A Cognitive Model for Meetings in the Software Development Process

Sanjay Misra and Ibrahim Akman

Department of Computer Engineering, Atilim University, Ankara, Turkey

Abstract

Meetings are at the heart of the software development process (SDP) and can be of different types. The present article first proposes an abstract cognitive model for meetings, which represents how different types of meetings are affected by cognitive activities at different stages within the SDP. Second, and based on the analysis of meetings at different stages of SDP, it proposes the removal of such meetings from some of the stages within the program by using a cognitive evaluation model for meetings and their replacement, instead, with information and communication technology tools and techniques by means of a cognitive evaluation model. The abstract cognitive model and the evaluation model are validated empirically through experimentation, carried out through a detailed analysis of a target group composed of information technology professionals.

Keywords: Cognitive factors; Cognitive model; Development process; Meetings; Phases; Software

1. INTRODUCTION

The need for meetings in the software development process (SDP) is due to the fact that the process requires constant, well-measured, and appropriate decisions. Unlike other industries, in most cases the requirements cannot be clearly defined from the beginning of the process. Although such requirements are normally gathered by experts through several meetings with the stakeholders, there is no guarantee that the software products will fulfill the customers' needs. Common practical problems include complexity of the requirement, customers' inability to know all the requirements in advance, changes in requirements, difficulties in managing frequent changes, process bureaucracy, difficulties in estimating development times, budget overrun, and the application of wrong processes during SDP (Hazzan, 2005). To overcome these hurdles, meetings are among the effective tools that reduce complexity (see, e.g., Berntsson-Svensson & Aurum, 2006; Fagan, 1976; Gilb & Graham, 1993) and, hence, they are crucial in every step of SDP. Meetings have also been recommended to improve the quality of the product itself; for instance, meetings for the review process (Gib & Graham, 1993). There are, in contrast, disadvantages to meetings, such as elongating the development time by reducing the speed of SDP. Meetings can also result in wasting the time of some professionals whose contribution to such meetings is less and not significant compared to that of others.

The literature provides many inconclusive discussions on the relevance of meetings in the SDP. Some researchers fully support meetings (Berntsson-Svensson & Aurum, 2006; Gilb & Graham, 1993) whereas others are not in favor of them. According to the second group, meetings are not always valuable in improving the quality of the product and, as such, should be avoided in some cases (see, e.g., Johnson & Tjahjono, 1998; Porter & Johnson, 1997; Vota, 1993). Robert, Martin, & Sibbet (1991) suggest adopting physical meetings if the team members are working within the same time

Correspondence to: Sanjay Misra, Department of Computer Engineering, Atilim University. Phone: +903125868348; e-mail: smisra@atilim.edu.tr

Received: 27 December 2010; revised 21 April 2011; accepted 28 April 2011

View this article online at wileyonlinelibrary.com/journal/hfm

DOI: 10.1002/hfm.20344

frame and at the same location; they also suggest several techniques for holding the meetings, such as phone conversations, instant messaging, notice boards, pigeon–holes, e-mails, and voice mail. These techniques are explained in the *Discussion* section with examples.

According to a report by Gilb (2007), at least 10% of all software projects are "failures" (cancelled before completion), and another 50% are "challenged." One of the reasons for these crises is the dissatisfaction of the customers due to either unsatisfactory requirement(s) or a low-quality product. Both factors can be dealt with to a certain extent through meetings required at almost every step of the SDP. In fact, a majority of such problems can be solved in several meetings in the entire life cycle of the process when experts can interact. In these meetings, one of the quality factors is related to human behavior and can be treated as a cognitive activity. More specifically, the decisions made in the meetings are the collective efforts of all the participants, and so are generated in human terms – that is, a cognitive process. In other words, cognitive behavior (Sanderson & Fisher, 1994) and complexities of individuals (Fagan, 1976, 1986) play an important role in the meetings throughout the SDP. Activities involved in the process and in meetings, however, have so far been studied to a limited extent, establishing the motivation for this research to investigate the relevance of meetings in the SDP.

The preliminary work for this study was first introduced by Misra and Akman (2009). In related works, we have analyzed the role of leadership's cognitive complexity in terms of simplicity in software development products (Akman, Misra, & Cafer, 2011) and the impact of cognitive and sociodemographic factors in meetings within the SDP (Akman, Misra, & Altindag, 2011). The present article is entirely different from the related works, and we have extensively revised the previous one (Misra & Akman, 2009) to include experimental results and related analysis. We propose an abstract cognitive model for meetings – or ACMM – representing the effect of cognitive activities in different meetings at different stages of the SDP. Additionally, we propose a cognitive evaluation model for meetings - or CEMM - which is based on cognitive perspectives. Both models are validated empirically. For this purpose, a survey approach was adopted, and the necessary data were obtained by means of a questionnaire applied to a number of information technology (IT) professionals in a one-day workshop organized by a leading international software company. Although cognitive factors influence the entire development process in different ways, our study is restricted to the cognitive activities in meetings through the SDP.

The organization of the article is as follows: Section 2 discusses the relevance of meetings at different stages of software development along with the opinion of other researchers. This section also evaluates meetings from the cognitive point of view. Section 3 proposes an abstract model ACMM, which is based on the different types of meetings possible at different stages. Section 4 proposes the CEMM. Section 5 provides an empirical study to validate our models. In section 6, related observations and discussions are provided, and, finally, the conclusion appears in section 7.

2. EVALUATIONS OF MEETINGS AND COGNITIVE ASPECTS

The SDP is a complex activity; several meetings are required at each stage to come to a decision. In all these meetings, team members and group leaders make decisions that reflect their own cognitive behavior, which is defined as the ability to judge and reason effectively and to obtain a perception of surroundings. In other words, cognitive factors influence the success of meetings in the SDP. To achieve quality objectives, various types of meetings are important at different phases of the SDP where cognitive activities play key roles in succeeding in reaching a consensus. In the following sections, first, we discuss the relevance of meetings at different stages throughout the process and, second, we evaluate such meetings using cognitive features. This evaluation is paramount because it establishes the foundation on which the abstract model is based in the section following it.

2.1. Requirements Analysis Phase

The requirement phase is the most important phase of any SDP, as this is when one needs to find out exactly what is required (Bray, 2002). Loosely defined requirements or vague ideas in definitions often result in low-quality software and can also end up in budget deficits and even project failure (Daniel, 2002).

One of the outstanding problems in this phase is to understand and analyze both the user's and the stakeholder's needs so as to build a system that meets those needs. In fact, requirement specifications and dealing with customer demands are the two main issues in

the software industry (Leffingwell & Don, 2003) and, as such, two of the leading root causes of software failures. To overcome this barrier, the literature has encompassed a variety of techniques such as interviewing, workshops, meetings, brainstorming and idea deduction, storyboarding, and so forth (Leffingwell & Don, 2003). In this respect, meetings and workshops are known to be the most important and effective techniques to elicit the actual needs of the users and stakeholders at this stage. For detailed examples, we refer the readers to Leffingwell & Don (2003, p. 126).

On top of that, from the cognitive point of view, this type of gathering where all stakeholders meet together is helpful in increasing awareness of the collective activities involved in the SDP (Porter & Johnson, 1997). Also, it provides the necessary knowledge to assess the guidelines at hand and, henceforth, to adopt them to the end user's needs. The type of meetings depends, however, on the features and requirements of the products. For instance, if the product is large in size, a full-day workshop seems to be a good solution for eliciting requirements because such workshops are believed to be cognitively productive for the sake of learning about the actual needs of the users; they tend to assemble all key stakeholders together for a short, but intensely focused period (Leffingwell & Don, 2003). These meetings/workshops must be chaired by experienced software professionals, who also have a background in organizing such workshops/meetings. In addition, they require an effective group of software professionals able to extract all the necessary information regarding the requirements from the gathering of customers.

In contrast, for small products of which the number of users and the stakeholders are fewer, one can consider the organization of small and less-complicated meetings instead of workshops as they do not require as much experience to be organized, and are also relatively easier to manage. These workshops and meetings should be repeated until attendees reach a clear understanding of the real needs of the customer(s).

With this in perspective, all the just mentioned workshops/meetings – based on cognitive processes – prove crucial in identifying the actual needs of the customer. The output from this phase (the requirement phase) is the software requirements specification (SRS) – a formal document for the software specifications to be produced and prepared by the team members after several meetings with the customer.

2.2. Design Phase

After eliciting the requirements and producing the specifications of the software to be developed, the software design phase follows. Naturally, almost all of the decisive meetings in this phase are between the development team and the customer. These sessions are vital because, if any requirement is left out or missed in any way, such deficiencies will be reflected in the design and can be easily observed by the customer. At this stage, the design can be modified far more easily than in later stages and, because the design becomes the basis for the project, meetings should be held with customers and repeated until the customer is satisfied.

The design for the software based on SRS depends on the knowledge and intelligence of the designer – an activity based on memory (Wang, 2009a). Furthermore, the mechanism of knowledge in memory plays a crucial role in representation as the same problem/specifications can be arranged in different ways, naturally affecting the quality of the work depending on the cognitive skills of the designer. If possible in such circumstances, different people and teams should develop alternative designs and, later, evaluate them in collaboration with the customer to choose the one that fits and satisfies the customer's needs most. This practice may not always be possible, however, due to an insufficient number of qualified designers, consequently causing delays in projects (Laitenberger & Debaud, 2000).

2.3. Coding Phase

A gathering with the customers during the development of the code is neither a common practice nor a compulsory one, but meetings among the software developers can be held depending on the requirements, circumstances, and needs. Also, regular short meetings among the software developers (e.g., during tea time) may be useful to solve the minor problems faced by the individuals involved during the development of the code. These types of meetings (e.g., in daily Scrum: An agile development method) may not only save time for meetings, but they also boost the development process because most of the individual's problems can be solved during such types of interaction. It is worth mentioning here that the developer should not intend in such meetings to solve major problems (for which more appropriate and formal gatherings need to be organized between the team leader and the experts).

Furthermore, this way of communication, as a stepping stone in the exchange of ideas for development, is a sound practice in a global software development environment where any doubt should be clarified at the earliest stage through the exchange of ideas with other developers.

From the cognitive point of view, the quality of coding depends on the individual's cognitive characteristics (Hazzan, 2005; Wang, 2009b), such as the ability to define and solve problems. Having meetings at this stage, nevertheless, is not the appropriate way to improve the quality of the code. For this reason, meetings may be organized to review the code in its entirety – an issue to be discussed in the next section.

2.4. Software Review Phase

The software review process is an integration of several technical tasks carried out by individuals using cognitive activities. In fact, the process itself is a cognitive one. Fagan (1976, 1986) was the first to introduce code/software review to improve the quality of code and increase software developers' productivity; later, several similar methods were established by others for software inspections. We refer the readers to Laitenberger and Debaud (2000) for a survey on different software inspection methods. The software community has adopted Fagan's approach as a technique for best practice in the industry. The technique consists of six steps: planning, overview, preparation, group meeting, reworking, and follow-up. At the heart of this process lie meetings by aiming to raise the awareness of the reviewers to understand the overall scope and the purpose of software review after planning the review process.

Since software review was first introduced by Fagan (1976), group meetings as a part of the inspection process – where the review teams meet to find and collect defects – have been a common practice. In fact, Fagan insists that group meetings are more effective in reducing errors due to their synergy effect, also regarded as a cognitive activity. He assumes that such an effect can yield the identification of new faults that were previously undetected by individuals. Cognitive principles support Fagan's idea for group meetings because interactions increase productivity and the presentation of new ideas. Also, a number of other researchers have fully supported such meetings in their works (examples include Gilb & Graham, 1993; Sauer, Jeffery, Land, & Yetton, 2000). Another group of researchers emphasizes that meetings are not as valuable in improving the quality of the product and, as such, should be avoided (see Johnson & Tjahjono, 1998; Porter & Johnson, 1997; Vota, 1993). It is important to mention here that the authors of the present article are not in favor of Fagan's original idea of group meetings, mainly because each group meeting covers only a small portion of the software, demanding the entire process to call for several meetings in this way. Additionally, in our opinion, the value of group meetings depends on various factors, such as the managerial experience at various levels, the availability and types of tools to be used, limitations and circumstances of the project, and so forth.

With the highlights just mentioned, one cannot overlook the importance of meetings at the review stage. It is a highly efficient method that cannot be substituted by any other method (Wang, 2006). Sometimes it may be time-consuming but, according to the statistics, it will solve up to 90% of the errors if done properly (Wang, 2006). This article agrees with the usefulness of meetings for software review but, in our opinion, strict rules should not be established at the beginning of the development process. For instance, if the product is small, then the meeting may be confined to the author and the reviewer. Other factors, such as experience and mental mechanisms (or cognitive processes) of the reviewers and the authors, are also important in deciding whether to set up meetings and, if so, of what type. If the software author does not have sufficient experience in software development, then the chances of causing errors in his product/subproduct are relatively high in comparison to other modules of the system developed by other software developers. As a consequence, meetings should focus on errors in those modules instead of spending time on the other, less error-prone parts.

The same discussion is also true in the case of the reviewers' experience, mental mechanisms, and expertise. The authors believe that the reviewers should be selected from the ones familiar with the concerned product. This way, time can be saved for training the reviewers, leading to a better inspection of the code. Here, we should point out that if the product is small and the errors related to software review can be solved by other means, then there is no need for any meetings. On the contrary, if the software is large, obviously it will need to be divided so that thereviewers inspect the parts separately and in isolation. In this case, error-prone modules should be identified, and meetings should be concentrated especially on

those modules between the reviewers of those modules and their authors. Furthermore, we suggest using other means for less error-prone modules. Information and communication technology (ICT) techniques – phone conversations, e-mails, chatting, computerized tools, video-conferencing, voice chatting, blogs, and so forth – which support facilities such as computerized checklists, annotation support, automatic defect detection, distributed meeting support, decision support, and metric collections, provide effective media and are promising alternatives in this case. These tools are also able to detect the amount of effort made by the reviewers in a review process, and to support separation of development teams due to geography and country.

As stated by Robillard, Astus, Detienne, and Visser (1998), the physical meeting at the review process is composed of three types of cognitive activities: review, elaboration of alternative solutions, and synchronization. Full software review is made up of a set of technical tasks, and individuals are responsible to carry out these tasks using cognitive activities. Here, alternative elaborations are concerned with a reviewer's proposal for a solution not originally described in the review documents. Another process, cognitive synchronization, relates to a common solution/representation of the design solution or evaluation criteria among the participants, and depends entirely on the cognitive characteristics of those individuals (especially the authors and the reviewers). Furthermore, the decision on controversial issues arising in meetings is also relevant to the cognitive skills and features of the quality engineer.

Based on this discussion, we can easily observe that software review meetings are beneficial; yet, such benefits and outputs should be analyzed equally and compared with the approaches that favor a no-meeting-at-all attitude. What is more, it is important to keep in mind – prior to arranging any meetings at all – that one of the major objectives in SDPs is to reduce time and budget, whereas physical meetings cause an increase in both cases.

2.5. Software Testing Phase

To a large extent, the type of meetings at the testing phase depends on the method of the development process. Normally, after completing the software review, it is assumed that all the modules are correct and that they satisfy most of the requirements. Later, and upon the integration of all the modules, the final product has to be tested extensively – for instance, tests carried out on documentation, availability, reliability, stress, security, usability, and so forth (Daniel, 2002), for which the assurance of completeness requires meetings among the persons involved.

Companies of medium and small scale, in particular, tend to have the misconception that testing is a process to be taken up only when the software has been completed. Hunt, Thomas, and Hargett (2008) also support the above statement by stating that many organizations have grand intentions for testing only toward the end of a project. When testing is organized after the completion of the code, meetings are indispensable to check whether the product fulfills all the requirements, and that it is error-free and complete in all aspects. Any modifications or improvements required should be, then, accomplished after meetings have been concluded. Again, if such modifications and improvements are of serious types, additional gatherings will be required to check whether the improvements do, in fact, satisfy the user's demands. At this stage, meetings are necessary because, if a major modification/improvement has been done, there is still the risk of new errors, those which could otherwise be overlooked without meeting.

In contrast, software testing is not a task to be accomplished when the product has already been completed, and should begin with a proper plan designed as early as the requirements stage itself. If this process is adopted, then meetings are required according to a test plan and at each phase as discussed in the previous sections. In addition, when the software has been tested starting from the requirement phase, only one meeting is required at the end, merely to assure that the software is complete in all aspects.

Evaluated based on relevant cognitive factors, meetings are required at this stage with top officials and the development team members (including testers) because important decisions are to be made; for example, those on issues that cannot be solved easily by the development team and – if time is a pressing factor to release the product – a collective decision arrived at through a meeting is a better solution. In a worst-case scenario such as project failure**,** decisions made in this way also reduce the liability of each individual by being collective. The chances for such circumstances are even lower if the testing process starts at the beginning of the SDP, as detailed in the previous paragraph.

3. AN ACMM AT DIFFERENT PHASES OF SOFTWARE DEVELOPMENT

Based on the discussion on the relevance of meetings at the different stages of the SDP as well as the cognitive aspects mentioned beforehand, it is appropriate to develop an abstract model for meetings which can help to understand the proposed cognitive model more elaborately. Figure 1 shows the ACMM, which includes different types of meetings affected by cognitive activities at different stages of the SDP.

The proposed ACMM is the combination of submodels for meetings at the different stages of the SDP and is representable by the following mathematical expression:

ACMM = {Rc, Dc, Cc, Ic, Tc},

where

Rc is the cognitive effect on meetings at the requirement phase;

Dc is the cognitive effect on meetings at the design phase;

Cc is the cognitive effect on meetings at the coding phase;

Ic is the cognitive effect on meetings at the inspection phases of the code (i.e., software review); and

Tc is the cognitive effect on meetings at the testing phase.

Each of the submodels (Rc, Dc, Cc, Ic, and Tc) is composed of a set of different types of cognitive effects on meetings at each phase; that is:

Rc = {Uc, Oc},

where Uc is the cognitive characteristics of the client(s)/user(s), and Oc is the cognitive characteristics of the organizer(s).

Dc = {DEc},

where DEc is the cognitive characteristics of the designers.

Cc = {COc},

where COc is the cognitive characteristics of the coders.

Ic = {Rc, Sc, Mc, Qc},

where

Rc is the cognitive characteristics of the reviewers;

Sc is the cognitive characteristics of the software developers;

Mc is the cognitive characteristics of the moderators; and

Qc is the cognitive characteristics of the software quality engineer (SQE).

Here, S refers to the software developer, who is different from thecoder (previously represented by CO), in the sense that a software developer may be any individual involved in the development process and attending the meetings for the review process, such as the group leader.

Tc = {TEc, Lc},

where TEc is the cognitive characteristics of the testing team members, and Lc is the cognitive characteristics of the testing team leader.

If we integrate all the factors cognitively affecting the meetings in the entire SDP cycle, they can be expressed as follows:

ACMM = Cognitive characteristics of {clients/user + designer + coder(s) + Reviewer(s) + Moderator (Software Quality Engineering) + Software development team members + Testers}

As a summary, our model represents that the quality of the final software product also depends on several meetings normally occurring at the SDP's various stages while depending extensively on the cognitive capabilities of all the persons involved in the process.

4. A CEMM

There exists a large number of issues related with the software development team members and customers arising from intangible behavior in the software (Hazzan, 2005, 2006), thereby making the development process nontransparent and more complicated in comparison to a tangible process. The intangible behaviour of the software also increases the cognitive complexity of the SDP. To overcome these drawbacks, meetings are good alternatives to dilute problems related with this (intangible behaviour).

In the previous section, cognitive effects that can play a role at the different stages of SDP were highlighted. In fact, software development is knowledge-intensive (Robillard et al., 1998) and made up of a set of practices carried out by individuals using cognitive activities. Figure 1 demonstrates an abstract model representing the different types of meetings affected by cognitive activities at various levels of the SDP. To our knowledge, however, cognitive activities in this respect have not been studied in detail yet – which issue can be explained by the fact that cognitive science as a part of the curriculum for software scientists and/or engineers is open to further discussion (Robillard, 1999).

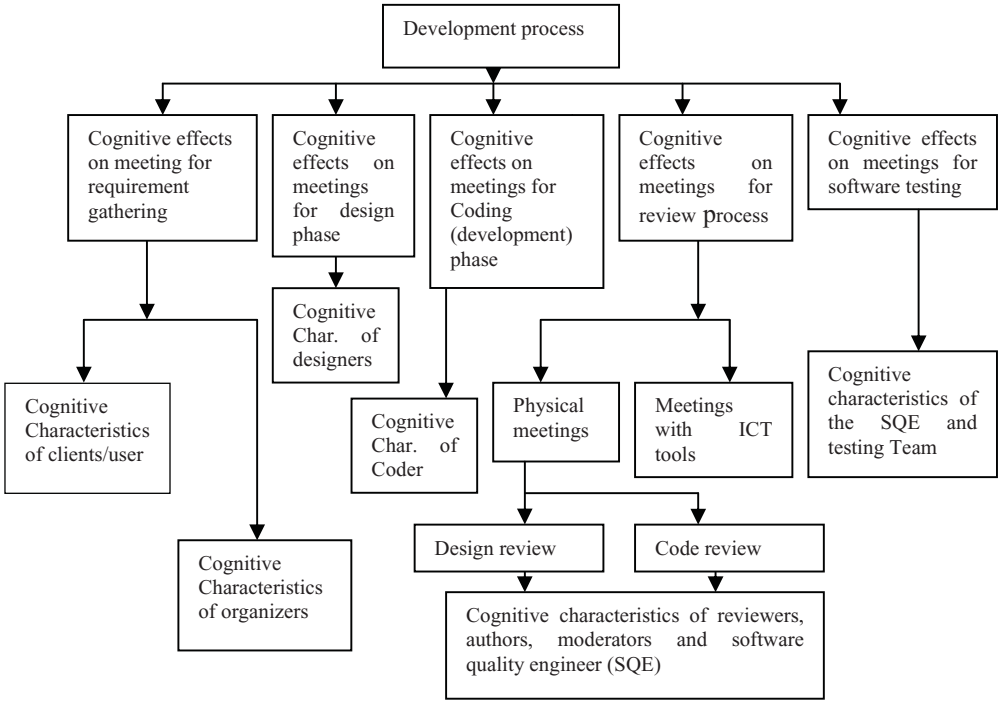

Figure 1 ACMM: Cognitive effects on meetings at different levels of the SDP.

Meetings have both merits and perils when cognitive factors are considered. An important positive cognitive effect of meetings is that they increase the knowledge and awareness of the collective activities. Nonetheless, such activities also possess negative consequences brought about through human behavior. The main problems arise from the naturalness of such meetings. In most cases, attendees tend to engage in behaviors and habits, such as using technical words that others cannot understand, speaking at the same time, interrupting others, start discussing outside the agenda, and so forth (Robillard, 1999). All these, of course, have negative effects on the outcome of these meetings and, consequently, the development process itself. By considering all these points, we propose a model (Figure 2) for making appropriate decisions for meetings. In this model, to arrange physical meetings only for unavoidable circumstances and to prevent – or, at least, minimize – the negative impacts of cognitive activities, we propose the following practices:

- **Modeling meetings**
  The type of meetings and their structure should be decided at the beginning of the development life cycle. The model, then, is developed to include all the related items and the corresponding agenda. In addition, for each stage of the SDP, the scope and the objectives of the meetings must be decided in advance. (Steps 1, 2, and 3)
- **Analyzing the proposed meetings**
  Analyze critically the usefulness of each one of the meetings for each phase of the SDP. At this stage, it is also possible that there exist no clear-cut proposals for meetings for some of the phases due to unclear requirements – in which case, additional meetings with the customers will be required. (Step 4)
- **Using cognitive modeling**
  This will reduce the effect of interferences. A cognitive psychologist may initiate dialogs with individuals for different purposes in a formal way and, then, produce certain models for cognitive activities. This approach not only models successfully the channeling of the mental workload, but also simulates the driving performance, thereby reflecting the workload from both subjective and performance-based measurements (Changxu & Yili, 2007). (Step 5)
- **Using metrics**
  Some metrics should be developed and used effectively regarding the complexity of meetings and the evaluation of cognitive activities. These criteria can be created using experience as well

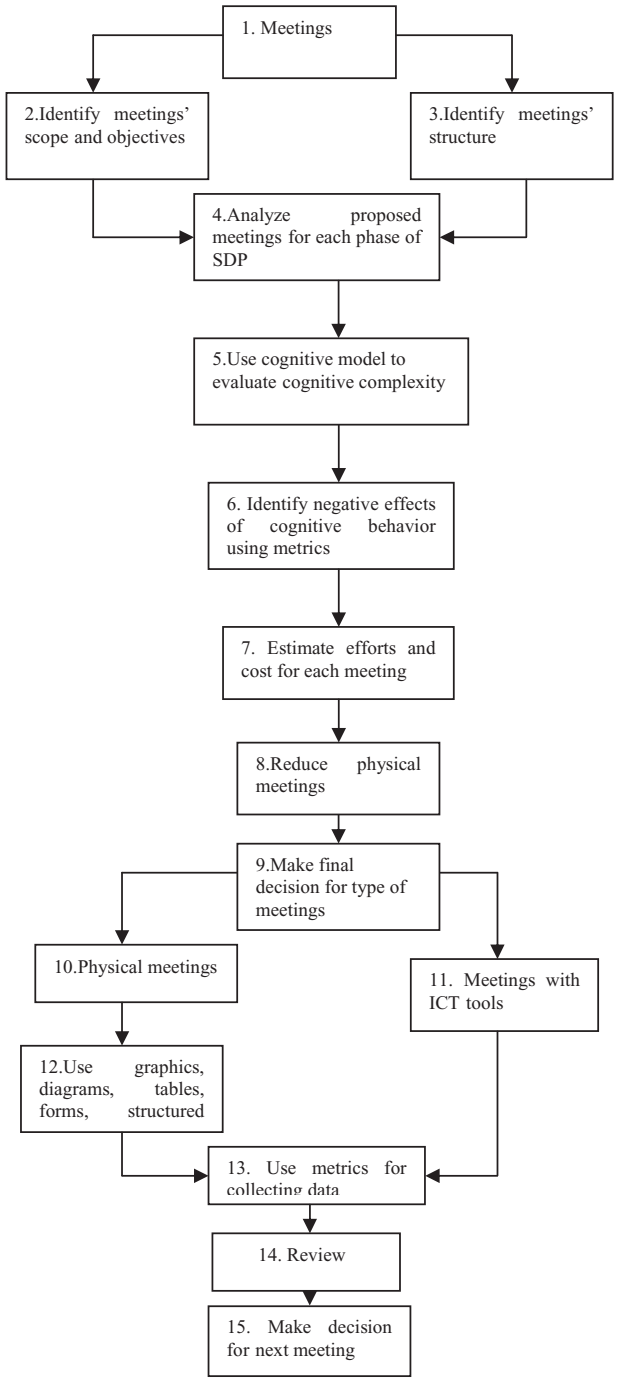

Figure 2 A proposed model for meetings in the SDP.

as the metrics database available from the past projects undertaken by the software company. A cognitive complexity measure can be used to decide if it falls beyond an acceptable level for a meeting. If this is the case, then the issue to be covered in the meeting should be assessed for decomposability (Klemola & Rilling, 2003). (Step 6)

- **Estimating meeting costs and efforts**
  By analyzing the corresponding metrics from previous projects, costs and other spending can be figured for the proposed meetings. If such expenditures are to be higher than anticipated, then physical meetings can be replaced by other means of communications using ICTs. (Steps 7 and 8)
- **Deciding on the type of meetings**
  Based on the steps just listed – that is, using different means such as past experience, cognitive complexity, and estimation models to analyze and estimate the costs, efforts, and applicability of meetings – the final decision should be made regarding the exact nature of such meetings. In the requirements and the analysis phases, physical meetings are necessary and the management of planning and coordination in them should be at an optimal level. This can be achieved by assigning experienced practitioners to critical meetings. (Steps 9 and 10)
- **Considering ICT**
  The use of ICTs such as phone conversations, e-mails and chatting, video-conferencing, voice chatting, blogs, and Internet facilities can help to reduce negative cognitive effects. Thus, if planned and coordinated effectively and systematically, they can be good alternatives to physical meetings. (Step 11)
- **Developing tools**
  Graphics, diagrams (Hungerford, Alan, & Collins, 2004), tables, forms, structured documents, and so forth, can also be useful to reduce the negative effect of cognitive behavior, because they help to increase cognitive synchronization and may be used to avoid (or limit) unwanted interferences. In contrast, such tools, better if computerized, can be developed using the past experiences of the company. An appropriate option here is software already available in the market. (Step 12)
- **Collecting data**
  In both types of meetings, metrics should be used to collect the data. These data will be useful to review meeting outcomes and to make decisions for the following ones. (Steps 13, 14, and 15)

The reader should note here that there is no one-to-one correspondence between the subtopics of the functions just mentioned and the steps in Figure 2 because the functions and steps in Figure 2 have been provided in an abstract form for the purpose of flexibility in their practical acceptance and applicability.

5. EMPIRICAL VALIDATION OF ACMM AND CEMM

The proposed ACMM and CEMM are validated using an empirical approach, and the variables of the study were selected accordingly, leading to an investigation of the relationships between meetings in the development process and the main cognitive factors included in the proposed model. A survey approach was adopted for this purpose, and the data were obtained through a questionnaire prepared in Turkish and English. The respondents were senior IT professionals/managers from major government and business sector organizations participating in a one-day seminar organized by a leading international IT company. The respondents participated in the study voluntarily. The questionnaires were delivered at the registration desk, and a total of 69 completed survey questionnaires were received at the end of the day. The survey instrument contains five main questions corresponding to both dependent (y) and independent (x_i, $i = 1, 2, 3$) variables, and the inquiring data as follows:

I) What is the effect of meetings in a SDP life cycle (5 = very much – 1 = very little) (y)?
II) What is the impact of existence of a cognitive model at meetings in a software development life cycle (5 = very much – 1 = very little) (x_1)?
III) What is the impact of the type of meetings in a software development life cycle (5 = very much – 1 = very little) (x_2)?
IV) What is the impact of cognitive behavior of team members in the meetings in a software development life cycle (5 = very much – 1 = very little) (x_3)?
V) What is the impact of deciding in advance on the purpose, scope, and type of meetings in an SDP (5 = very much – 1 = very little) (x_4)?

The variable x_4 (question V) is used for descriptive purposes in the discussions. The Likert Scale is one of the most effective tools in collecting data in survey-type studies. It is an ordered, one-dimensional scale from which respondents choose one option that best aligns with their view. In this study, the data were collected using a five-point.

Likert Scale (5 = very much, 4 = much, 3 = moderate, 2 = little, 1 = very little) for each item in the empirical categories just mentioned.

The significance concerning the extent and direction of the main effects of the independent variables on the dependent ones was determined using univariate regression analysis. We treated the problem as linear, namely for a given independent variable (x_i, $i = 1, 2, 3$). The mean distribution of the dependent variable y is given by

$$Y_j = a_i + \underline{b}_i \underline{x}_i, \quad i = 1, 2, \ldots, 4.$$

The results of the standardized univariate regression model have been summarized in Table 1.

The results concerning the main effects are presented as standardized regression coefficients in Table 1. The examination of p values in the last column of Table 1 indicates that the existence of "cognitive model," "cognitive behavior of team members," and "type of meetings" have a significantly predictive effect on SDP meetings at the level of $p < .05$. In other words, all of the main factors regarding cognitive aspects in the proposed cognitive abstract model are statistically and significantly related with the effect of meetings in SDP.

TABLE 1. Univariate Regression Results

Dependent Variable	Independent Variable	Coeff.	p Value*
Effect of meetings (y)			
	Existence of cognitive model (x_1)	0.318	.023*
	Type of meetings (x_2)	0.308	.003*
	Team members' cognitive behavior (x_3)	0.173	.045*

*Indicates statistically significant at 5% significance level.

6. DISCUSSION

Initially, this study proposes an ACMM to be applied in the SDP (Figure 1). The model considers cognitive aspects because meetings are affected by cognitive activities (Misra & Akman, 2009; Sanderson & Fisher, 1994). The variables "type of meetings," "team members' cognitive behavior," and "existence of a cognitive model" represent the main factors in the model. The justification of the proposed model is based on survey data from software professionals and findings available from the literature. The empirical results (Table 1) validate the significance of the variable "type of meetings" (coeff. = 0.308, $p = .003$), the cognitive aspects as in the variable "team members' cognitive behavior" (coeff. = 0.173, $p = .045$), and "existence of a cognitive model" (coeff. = 0.318, $p = .023$) for ACMM. The empirical study does not take into account, however, the differences among the phases because this task falls outside of the scope of the present work.

The empirical results can also be used for the explanation and validation of a CEMM. According to the analysis, it has been observed that meetings are important in the SDP because most of the respondents (73.3%) stated the impact to be high. Interestingly, all of the respondents (100%) agree that determining the purpose, scope, and the structure of meetings in advance is equally important for meetings to positively contribute to the SDP. The above sentence is also supported by Robert et al. (1991), who suggested that the structure of team meetings should be decided at the beginning of the development life cycle. In contrast, the contribution of meetings depends on the decisions made at the meetings (Berntsson-Svensson & Aurum, 2006), although the literature is not conclusive regarding the usefulness of team meetings (Johnson & Tjahjono, 1998; Laitenberger & Debaud, 2000; Porter & Johnson, 1997). It has been argued that team meetings may reduce the speed of the SDP, hence increasing the development time depending on the circumstances. These discussions may be the evidence of the fact that decisions made through meetings need to be analyzed carefully depending on their purpose, scope, and structure. Against this backdrop, we propose Steps 1–4 in CEMM (see Figure 2).

There are different factors affecting the overall outcomes of SDPs, the decisions made in meetings being among these factors. According to Bandura (1997), cognitive processes and complexities play an important role in decision making; Wang and Shao (2003) as well have proposed the use of metrics based on cognitive weights. Interestingly, our empirical results (Table 1) show that the existence of a cognitive model has a significant impact on the quality of meetings. Additionally, physical meetings in the SDP tend to involve team members and are governed by group leaders. Bandura (1997) explains how cognitive processes affect the project teams and the leadership style. Weiss and Wysocki (1992) also claim that cognitive aspects play a central role in achieving successful project management. Cognitive modeling should simulate the driving performance, hence the use of performance-based measurements (Macdonald, Miller, Brooks, Roper, & Wood, 1995). The results of this research (Table 1) also highlight the significant impact of cognitive behavior among team members (coeff. = 0.173, $p = .045$) at meetings throughout the SDP. All these can be considered as the justifications for Steps 5, 6, and 7 in the proposed model (Figure 2).

Meetings are intended to be planned, scheduled, and conducted with certain objectives in mind, and the type of meetings should depend on the situation and circumstances (Johansen, 1991). Although there are several variations regarding meetings, they can be classified into two types: physical style and conference style (Ronald, 1996). In the first case, the attendees are physically together at the same location and time, whereas the conference style has become common only since the proliferation of ICT around the world. In fact, in the past, team members worked in a closer physical environment, making it easier to conduct physical meetings among professionals. Over the past decade, the concept and practice of having dispersed or virtual teams have emerged (Hughes & Mike, 2006). The empirical study carried out in this research (Table 1) indicates that the type of meetings has a statistically significant predictive effect on the performance of meetings (coeff. = 0.308, $p = .003$). Additionally, the correlation between the type of meetings and the effect of meetings has been found to be significant ($p = .009$). With this in perspective, we propose Steps 8–11 in CEMM (Figure 2). The reader should note here that, although there are several advantages in using "off-shore" staff regarding software development – benefits that include cheap labor from the developing countries, reduction of overhead arising from having one's own employee on-site (cost of accommodation, social security, training, etc.), and full utilization of time due to different time zones (Hughes & Mike, 2006) – coordination among software professionals across the globe is yet a

major challenge rising against producing a high-quality product.

As also stated by Robillard (1999), meetings are technical tasks carried out by individuals using cognitive activities, and the use of graphics, diagrams, tables, forms, structured documents, and so forth, can be helpful in reducing the negative effect of cognitive behavior because such tools increase cognitive synchronization (Lanubile, Mallardo, & Calefato, 2004). This statement is also supported by our finding (Table 1) that the existence of a cognitive model has a significant impact on the effectiveness of meetings (coeff. $= 0.318$, $p = .023$). For this reason, we have proposed Step 12 (Figure 2). In contrast, one may find numerous metrics in the literature available on SDPs that are either related to the process itself or associated with the software products and, as a result, require an extensive use of data (Sommerville, 2006), hence making the use of metrics for data collection a vital task. Besides, the reviewing stage has been established as one of the most cost-effective quality assurance techniques in improving the development process, and it should involve decisions for further steps (Fagan, 1986). These statements support Steps 13–15 in the proposed model (see Figure 2).

The empirical study in this article provides additional interesting inferences. Experiments have revealed how cognitive factors impact SDP. More specifically, Table 1 indicates that the coefficients of independent variables are all positive. This finding means that the type of meetings and the cognitive factors in the analysis have a significant positive impact on the level of contribution of meetings in SDP.

Although the empirical approach has the advantage of reflecting experiences from the practical aspect of the analysis, it may suffer from subjectivity as a consequence of being confined only to the factors included in the analysis and ignoring the mutual interactions. In this case, the direct – or "live" – survey approach can be used as an appropriate replacement.

The evaluations show that meetings are important and can be held at any time starting from the requirement phase to the end of the SDP (Fagan, 1986). Although face-to-face meetings are generally accepted to be more effective, it is likely that different methods of communication may be preferred at different phases of a project (Bob and Cotterell, 2006). The observations also reveal that cognitive factors are important in all phases within the SDP's life cycle. In this respect, the authors expect that the proposed models ACMM and CEMM to be valuable in guiding software development practitioners as they initiate effective strategies and tactics for development projects. Readers here should note, however, that the contribution of meetings also depends on the requirement for meetings and their circumstances at different phases of the SDP.

This perspective calls for additional studies regarding the contribution of different types of meetings at different phases of the SDP, rendering value to future work. Also, the type and size of the project prove to be among the decisive factors in the SDP, and are related with the cognitive aspects involved in the process. Whereas cognitive science is simply defined as "the interdisciplinary study of mind and intelligence which examines how information concerning perception, language, reasoning, and emotion, is represented and transformed in a brain or machine (e.g., a computer)" (Medicine.net.com, 2011), in terms of future research, the analysis of additional cognitive factors such as perceived usefulness, perceived ease of use, and behavioral intentions may have an important role in the development and justification of extensions to the proposed model. Finally, studying the various dimensions of different levels of awareness concerning the cognitive aspects may provide valuable insights as to the attitude of professionals in meetings.

7. CONCLUSIONS

Software development is a complex and often difficult process, significantly affected by meetings organized with different structures. The article at hand has proposed an abstract model for decisions to set up meetings and a model for the evaluation of meeting decisions in the SDP. The models set forth involve cognitive factors and their evaluation while considering the fact that the results of meetings depend on collective cognitive activities of the individuals involved.

Both of the proposed models were tested through an empirical investigation using appropriate common variables. The empirical results for ACMM validate the significant impact of the cognitive characteristics possessed by the stakeholders (all those involved, including the clients and the development team) at SDP meetings. Similarly, the empirical results and findings from the available literature support all the steps used in the proposed CEMM, leading to the conclusion that the proposed models can be beneficial and productive if used systematically for meetings throughout the

various stages of SDP. In other words, we can conclude that the decisions concerning meetings have to be made with due consideration because there are risks involved in the process, such as excess time expenditure leading to speed reduction. For these reasons, the proposed models serve as an appropriate guide in deciding whether to hold a physical meeting or not. We hope that the present work has the potential to assist specialists in the software industry to not only maximize efficiency and time expenditure by eliminating unnecessary meetings, but also to achieve the high-standard, quality objectives of any information system development in a smoother and more organized and systematic way.

Finally, the management of software development organizations (small- and medium-scale establishments in particular), or those of the units within such organizations, need to pay closer attention to the workforce and their needs for professional development, and to encourage them in this respect. With this purpose in mind, and by means of orientation and training programs, firms can increase overall awareness among their staff of the cognitive factors involved in interpersonal interactions, the various degrees of contributions made by the different types of meetings under different circumstances, and of the cognitive roles and aspects of such meetings in the SDP. We hope that our results can provide a basis for further discussion regarding the promotion of the role that human mind and cognition play at meetings along with the development of effective strategies for organizing more productive meetings in the SDP.

ACKNOWLEDGMENTS

We wish to express our appreciation to the reviewers and the editor for their valuable comments and input on this article, and also to Basak Akman, an instructor of English at the School of Foreign Languages at the Middle East Technical University, and Payam "Paul" Danesh and Ismail Erton, of the Academic Writing and Advisory Center, Atilim University, for their assistance in proofreading the manuscript.

References

Akman, I., Misra, S., & Altindag, T. (2011). The impact of cognitive and socio-demographic factors at meetings during software development process. Technical Gazette, 18(1), 51–56.

Akman, I., Misra, S., & Cafer, F. (2011). The role of leadership cognitive complexity in software development projects: An empirical assessment. Human Factors and Ergonomics in Manufacturing & Service Industries, 21(5), 516–515.

Bandura, A. (1997). Self-efficacy: The exercise of control. New York: W. H. Freeman/Times Books/Henry Holt & Co.

Berntsson-Svensson, R., & Aurum, A. (2006). Successful software project and products: An empirical investigation. In Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (pp. 144–153). Rio de Janeiro, Brazil, during 21–22 September 2006.

Bob, H., & Cotterell, M. (2006). Software project management (4th ed.). New York: McGraw-Hill.

Bray, I. K. (2002). An introduction to requirements engineering (p. 7). Boston: Addison-Wesley.

Changxu, W., & Yili, L. (2007). Queuing network modeling of driver workload and performance. IEEE Transactions on Intelligent Transportation Systems, 8(3), 528–537.

Daniel, G. (2002). Software quality assurance, from theory to implementation (pp. 77–77). Boston: Addison-Wesley.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(3), 182–211.Fagan, M. E. (1986). Advances in software inspections. IEEE Transaction on Software Engineering, 12(7), 744–751.

Gilb, T. (2007). No cure no pay: How to contract for software services. Journal of Computer and Information Science, 4(1), 29–42.

Gilb, T., & Graham, D. (1993). Software inspection. Harlow, UK: Addison-Wesley.

Hazzan, O. (2005). How does software intangibility influence software development, processes? Featured Column, System Design Frontier Journal,1—3. Retrieved July 2010 from http://edu.technion.ac.il/faculty/orith/homepage/FrontierColumns/OritHazzan_SystemDesignFrontier_Column1.pdf

Hazzan, O. (2006). Agile software development and the nature of software dvelopment. Featured Column, System Design Frontier Journal, 1–5. Retrieved July 2010 from http://edu.technion.ac.il/faculty/orith/homepage/FrontierColumns/OritHazzan_SystemDesigFrontier_Column6.pdf

Hughes, B., & Mike, C. (2006). Software project management (4th ed., pp. 251–252). New York: McGraw-Hill.

Hungerford, B. C., Alan, R. H., & Collins, R. W. (2004). Reviewing software diagrams: A cognitive study. IEEE Transaction on Software Engineering, 30(2), 82–96.

Hunt, A., Thomas, D., & Hargett, M. (2008). Pragmatic unit test in C# with NUnit. ISBN-13: 978-0-9776166-7-4, The Pragmatic Bookself.

Johansen, R. (1991). Leading business teams. Boston: Addison-Wesley.

Johnson, P. M., & Tjahjono, D. (1998). Does every inspection really need a meeting? Empirical Software Engineering, 3, 3–35.

Klemola, T., & Rilling, J. (2003). A cognitive complexity metric based on category learning (pp. 103–108). In Proceedings of the Second IEEE International Conference on Cognitive Informatics, 18–20 August, London, UK.

Laitenberger, O., & Debaud, J. M. (2000). An encompassing life cycle centric survey of software inspection. The Journal of Software and Systems, 50(1), 5–31.

Lanubile, F., Mallardo, T., & Calefato, F. (2004). Tool support for geographically dispersed inspection teams. Software Process Improvement and Practice, 3(1), 3–21.

Leffingwell, D., & Don, W. (2003). Managing software requirements (2nd ed.). Boston: Addison-Wesley Professional.

Macdonald, F., Miller, J., Brooks, A., Roper, M., & Wood, M. (1995). A review of tool support for software inspections (pp. 340–349). Proceedings of the Seventh International Workshop on Computer-Aided Software Engineering (CASE-95). July 10–14, Toronto, CANADA.

Medicine.net.com. (2011). Definition of cognitive science. Retrieved March 2011 from http://www.medterms.com/script/main/art.asp?articlekey=9257

Misra, S., & Akman, I. (2009). A cognitive evaluation for meetings in software development process (pp. 247–254). In Proceedings of the 4th International Conference on Rough Sets and KNOWLEDGE TECHNOLOGY, Lecturer Notes in Computer Science, July 14–16, Gold-Coast, Australia.

Porter, A., & Johnson, P. (1997). Assessing software review meetings: Results of a comparative analysis of two experimental studies. IEEE Transactions on Software Engineering, 23(3), 129–145.

Robert, J., Martin, A., & Sibbet, D. (1991). Leading business teams:How teams can use technology and group process tools to enhance performance. Boston: Addison-Wesley.

Robillard, P. N. (1999). The role of knowledge in software. Communications of the ACM, 42(1), 87–92.

Robillard, P. N., Astus, P., Detienne, F., & Visser, W. (1998). Measuring cognitive activities in software engineering (pp. 292–300). In Proceedings of the ICSE98, 20th International Conference on Software Engineering, April 19–25, 1998, Kyoto, JAPAN.

Ronald, J. N. (1996). Object oriented systems analysis and design (1st ed.). Upper Saddle River, NJ: Prentice Hall.

Sanderson, P. M., & Fisher, C. (1994). Exploratory sequential data analysis: Foundations. Human Computer Interaction, 9, 251–317.

Sauer, C., Jeffery, R. D., Land, L. & Yetton, P. (2000). The effectiveness of software development technical reviews: A behaviorally motivated program of research. IEEE Transaction on Software Engineering, 26(1), 1–14.

Sommerville, I. (2006). Software engineering (8th ed.). Boston: Addison Wesley.

Vota, L. G. (1993). Does every inspection need a meeting? ACM Software Engineering, 18(5), 107–114.

Wang, Y. (2009a). Formal description of the cognitive process of memorization. LNCS Transactions on Computational Science V: Special Issue on Cognitive Knowledge Representation, 81–98.

Wang, Y. (2009b). On cognitive properties of human factors in engineering (pp. 174–182). In Proceedings of the Fourth IEEE International Conference on Cognitive Informatics. June 15–17, 2009, Kowloon, Hong Kong.

Wang, Y., & Shao, J. (2003). A new measure of software complexity based on cognitive weights. Canadian Journal of Electrical & Computer Engineering, 28(2), 69–74.

Wang, Y. K. (2006). Modern software review: Techniques and technologies. UK: IRM Press. London, UK.

Weiss, J. W., & Wysocki, R. K. (1992). 5-Phase project management: A practical planning and implementation guide. Cambridge, MA: Perseus Books.