower level activities. After collecting requirements from the customer, the project team would count number of function points by using different methodologies available. When using function points (fp) (Albrecht, 1979; IFPUG 1999,) or cosmic functional size units (cfsu) (Abran, Symons & Oligny, 2001), the estimator needs to understand the required functionalities and then use either of the above mentioned methodologies to count function points. But all these methodologies would give accurate results for large size new application development. To make a reliable estimate for (new) development project the size should be over 200 fp or 100 cfsu. Most of the upgrade projects are smaller sized applications and hence can not use these methodologies. Jones (1986) published a method based closely on that of Albrecht, called 'Feature Points'. This method aims to extend functional size methodology to scientific algorithms. However, this method cannot estimate size for applications where functionalities are being upgraded or modified and hence is not in use for upgrade projects. Symons (1988) modified Albrecht's (1979) Function Point and developed the 'MkII Function Point Method' which aimed to take care of complexity of business application software which are 'data-rich'. However it was difficult to measure and classify complexity related to data requirements in business application software and hence this method could not be used for upgrade projects in commercial software development.

Using Albrecht's (1979) approach, Whitmire (1992) developed '3D Function Points' for estimating size of scientific and real-time software. The three dimensions in 3D function points are data, function and control. The data dimension is similar to Albrecht's function points. The function dimension adds transformations, which are similar to the algorithms and the control dimension adds transitions, which explains changes in application state. This approach was a proprietary of Boeing and not widely used because, it is not easy to use for counting function points and compared to feature points, does not help in counting function points for algorithms in scientific software. NESMA (1997) developed its own variant for counting function points. The variations from IFPUG method of counting function point were related to "further data processing" and data display which is also known as "implicit enquiry". This approach also does not take care of complexities involved in the algorithm and mostly used for development projects. For smaller size of the application and upgrade projects, this method is not suitable. The University of Québec, Montréal and others published the 'Full Function Point Method' in 1997 which used the IFPUG rules for business application software and added extra components for sizing real-time software; however this has not been accepted by practitioners for upgrade projects because this method can be accurate for large sized application development only.

Suitability of using use cases

Jacobson (1986) came up with a top down approach for sizing software applications using "use cases". However the approach had following demerits: 1. It can not be used for measuring non-

functional requirements (such as platform, performance, timing or safety-critical aspects. Also complex algorithms and mathematical requirements can not be captured through use cases. 2. Use cases depend on individual skill to define templates which brings inconsistency to estimation. 3. It becomes difficult to visualize the level of complexity involved in User Interface. In upgrade projects, any requirement for improvement on the performance aspects (non-functional requirement) or improvement on the look and feel (User Interface) can not be measured using use cases. Hence this methodology can not be used in all upgrade projects. Jaime Campos, Jantunen and Prakash (2007) had come up with an estimation methodology for web and mobile technologies using used case approach.

Suitability of using consensus based estimation

The Suitability of using consensus based estimation (http://www.sei.cmu.edu/intro/process/technqs/q_estm.htm#14) technique uses the method of getting a small group of people to decide estimation required for an activity. In this method the group is unanimous in deciding on the value for an activity. However sometimes it becomes difficult to get the entire group to arrive at consensus and this technique is best used in a workshop and is not suitable for commercial upgrade projects. It will not be a viable option all the time because of lack of consensus and lack of continuity of same group members for all upgrade projects.

Suitability of using object based estimation technique

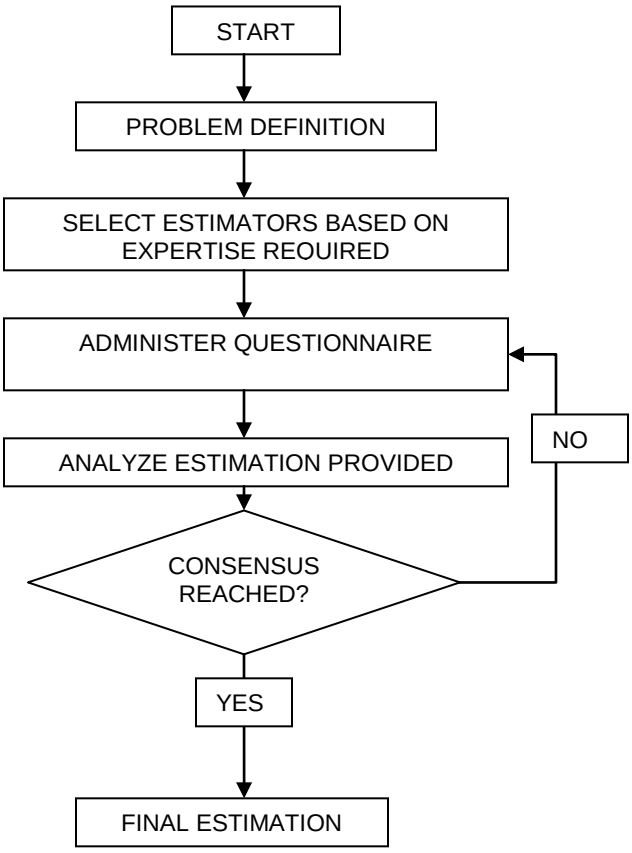
This is a bottom up estimation technique (Armstrong, 2006; Jacobson, 1995) where an application is broken into different objects. Estimation for each object is then found out and then effort is summarized at the aggregate level. The estimation is dependent on identification of number of objects and effort required for each object is calculated based on experience. The technique gives good results if all the objects are well known at the start of the project. For a new development project, an initial estimate based on object oriented approach should be refined as the project progresses as with progress of project, clarity is obtained with respect to objects; thus for an upgrade project this technique can not be used because upgrade projects are usually for relatively small duration (starting from few hours to 4 - 6 weeks maximum) and also initial estimate for the available application must have been done using object oriented approach.

Other available estimating techniques are Weighted Average (WAVE) and Quickest (http://www.sei.cmu.edu/intro/process/technqs) and these two techniques are not popular as they do not provide accuracy, consistency and predictability for estimation in upgrade projects. COCOMO (Boehm, 1981) is a popular technique used for software estimation. This technique further refined as COCOMO II considers a set of four “cost driver attributes” such as 1. Product attributes 2. Hardware attributes, 3. Personnel attributes, 4. Project attributes. Each of these four factors is given a rating on a 6 point scale and these ratings are added up using a multiplier to arrive at final “effort adjustment factor”. For upgrade projects, which are for small durations, usually delivery schedule expected range from 1 day to 3 weeks and it will be an overkill to consider all “cost driver attributes” and arrive at effort adjustment factor using COCOMO II for upgrade projects.

Suitability of using estimation using Delphi Method

In 1969, the RAND Corporation developed the Delphi Method which is essentially a group decision process about the likelihood that certain events will occur. It can also used for environmental, marketing and sales forecasting. It is a structured process for collating knowledge from a group of experts by administering questionnaires (Madu, Kei, & Madu, 1991). Usually all participants maintain anonymity which does not allow them to dominate using their authority or personality minimizes the "bandwagon effect" or "halo effect", and allows them to freely express their opinions and encourages open critique and admitting errors by revising earlier estimation. Problems associated with Delphi are: 1. its inability to make complex forecasts with multiple factors, 2. future outcomes were usually considered as if they had no effect on each other, and 3. future developments are not always predicted correctly by iterative consensus of experts. A flowchart is given in Figure 1 to show steps involved in Delphi method.

Figure 1: Steps in the Delphi Method.



In practice this means that the use of the extended methods is (only) applicable in organizations that work with releases. If every request for chance is put through in the system immediately, it is better to use “expert” estimates. There is also another generic limitation. The requirements or request for changes should be defined clearly and completely otherwise functional size measurement will give a result accompanied with a lot of hypotheses.

SUMMARY OF SUITABILITY OF EXISTING MODELS

As seen in different literatures, there are number of estimation approaches available for upgrade projects. These approaches can be categorized as intuition based and model based estimation. In intuition based estimation approach, the estimate is produced based on judgmental processes. In model based estimation approach, formula derived from historical data is used for estimation. The degree of accuracy is different in different approaches. Model based estimation provides consistency and if the formula is derived from historical data, then the accuracy also increases. The models that have been discussed above are useful if the applications are large and are executed over a schedule of more than three months. This is so because processes that are used for executing the projects become stable and as the projects progress the stability in requirement management brings in better analysis and design. However, in upgrade projects, the team does not have that luxury and requirements need to be understood, designed and code to be written in quick succession. This requires the estimation approach to be fast using empirical based model, to be as accurate as possible. A delay of two to three days in schedule will make an impact of total schedule delivery in terms of percentage terms. For example a delay of four days in a six months duration project will delay the project by (four divided by one hundred eighty days) or less than three percent; where as a delay of three days in an upgrade project of two months duration will impact by (two divided by sixty days) or by five percent. Hence the available models need to be improved to provide better accuracy in estimation.

In summary, the available literature shows that there are number of estimation models available for software engineering. However, these estimation models can not be applied to upgrade projects directly. This leaves gaps for practitioners and researchers as well to estimate and predict the effort required for upgrade projects. There are also issues which relate to adopting the new estimation methodology by the team. The culture in IT organizations (Nord, Nord, Cormack, & Cater-Steel, 2007) need to go through a change management process for sustainable benefit from a better estimation approach. These gaps have been addressed in this paper.

APPROACH FOR ESTIMATION IN J & B SOFTWARE FOR UPGRADE PROJECTS

In an upgrade project, the set of activities can be easily predicted. This set of activities is arrived after a period of time when the product development team can predict activities involved while upgrading the present installed products. These set of activities are made transparent to the customer; a model has been developed which will calculate the total effort required to complete these set of activities. However, in case a new activity is required to complete the upgrade project which has not been envisioned earlier, then the estimation is done based on domain expertise available in the company. Actual results after completing this new activity is then added to the model which is further refined as similar upgrade requests arrive. Also as the knowledge and experience level in the team increases, the productivity of the team also increases. The capability to turn around any upgrade project also increases; the values of different parameters in the model are then fine tuned to take care of increased productivity. The benefits of the increased productivity are passed onto customers because of which, the customers can meet business requirements faster and use their budget for upgrading projects effectively. In turn, customers order for more upgrade projects, translating to increased revenue earnings for vendor organization. Thus a win-win situation is ensured for both vendor and customer fostering

a continued, well established, stable business relationship between the two. Table I and table II below explain the approach taken for model based estimating in J&B Software Inc.

Table I describes guidelines that are required to categorize an activity. An activity is categorized on a scale of 1 to 5 based on the level of their complexity. Category 1 is the least complex situation where as category 5 mean the most complex situation. For example, in sl. No.12, the required feature (PRF reports (porting)) is simple when upgrade requires only simple data porting, where as if PRF reports (porting) requires sorting and grouping, then complexity level is 3. Point to be noted here is that at present there is no category 4 and category 5 for this required feature; this so because no scenario can be envisioned at the moment whose complexity can be categorized as 4 and 5 on a complexity scale of 1 to 5. This helps to accommodate any complex upgrade situation that can arise later on. However, for the "required feature" where the upgrade team is quite confident that all scenarios have been taken care of (please see feature requirements Extracts (porting ExtGen) and Extracts (New Extgen) as in sl. No. 4 and sl. No.5), then scenario for complexity level 5 has been described. However, later on if a more complex situation arises in "Extracts" then "complexity levels" in all the scenarios will be adjusted on a scale of 1 to 5. As of now, there are 24 known "required feature" that the customer can request for upgrades and these entire "required feature" are listed in table 1.

Table II lists effort required for each unit of these levels of complexity and is used in conjunction with guidelines explained in table 1. At the start of an upgrade project, effort estimations are arrived at based on past experience of the project manager. These efforts are fine tuned as the team gets more expertise on the matter and is able to deliver the required feature at a faster rate. Thus estimates arrived at the initial stages of upgrade projects can have dependency on project manager's expertise level, but as the project progresses, efforts are calculated on actual time undertaken to complete these activities and thus becomes accurate. All these efforts are given in person hours and vary depending on the level of complexity. For example, for the feature "PRF reports (porting)" (sl. No. 12), simple direct porting (please see table 1, sl. No. 12) has been categorized at complexity level 1 and would take Y_{121} hours to complete this. If there is more one porting required, then this number would be multiplied with Y_{121} to arrive at the estimation for the required feature. This process of estimation is done for the entire required feature and the total is arrived at the end using the mathematical model as shown below:

Total Effort in person hours E =

$$\sum_{\substack{m = 1 \text{ to } 24 \\ n = 1 \text{ to } 5}} X_{mn} * Y_{mn} \text{ Person Hrs.}$$

Where X stands for number of features required at different complexity level and Y stands for person hours required to complete the required feature at the corresponding complexity level.

This estimation in person hours is the effort required for complete lifecycle of upgrade projects.

Table 1: Guidelines.

Sl No	Feature	Complexity Level 1	2	3	4	5
1	Sortpattern - Total Number Of Modes (Each of the mode has to be classfied based on the complexity level as per the guide lines)	Singles EO COW SO Page Works	Image Pageworks Multiples Standard Modes with Custom Data Entry, Custom Stagers	Image Page Works with ICR Recoginition Standard Modes with Custom Data Entry and Custom Stagers	Image Page Works with ICR and Form Recoginition	Not Applicable
2	Sortpattern - Total Number Of Worksources	Standard Flow	No of Data Entry Formats <= 2	No of Data Entry Formats > 2 and <= 4	No of Data Entry Formats > 4 and <= 6	No of Data Entry Formats > 6 and <= 8
3	Sortpattern - Customization (CustSpg Changes)	Custom CDV Changes	Creation of New functions to be used in Sort Pattern	Not Applicable	Not Applicable	Not Applicable
4	Extracts (Porting ExtGen)	Simple extract with standard database fields output	Out put from custom fields like the userdata fields, Look ups.	With Custom Coding	With Field map, Sorting	With Field map, Sorting and Grouping
5	Extracts (New ExtGen)					
6	Extracts (Porting Dextract)	Direct porting without any modification	Code has to be relooked or optimised	With Sorting	With Sorting and Grouping	Not Applicable
7	Extracts (New Dextract)	Simple extract requirement	Transaction based requirements	With Sorting	With Sorting and Grouping	Not Applicable
8	Custom Coding (Porting for Data Entry Modules)	An entry for each function point. Custom hooks already exists, Code can be ported without any changes and a good understanding of the requirement.	An entry for each function point.New custom hooks has to be created	An entry for each function point.Code has to be relooked or optimised.	Not Applicable	Not Applicable
9	Custom Coding (New for Data Entry Modules)	Already existing Custom hooks	New custom hooks has to be created	Complex routines with transaction processing logic	Not Applicable	Not Applicable

10	Exports (Porting)	Simple with direct porting.	Custom Image Handling Requirements	Custom Image Handling Requirements as a Service instead of a application	Not Applicable	Not Applicable
11	Exports (New)	Simple	Custom Image Handling Requirements	Custom Image Handling Requirements as a Service instead of a application	Not Applicable	Not Applicable
12	PRF Reports (Porting)	Simple direct porting	With Sorting	With Sorting and Grouping	Not Applicable	Not Applicable
13	Crystal Reports (New)	Simple	With Sorting	With Sorting, Grouping and Sub Total	Complex Reports with Image Printing Requirements	Not Applicable
14	Data Entry Application (Porting)	Simple direct porting	Different Data Entry Formats, Conditions and Validations, coding has to be rewritten or optimized	Complex Image Handling Requirements	Multiple Screens, Complex Image and Transaction Based Data Entry Requirements	Not Applicable
15	Data Entry Application (New)	Simple direct coding	Different Data Entry Formats, Conditions and Validations	Complex Image Handling Requirements	Multiple Screens, Complex Image and Transaction Based Data Entry Requirements	Not Applicable
16	Custom Stager (Porting)	Simple direct porting	Code has to be relooked or optimised	New code has to be written in addition to porting to cater to the requirements	Complex Logic like transaction processing, applying balancing logic from variance.spg, print, merge stager	Not Applicable
17	Custom Stagers (New)	Simple direct coding	performance based stagers	New code has to be written in addition to porting to cater to the requirements	Complex Logic like transaction processing, applying balancing logic from variance.spg,	Not Applicable

					print, merge stager	
18	Track Driver Changes (Porting)	Simple direct porting	Code has to be relooked or optimised	New code has to be written in addition to porting to cater to the requirements	Complex Logic like pocket sorting, auto batching	Not Applicable
19	Track Driver Changes (New)	Simple direct coding	performance based coding	Complex Logic like pocket sorting, auto batching	Not Applicable	Not Applicable
20	Custom Modules (Porting)	Simple direct porting	Code has to be relooked or optimised	New code has to be written in addition to porting to cater to the requirements	Complex processing logic, service application	Not Applicable
21	Mark Sense (New)	Number of Zones for one sort pattern <= 4 (The units has to be specified based on the number of sortpatterns)	Number of Zones > 4 and <= 6	Number of Zones > 6 and <= 8	Number of Zones > 8 and <= 10	Number of Zones > 10 and <= 12
22	Custom Module (New)	Simple direct coding	performance based coding	Complex procesing logic, service application	Not Applicable	Not Applicable
23	ARC Integration	Only Forward Flow	Both Forward and Returns Flow		Not Applicable	Not Applicable
24	Check 21 - IQA, IUA	Only C21 Extract	C21 Extract with IQA \ IUA	C21 Extract with IQA, IUA	Not Applicable	Not Applicable

Table 2: Effort Estimation Table.

Sl No	Required Features	Complexity Level (X mn)					Effort required for each complexity level (Hrs) (Ymn)				
		1	2	3	4	5	1	2	3	4	5
1	Sortpattern - Total Number Of Modes	X11	X12	X13	X14	X15	Y11	Y12	Y13	Y14	Y15
2	Sortpattern - Total Number Of Worksources	X21	X22	X23	X24	X25	Y21	Y22	Y23	Y24	Y25
3	Sortpattern - Customization (CustSpg Changes)	X31	X32	X33	X34	X35	Y31	Y32	Y33	Y34	Y35
4	Extracts (Porting ExtGen)	X41	X42	X43	X44	X45	Y41	Y42	Y43	Y44	Y45
5	Extracts (New ExtGen)	X51	X52	X53	X54	X55	Y51	Y52	Y53	Y54	Y55
6	Extracts (Porting Dextract)	X61	X62	X63	X64	X65	Y61	Y62	Y63	Y64	Y65
7	Extracts (New Dextract)	X71	X72	X73	X74	X75	Y71	Y72	Y73	Y74	Y75
8	Custom Coding (Porting)	X81	X82	X83	X84	X85	Y81	Y82	Y83	Y84	Y85
9	Custom Coding (New)	X91	X92	X93	X94	X95	Y91	Y92	Y93	Y94	Y95
10	Exports (Porting)	X101	X102	X103	X104	X105	Y101	Y102	Y103	Y104	Y105
11	Exports (New)	X111	X112	X113	X114	X115	Y111	Y112	Y113	Y114	Y115
12	PRF Reports (Porting)	X121	X122	X123	X124	X125	Y121	Y122	Y123	Y124	Y125
13	Crystal Reports (New)	X131	X132	X133	X134	X135	Y131	Y132	Y133	Y134	Y135
14	Data Entry Application (Porting)	X141	X142	X143	X144	X145	Y141	Y142	Y143	Y144	Y145
15	Data Entry Application (New)	X151	X152	X153	X154	X155	Y151	Y152	Y153	Y154	Y155
16	Custom Stager (Porting)	X161	X162	X163	X164	X165	Y161	Y162	Y163	Y164	Y165
17	Custom Stagers (New)	X171	X172	X173	X174	X175	Y171	Y172	Y173	Y174	Y175
18	Track Driver Changes (Porting)	X181	X182	X183	X184	X185	Y181	Y182	Y183	Y184	Y185
19	Track Driver Changes (New)	X191	X192	X193	X194	X195	Y191	Y192	Y193	Y194	Y195
20	Custom Modules (Porting)	X201	X202	X203	X204	X205	Y201	Y202	Y203	Y204	Y205
21	Mark Sense (New)	X211	X212	X213	X214	X215	Y211	Y212	Y213	Y214	Y215
22	Custom Module (New)	X221	X222	X223	X224	X225	Y221	Y222	Y223	Y224	Y225
23	ARC Integration	X231	X232	X233	X234	X235	Y231	Y232	Y233	Y234	Y235
24	Check 21 - IQA, IUA	X241	X242	X243	X244	X245	Y241	Y242	Y243	Y244	Y245

TOTAL EFFORT = $\sum$ X mn * Y mn,
m= 1 to 24 and n = 1 to 5

APPLICATION TO LIVE PROJECTS

For application of the model, live projects were selected. The selected projects had the following characteristics:

a. They were from banking domain, and had development life cycle of twenty five man days to one man month (thirty man days).
b. The upgrade projects were started in May 2007 and completed in Dec 2007, which used many of the features studied in objective two.

After completion of projects, different metrics were measured such as effort deviation, schedule variation from estimated values. The final values were computed using similar formula used in earlier projects. The benefits obtained are discussed in the next section.

BENEFITS OBTAINED FROM THIS APPROACH

The approach helped to change the estimation culture in the organization. Estimation became systematic and scientific and predictability was higher. It changed culture and had impact on the work life balance of the team members. Nord et al. (2007) have indicated similar change in the culture in work relationships in their study. The model was applied in live project that was developed between May 2007 till Dec 2007. The benefits were many. It helped in accurate estimation (the accuracy increased by 11%) as well as helped in consistency for upgrade estimation process. The method for measuring the improvement in accuracy was dine by calculating process capability baseline for projects using the new model and comparing the values with respect to process capability baseline calculated for projects not using new models. Because of accuracy and consistency available, customers could also predict the effort required to complete the required features there by they (customers) were able to assess their upgrade budgets correctly. Because of the consistency and predictability customer's IT department could prepare and adhere to future roadmap for the installed products which would help them to map with business requirements of end users. J&B Software Inc. on the other hand can assess manpower requirements for upgrade projects which helped the vendor organization (J&B Software Inc.) to organize the upgrade team effectively, and above all this model helped in projecting annual revenue from upgrade projects accurately.

DISCUSSIONS

An upgrade project being a small project with less than a month man month duration, the requirements are usually of specific nature. They would consist of changing the database structure or improving performance factor or upgrading the application to the next version and while doing this upgrading, regression testing is carried out. The existing estimation models can not provide stable and predictive estimation as the average effort variation measured in process capability baseline (Process capability baseline indicates the parameters for measuring performance of different processes being executed in projects such as productivity, schedule variation, effort variation etc.) was higher than 20 percent. The effort variation showed that while existing models do not provide desired accuracy, the customer as well as project team members get worried about time to market and actual implementation period of the projects. The new model as shown in previous section, improved considerably (average effort variation was only 9 percent). The usefulness of the new model could be seen from the fact that resource utilization was higher than earlier (revenue productivity, measured by revenue divided by number of employees, increased by 3 percent) and implementation of projects was done as negotiated with the customers. Average implementation accuracy, as measured by schedule variation, was at 7 percent compared to 13 percent earlier.

APPROACH FOR ADDRESSING CULTURAL ISSUES IN CHANGE MANAGEMENT

The organization in question, J&B Software, had been using intuition based estimation for long time. The estimation technique was different for different persons and the final result varied based on experience, maturity and level of skill available with team. These factors affected estimation culture in the team and there was a variation in team culture. With introduction of model based systematic approach, the estimation technique will change and so also the existing

culture in the team. A change in the culture needs to be handled properly and effectively so that change management process becomes a smooth and sustainable process.

Ivancevich and Matteson, (1999) studied the relationship between culture and person dependent, intuitive approach. They found that a change in culture imposed on the resources would be seen as a shift in power in the team and would be have negative impact on the morale of the team members. As per Hearn and Southey (1996), Jim, Swamy, & Hicks, (2007), a systemic need for password change for better security measures is always considered as 'imposing' by the team members and they resent it resulting in de-motivation, loss of productivity, attrition etc. When these types of cultural issues come up, the relationship between teams becomes sour and organization as a whole would suffer.

To address these issues, a framework was developed by J&B. The framework was developed through discussions with consultants, senior managers and experts in human resources department. The approach adopted was:

1. A core team, headed by vice president (delivery) was set up which included business analysts who were part of estimation process. Roles and responsibilities of each team member was defined; critical success factors for measuring success of change in the culture were defined, such as, the number of teams using the new model for estimation, number of upgrade projects as a percentage of total upgrade projects using the new model, degree of accuracy obtained in each upgrade project that has used the new model.
2. Awareness for change in estimation techniques were increased among employees through communications. Several email communications were sent out by Managing Director and Vice President (delivery) stating that the estimation techniques need to change for better customer relationship. Meetings were held to allay any fear and doubts that the employees had.
3. Every Monday, status review meetings were conducted to understand different technical as well as behavioural issues related to usage of the new model. Issues such as fear for using the new model and losing jobs because of higher productivity were addressed.
4. Training sessions were conducted for using model based estimation techniques as a basis for estimation.
5. Rewards schemes were declared for teams that adopted model based techniques. Cash awards were offered to the teams that used the models.
6. Best practice sessions were held to share the experience from different teams. Care was taken that both positive and negative results were discussed and these learning were recorded for future reference.

REFERENCES

Abran, A., Symons, C., & Oligny, S. (2001). An Overview of COSMIC-FFP Field Trial Results. 12th European Software Control and Metrics Conference – ESCOM 2001, April 2-4, London (England).

Albrecht, A. J. (1979). Measuring application development productivity. Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, pages 83–92.

Albrecht, A. J. (1979). Measuring Application Development Productivity. IBM Applications Development Symposium. GUIDE Int and Share Inc., IBM Corp., Monterey, CA Oct 14-17, 83.

Armstrong, D. J. (2006, February). The Quarks of Object-Oriented Development. *Communications of the ACM*, 49(2), 123–128.

Boehm, B. (1981). Software Engineering Economics (original COCOMO). Prentice-Hall.

Box, J. (1983). Extending product lifetime: Prospects and opportunities. *European Journal of Marketing*, 17, 34-49.

Campos, J., Jantunen, E., & Prakash, O. (2007). Development of a maintenance system based on web and mobile technologies. *Journal of International Technology and Information Management*, 16(4), 1-8.

Chen, F., Romano, N. C. & Nunamaker, J. F. (2006). A Collaborative Project Management Approach and a Framework for Its Supporting Systems. *Journal of International Technology and Information Management*, 15(2), 1-16.

Conde, D. (2002). Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales (Execenablers), Aspatore Books.

Day, G. (1981). The product life cycle: Analysis and applications issues. *Journal of Marketing*, 45, 60-67.

Dhalla N. K. & Yuspeh, S. (1976). Forget the product life cycle concept. *Harvard Business Review.*

Dver, A. S. (2003). Software Product Management Essentials, Anclote Press.

Function Point Analysis for Software. (1997). Upgrade, August 2001.

IFPUG. (1998). Function Point Counting Practices Manual, version 4.1, International Function Point Users Group. http://www.ifpug.org

Jacobson, I. (1987). Object oriented development in an industrial environment, OOPSLA '87: Object-Oriented Programming Systems, Languages and Applications. ACM SIGPLAN, 183-191.

Jacobson, I. (1995). The Object Advantage: Business Process Reengineering With Object Technology, Addison Wesley.

Jim, T., Swamy, N., & Hicks, M. (2007). Defeating script injection attacks with browser-enforced embedded policies. *WWW '07: Proceedings of the 16th international conference on World Wide Web,* 601–610.

Jones, C. (1986). The SPR Feature Point Method, Software Productivity Research Inc.

Levitt, T. (1965). Exploit the product life cycle. *Harvard Business Review*, 43, 81-94.

Liu, Y., Adkins, G., Yao, J-F., & Williams, G. (2007). Discovering software reliability patterns based on multiple software projects. *Journal of International Technology and Information Management*, 16(3), 77-86.

Madu, C., Kei, C-H., & Madu, A. (1991). Setting Priorities for the IT Industry in Taiwan - A Delphi Study. *Long Range Planning*, 24(5), 105-118.

NESMA. (1997). Definitions and Counting Guidelines for the Application of Function Point Analysis. www.nesma.org.

Nindel-Edwards, J. & Steinke, G. (2007). The Development of a Thorough Test Plan in the Analysis Phase leading to more Successful Software Development Projects. *Journal of International Technology and Information Management*, 16(1), 65-72.

Nord, J. H., Nord, G. D., Cormack, S. & Cater-Steel, A. 2007. IT culture: its impact on communication and work relationships in business. *International Journal of Intercultural Information Management,* 1(1), 85–107.

Symons, C. R. (1988). Software Sizing and Estimating MK II FPA (Function Point Analysis) John Wiley & Sons.

Wu, R. (2009). Case study of micro-component design and industry integration. *International Journal of Intercultural Information Management*, 1(3), 248 - 258.

Whitmire, S. A. (1992). 3D Function Points: Scientific and Real-time Extensions to Function Points. *Proceedings of the 1992 Pacific Northwest Software Quality Conference*.

www.sei.cmu.edu/intro/process/technqs/q_estm.htm#14

www.UseCases.org