ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 5, May 2017

Review on Amalgamated Metrics Estimation Methods for Software Project Management

Vishakha Sharma[1], Gaurav Garg[2]

M.Tech Student, Department of Computer Science Engineering, AITM, Palwal, MDU University, India[1]

Asst. Professor, Department of Computer Science Engineering, AITM, Palwal, MDU University, India[2]

ABSTRACT: Estimating a project's effort, cost or schedule is a crucial task for software project management. However, project leaders are often overwhelmed when selecting an appropriate estimation method to best match the project characteristics and context.Amalgamated systems are applications that typically support online users when confronted with large sets of choices. Knowledge-based recommenders are a specific variant of these systems that exploit explicit knowledge models in order to infer matching items based on a set of specific requirements. This paper's contribution lies in its application of knowledge-based amalgamation mechanisms to the domain of software project management and presents a recommender for effort, cost, or time estimation methods. An initial evaluation among software professionals showed promising results and disclosed helpful hints for further development.

I. INTRODUCTION

In today's competitive software development environment it is of upmost importance to deliver products on time, with the desired quality attributes and at the specified cost. Depending on the specific project, estimation of size, effort and schedule can be done in an iterative way that is by primarily estimating requirements of the next development step. However, ever increasing competition among software companies often results in fixed-price projects. It is thus important to estimate the effort required, size and schedule of a specific project even in the early stages of software development. Thus accurate estimation methods like, for example, the Function Point method, T-shirt sizing or calibration with project specific data have gained increasing importance [1], [2]. Notably, this also holds for the development of custom software and integration projects, where functionality is primarily extended or substituted.

Although there is a large amount of literature on software estimation methods [1], [2], there is often no detailed knowledge about when to apply which method. To the best of our knowledge, no out of the box knowledge base exists that supports the selection and prioritization of adequate techniques depending on a specific project's characteristics. Consequently, there is a need for personalized advice, taking into account the specific project characteristics, the availability and granularity of comparable data and the current state of development.

In particular, the multitude of potential users, the fact that project managers are solely aware of a small number of relevant techniques and the availability of various techniques for diverse project settings suggests that constructing a amalgamation application for software estimation techniques would be worthwhile. Furthermore, harnessing a web based amalgamation system may also considerably contribute to the widespread applicability of software estimation techniques in today's mainstream software engineering projects.

Publicly available amalgamation applications for software engineering are in the scope of the Austrian Soft net research and innovation program. This paper contributes a knowledge based recommendation application that allows software project managers to select appropriate methods for effort, size and schedule estimation.

In Section 2 we discuss related work, namely amalgamation applications for software engineering, and Section 3 outlines our novel knowledge model for software estimation techniques. Notably, our model takes the cone of uncertainty [2] into account and thus reflects the accuracy of the different estimation methods at various stages of software development. Section 4 presents our pilot application and its user interface. Moreover, we report on a survey

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

about the proposed interaction flow, the results obtained and the underlying hypothesis. Finally, Section 6 concludes our paper.

II. RELATED WORK

The authors of [1] and [2] present more than 30 estimation methods and their associated properties. It is however hard and time consuming for a user to consider all the methods and then decide which one is best suited for the current project.

To the best of our knowledge no work has been done in trying to implement a amalgamation system for software estimation techniques. The authors of [3] present a amalgamated organization for project planning. This approach mainlysupports project planning by taking previous projects into account. It can be used solely when project-specific data is available from previous projects. Additionally this approach requires information like cost, size and well defined user requirements to be known. In contrast, our technique does not necessitate project specific data, but rather takes project specific data into account if it is available.

Amalgamation applications enjoy wide spread used in diverse fields of software engineering. The authors of [4] present a survey of different amalgamation tools for software maintenance and development. Based on automatic or manual queries, each tool presented in the survey is capable of amalgamating what should be done in the upcoming stage of ongoing software development efforts. Other works deal with improvement to developer productivity. The authors of [5] present a amalgamated system for identifying the most appropriate functions for implementing a specific software feature. Their approach is based on collaborative filtering algorithms [6]. Moreover the literature reports on successful applications of amalgamation systems for detecting software conflicts, as presented in [7], or for focusing software testing on specific modules or components [8].

III. KNOWLEDGE MODEL FOR AMALGAMATION

Knowledge-based amalgamated systems exploit a knowledge base that explicitly mediates between user requirements and the characteristics and limitations of different effort estimation methods in order to identify those items that best fulfill them. An overview on knowledge-based amalgamation systems can for example be found in [15]. The knowledge base for the amalgamation application presented here was derived by interviews with domain experts and by studying the popular literature on this topic [1], [2]. More concretely, the system for advising about effort estimation methods elicits the project's context like the development style or the phase the project is currently in via a forms-based dialogue (see Table 1 for the most important domain characteristics). The recommendable items and *effort estimation methods* constitute the product catalog*I* containing a finite set of database instances. Items are characterized by a set of properties like the required level of detail for historical data or their applicability to different project phases. Furthermore, the knowledge-based system takes a set of constraints, i.e. the knowledge base (*C*) and the specific requirements (*SRS*) describing the project context as input. We employ constraints that are represented as logical implications and consist of a condition (*If…..*) and a consequent (*then……*). In addition, constraints (*c*) possess a weight (w_c) that characterizes their relative importance and may be defined as either hard ($c \in C_{hard}$) or soft ($c \in C_{soft}$). The latter means that they can be ignored if no other solution can be found. The specific requirements and the condition parts of constraints are represented by terms utilizing the context variables described in Table 1. Therefore, the function *app*(*C; SRS*) returns all constraints from *C* that are applicable in conjunction with *SRS*. Consequently, recommended items need only to satisfy the consequent parts of applicable constraints. We denote the consequent part of constraint *c* by *cons*(*c*).

In general, a recommender computes a function *rec*(*i, u*) for a given item *i* and an user *u* that returns a amalgamation score. It expresses the system's belief as to whether the item is of interest for the user. In case the amalgamated system's knowledge base consists solely of hard constraints, the assumed utility scores either 1 if an item satisfies all applicable constraints or 0 otherwise. However, in cases where the knowledge base contains soft preference constraints the constraint-based amalgamation system is able to compute scores that reflect the sum of weights of fulfilled constraints with respect to the sum of weights of constraints that were *relaxed* in order to include the specific item in the result set. Given product table I, specific contextual requirements SRS_u and the relational

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

selection operator δ we define that item i satisfies the conditional constraint c iff $i \in \delta_{[cons(c)]}(I)$. Therefore, our implementation computes amalgamation scores as follows:

$$Rec(i,u) = \begin{cases} \dfrac{\sum_{c \in app(C,\, SRS_u)\ i \in \delta_{[cons(c)]}(I)} w_c}{\sum_{c \in app(C,\, SRS_u)} w_c} & : * \\ 0 & : \text{else} \end{cases}$$

$*$for all $c \in app(C_{hard}, SRS_u)\ i \in \delta_{[cons(c)]}(I)$

Consequently, $rec(i; u)$ is the sum of the weights of all constraints that, in conjunction with the specific project context (SRS_u), support item i to be in the result set. $^{P}w_c$ in the denominator normalizes the utility score to the range 0..1. In cases where not all of the hard constraints are able to be fulfilled the score is set to 0. This approach was already proposed in [9] where constraints have been learned from past user interactions. Our computation scheme for recommendations contrasts the knowledge-based relaxation mechanisms presented in Mirzadeh et al. [10] or Jannach [11] who search for query relaxations on the product catalog that determine the result set. While they compute recommendations where each item satisfies the same set of constraints, we rather assign ranks to items based on their individual constraint fulfillments.

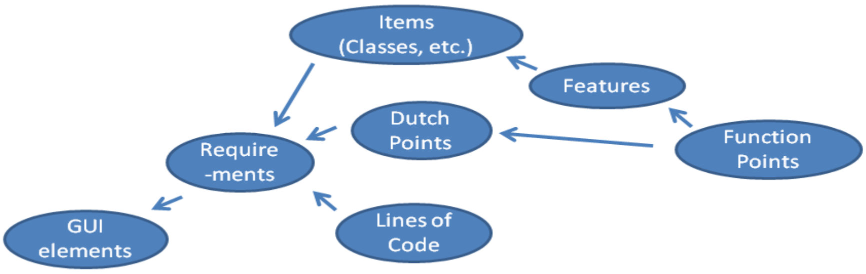

Figure 1. Dependencies among granularity levels of historic data and countable items

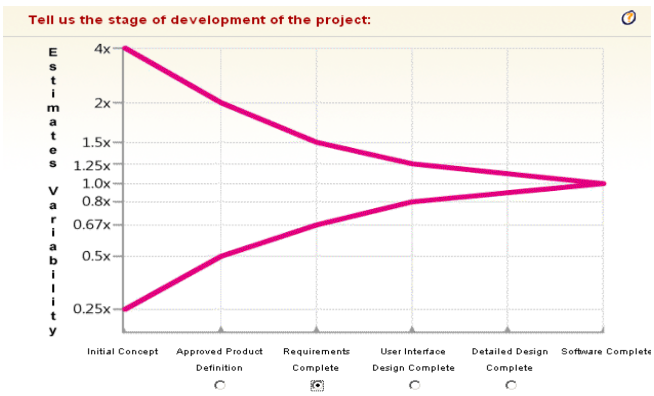

Figure 2. Sample question

The knowledge base consists of around 50 different constraints, some of which are rather obvious like *If the project follows development style X then the method should be applicable to development style X*. However, matching the historic data requirements of estimation methods with the granularity of the available historic data and the countable items of the current project data is more complex. Figure 1 sketches the assumed dependencies between different classes of countable elements for deriving software estimates. The directed edges indicate an informal containment

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 5, May 2017

hierarchy. For instance if one can compute Function Points (FP) then - in principle - Dutch Points (being only a subset of FP elements) or method-level features may also be computed.
Therefore, only estimation methods that are applicable to the figuratively *greatest common divisor* or lowest granularity of the available historic data and the countable data in the current project must be selected.

IV. IMPLEMENTATION

The implementation is based on a generic amalgamation framework [12] that can be instantiated in different application domains. Figure 2 depicts a sample dialogue page that requires the user to specify the current project phase and informs him/her of the variability that can be expected in effort estimation methods at this point. Figure 3 shows parts of the result page of the system, listing the estimation methods that are applicable to the given project context together with some additional explanations such as under what circumstances the method is applicable or some form of disclaimer like *T-Shirt sizing fits only the current iteration* when proposed in the context of an iterative development process. In addition the application has additional features such as dynamic interaction flow based on user input as well as a type of inconsistency detection for cases where user input seems to be contradictive as presented in [13].

The prefered estimation methods are:

Method	Description	Details
T-Shirt Sizing	Used when working with stakeholders. Helps the stakeholders understand the ratio between features and there development costs. Unnecessary functionalities can be eliminated. Wide area of the cone. Each feature/requirement receives two grades, one for its importance and one for its costs. Warning: T-shirt sizing fits only to the current iteration. This method is applicable to an iterative development style. This method is already applicable at the start of the project. The method is applicable to medium-sized projects.	Why?

Figure 3. Results page

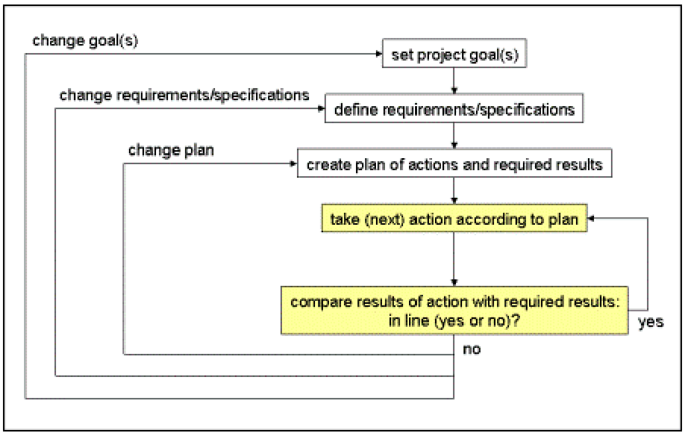

Figure 4. Implementation Phase

V. EVALUATION

To evaluate the initial version of our amalgamation application worker prepared a questionnaire and asked our industry partners (big companies as well as small and medium-sized enterprises) for their feedback. Primarily project managers, software engineers and IT consultants tested our application and subsequently answered the questionnaire. For this evaluation we hypothesized that potential users are coarsely familiar with software estimation techniques. Our survey's main intention was to evaluate (1) the practical applicability and usefulness of the system for mainstream software engineering projects, (2) the completeness and comprehensiveness of the proposed methods, (3) the characterization of the project and the project's context, and (4) the traceability of the given amalgamations. The

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 5, May 2017

seven participants of our survey reported diverse intricacies. In the following we discuss the most notable suggestions for further improvement categorized according to the criteria given above.

Magnitude of relative error(MRE): First calculate the degree of estimating error in an individual estimate for each data point as project .It is defined as:-

$$MRE = \frac{|\text{Predicate value} - \text{Actual value}|}{\text{Actual value}}$$

Root Mean Square Error(RMSE): It is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modelled or estimated. It is just the square root of the mean square error as shown in equation given below:-

$$RMSE = \sqrt{1/n \sum_{i=1}^{n} (\text{actual value} - \text{predicate value})^2}$$

Mean Magnitude of Relative Error(MMRE): It is another measure and is the percentage of the absolute values of the relative errors, averaged over the N items in the "Test" set and can be written as:-

$$MMRE = 1/n \sum_{i=1}^{n} \frac{|\text{Predicate value} - \text{Actual value}|}{\text{Actual value}}$$

PRED (N): Itis the third criteria used for the comparison and this reports the average percentage of estimates that were within N% of the actual values .It is commonly used and is the percentage of predictions that fall within p % of the actual, denoted as *PRED (p)*, k is the number of projects where MRE is less than or equal to p, and n is the number of projects.

$$PRED (p) = k / n$$

VI. CONCLUSION

This paper contributed a knowledge-based amalgamation application to the domain of selecting appropriate effort, cost and time estimation methods for software project management. The results of our survey on the pilot system confirmed our assumption that tailoring a amalgamation application for project managers and engineers - in contrast to the typical tailoring for sales representatives - requires a very precise formalization of the underlying problem. However, the answers to our questionnaire highlighted that such a amalgamation application might be used in practice. This holds particularly for a complex knowledge base as a amalgamation system might have genuine value for entering even more detailed facts about the project under consideration. In addition several technical issues and the need for further clarification of the capabilities of diverse methods (e.g. requesting the application domain directly vs. conveying that the application domain is an input to the method being proposed), traceability must be further improved if the system is to be widely adopted. In the near future we plan to extend the knowledge base to cover more project details. Besides of this, a user receiving a certain amalgamation might be interested in obtaining the necessary information to extend into an accurate method. We plan to address this by incorporating ideas from diagnosis [14] into our amalgamation engine.

REFERENCES

[1] M. Bundschuh and C. Dekker's, *The IT Measurement Compendium Estimating and Benchmarking Success with Functional Size Measurement*.Springer Publishing Company, Incorporated, 2008.
[2] S. McConnell, *Software Estimation: Demystifying the Black Art (Best Practices (Microsoft))*. Redmond, WA, USA: Microsoft Press, 2006.
[3] C.-S. W. Heng-Li Yang, "Recommendation system for it software project planning: a hybrid mining approach for the revised cbr algorithm," in *ICSSSM'08: Proceedings of the 5th International Conference on Service Systems and Service Management*, Melbourne, Australia, 2008, pp. 670 – 674.
[4] M. W. HappelHans-J¨org, "Potentials and challenges of recommendation systems for software development," in *RSSE '08: Proceedings of the 2008 international workshop on Recommendation systems for software engineering*. New York, NY, USA: ACM, 2008, pp. 11–15.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

[5] N. Ohsugi, A. Monden, and K. Matsumoto, "A recommendation system for software function discovery," in *Software Engineering Conference, 2002. Ninth Asia-Pacific*, 2002, pp. 248–257.
[6] M. R. W. Hill, L. Stead and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *In Proc. of the 1995 Conference on Human Factors in Computing Systems (CHI95)*, 1995, pp. 194–201.
[7] D. Prasun, "Dimensions of tools for detecting software conflicts," in *RSSE '08: Proceedings of the 2008 international workshop on Recommendation systems for software engineering*.New York, NY, USA: ACM, 2008, pp. 21–25.
[8] S. Kpodjedo, F. Ricca, P. Galinier, and G. Antoniol, "Not all classes are created equal: toward a recommendation system for focusing testing," in *RSSE '08: Proceedings of the 2008 international workshop on Recommendation systems for software engineering*. New York, NY, USA: ACM, 2008, pp.
6–10.
[9] M. Zanker, "A Collaborative Constraint-Based Meta-Level Recommender," in *2nd ACM International Conference on Recommender Systems (ACM RecSys)*. Lausanne, Switzerland: ACM Press, 2008, pp. 139–146.
[10] N. Mirzadeh, F. Ricci, and M. Bansal, "Supporting user query relaxation in a recommender system," in 5*th International Conference on E-Commerce and Web Technologies (EC-Web)*. Zaragoza, Spain: Springer, 2004, pp. 31–40.
[11] D. Jannach, "Finding preferred query relaxations in contentbased recommenders," in *IEEE Intelligent Systems Conference (IS)*. Westminster, UK: IEEE Press, 2006, pp. 355–360.
[12] M. Jessenitschnig and M. Zanker, "ISeller: A Flexible Personalization Infrastructure for e-Commerce Applications," in *10th International Conference on Electronic Commerce and Web Technologies (EC-Web)*, Linz, Austria, 2009.
[13] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker, "An integrated environment for the development of knowledgebased recommender applications," *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 11–34, 2006.
[14] B. Peischl, F. Wotawa,'Model-Based Diagnosis or Reasoning from First Principles.' *IEEE Intelligent Systems* , pages 32-37, May/June 2003, Vol. 18, No. 3.
[15] R. Burke, 'Knowledge-based Recommender Systems.', *Encyclopedia of Library and Information Systems.*, Vol. 69, Supplement 32, 2000.