Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 178 (2017) 213 – 222

Procedia Engineering

www.elsevier.com/locate/procedia

16th Conference on Reliability and Statistics in Transportation and Communication, RelStat'2016, 19-22 October, 2016, Riga, Latvia

The Role of Communication and Meta-Communication in Software Engineering with Relation to Human Errors

Boriss Misnevs[a]*, Ugur Demiray[b]

[a]*Transport and telecommunication Institute, 1 Lomosova str.,Riga, LV-1019, Latvia*
[b]*Anadolu University, Yunus Emre Campus, Eskisehir,26470, Turkey*

Abstract

This paper examines and focuses on some issues and questions relating to how the use meta-communication concept in Software Engineering process to reduce human errors. The role of IT project communication and the project management tools, which can be regarded as vital for Software Engineering are investigated. Socio-cognitive modeling of Integrated Software Engineering using the TOGA meta-theory, has been discussed. Today the focus is especially on the identification of human and organization decisional errors caused by software developers and managers under high-risk conditions, as evident by analyzing reports on failed IT projects. Software Engineer's communication skills are listed. Several types of initial communication situations in decision-making useful for the diagnosis of Software developers' errors are considered. The developed models can be used for training the IT project management executive staff.

Peer-review under responsibility of the scientific committee of the International Conference on Reliability and Statistics in Transportation and Communication

Keywords: defect prevention, Socio-cognitive modeling, IT project processes, TOGA meta-theory

1. Introduction

The current research was based on the assumption that human error is an important cause of software defects. Communication and meta-communication studies were used to develop a deeper understanding of the human errors

* Corresponding author
E-mail address: bfm@tsi.lv

Peer-review under responsibility of the scientific committee of the International Conference on Reliability and Statistics in Transportation and Communication
doi:10.1016/j.proeng.2017.01.100

that occur during the software development process and help IT practitioners to detect and prevent those errors early in the software development lifecycle.

Early elimination of mistakes will improve software quality and reduce overall development cost. The target of the research was to develop an approach and model useful for miscommunication reducing in IT project activities. The research is related to the Software Engineering Master Program graduates competence evaluation project (Misnevs and Yatskiv, 2016) with the emphases on master students' communication skills.

The main attention was paid to understanding of meta-communication role in IT Project communication processes. The term "meta-communication" was suggested by Bateson in 1951, and then he elaborated in 1956 a critical fact that every message could have a meta-communicative element, and typically, each message held meta-communicative information about how to interpret other messages. He saw no distinction in type of message, only a distinction in function (Bateson, 1972).

The prefix can have various meanings but as used in communication, philosophy and psychology its meaning is best recognized as about. Thus, Meta-communication is communication about communication; meta-language is language about language; meta-message is a message about a message. You can communicate about the world - about the human errors in the software engineering process, the computer you are using, or the text you're reading right now. We refer to this as object communication; because you are talking about objects. And the language you are using is called an object language. But notice that you are not limited to talking about objects; you can also talk about your talk; you can communicate about your communication. And this is referred to as meta-communication. In the same way, you can use language (i.e., meta-language) to talk about language (i.e., object language). And you can talk about your messages with meta-messages.

Meta-communication mostly is the nonverbal cues (tone of voice, body language, gestures, facial expression, etc.) that carry meaning that either enhance or disallow what we say in words.

The distinction between object communication and meta-communication is not merely academic; it's extremely practical, and it is recognized that the difference between these two forms of communication is essential in untangling lots of conflicts and understanding a wide variety of interpersonal communication interactions. Actually, we use this distinction (as a meta-communication) every day, mostly without realizing it. For example, when you send someone an e-mail with a seemingly sarcastic comment and then put a smiley at the end, the smiley communicates about your communication; it says something like "this message is not to be taken literally; I'm trying to be humorous." The smiley is a meta-message; it's a message about a message. When you say, in preface to some comment, "I'm not sure about this but...." you're communicating a message about a message; you're commenting on the message and asking that it be understood with the qualification that you may be wrong. When you conclude a comment with "I'm only kidding" you're meta-communicating; you're communicating about the communication.

Frits Staal (2010) related the term to meta-language concept that is found in logic both in Western and Indian traditions. Staal considered the term meta-language, or its German or Polish equivalent, to have been introduced in 1933 by the logician Alfred Tarski.

In this research we discuss our ideas on the software quality improvement by the integration of the human factors engineering into the development process using Socio-cognitive Engineering methods.

2. Literature Review

There are many studies of human factors, however most of them are solely oriented on human-machine operations in terms of system and program usability, but not in terms of software engineering process (Spichkova *et al.*, 2015) or they are dedicated to so called Engineering Error Paradigm (Redmill and Rajan, 1997). By this paradigm humans are seen as they are almost equivalent to software and hardware components in the sense of operation with data and other components, but at the same time humans are seen as the "most unreliable component" of the total system.

Meta-communication studies in Computer Science mostly are related to Human Computer Interaction (HCI) and Semiotic Engineering. Semiotic perspectives on HCI take human-computer interaction as a special case of computer-mediated human communication.

Through the interface, systems designers communicate to users their design vision as well as how the system can or should be used for a variety of purposes. To date, there hasn't been enough empirical research in HCI exploring this complex phenomenon.

The paper "Meta-communication and Semiotic Engineering: Insights from a Study with Mediated HCI" reports an empirical research about meta-communication in HCI and discusses how and why semiotically - inspired research can contribute to advance knowledge in this field (Teixeira Monteiro, 2013). Another area related to meta-communication is values and culture in interactive systems design. Depending on the way technologist designed, it will afford behaviors that are intrinsically related to individuals and the complex cultural context in which they are using it (Neakrase *et al.*, 2011).

Individuals will interpret and behave through the technology influenced by the cultural systems (e.g., values, beliefs, behavioral patterns).Their behavior may be in disagreement or agreement with their values and the values of other people. This, in turn, will promote or inhibit certain values over others. The meta-communication research in Software Engineering is also related to the integration of architectures, protocols, and systems.

It is argued that meta-communication, i.e. communication about communication rules, is a general integration methodology that is applicable to the integration of architectures, protocols, and systems. Efforts towards the development of an automated methodology for meta-communication are discussed. The authors view meta-communication as a design problem. Meta-communicating entities exchange partially specified communication rules. Each entity, or a meta-communication center, applies a standard composition principle on the individual partially specified rules in order to derive the complete protocol architecture (Meandzija, 1990). Some authors study cultural values in Software Engineering as meta-communication entities (Pereira *et al.*, 2011; Pereira and Baranauskas, 2015). Value-oriented and Culturally Informed Approach (VCIA) to sensitize and support Computer Science and Engineering professionals in taking values and culture into consideration throughout the design of interactive systems.

The reflective practice becomes more important the more the differences in technologic standards, social values, norms, assumptions and interests, etc. in global contexts interfere the sphere of the Information Systems Development (ISD).

The paper (Yetim, 2004) extended the framework for reflective practice proposed by Ulrich (2001). Three different types of meta-communication are described:

- Ex ante meta-communication (taking place before action),
- Meta-communication in action (taking place during action), and
- Ex post meta-communication (taking place after action).

The meta-communication model itself consists of two levels:

- Clarification level (where conversation for clarification takes place). At this level there are eleven clarification issues to be reflected on.
- Discourse level (where the discursive examination of contested claims takes place). At this level, there are eight discourses, which are related to the clarification issues.

In Bateson's work (1972), meta-message was defined as a refinement of his earlier notion of "mood sign[al]"s from his works of the 1950s. Invoking Bertrand Russell's Theory of Logical Types, Bateson envisaged a potentially infinite hierarchy of messages, meta-messages, meta-meta-messages and so forth, each meta-message deterministically providing the full context for the interpretation of subordinate messages. Being rather technical, his definition was misunderstood, and meta-message appropriated with the same meaning as subtext, especially in the field business communications.

The issue is that people don't understand each other's code. Verbal communication is supported by a raft of non-verbal signs and cues that reinforce what we are saying or clear up any ambiguities. For example, we may cross our arms when we feel threatened by what somebody else is saying, or we nod our heads when we agree with what they are saying.

Although nonverbal communication gives clues to what speakers are thinking about or enhances what they are saying, cultural differences may also interfere with understanding a message (Pennycook, 1985). The rules are brought to our attention only in formal discussions of nonverbal communication, such as this one, or when rules are violated and the violations are called to our attention-either directly by some tactless snob or indirectly through the examples of others.

It must be mentioned that nonverbal behavior is highly believable. For some reasons we are quick to believe nonverbal behaviors even when these behaviors contradict verbal messages. Nonverbal reports on research demonstrating that compared to verbal cues, nonverbal cues are four times as effective in their impact on interpersonal impressions and ten times more important in expressing confidence. From a different perspective, Albert Mehrabian (1976) argues that the total impact of a message is a function of the following formula is: total impact = 7% verbal + non-verbal 38% + 55% facial.

It is essential to remember that the meta-communication which accompanies any message is very powerful. The receiver will use these clues to help them to interpret what you mean, but more importantly they will often take the meaning from the meta-communication rather than from the words themselves, particularly when what you are saying conflicts with what you are doing.

This may be particularly useful when the opportunity for face-to-face meta-communication is missing, as in distance teaching (McLean, 1999) or as in geographically distributed IT project teams.

Another example deals with etiquettes. Etiquettes are practicing in good manners or to know how to behave in given situation and to know how to interact with the people or others. Proper etiquette helps you make a great first impression and stand out in a competitive with others. From point of communication science, etiquettes have meta communicable function in communication process.

Some requirements from actual e-mail etiquette (E-Mail Etiquette, 2014) are mentioned below.

Write carefully. Once you send an email message, you cannot take it back or make it disappear. The reality is that your messages may be saved for a very long time. They may also be read by others, or forwarded to others without your knowledge.

Sign your messages with at least your name. It's nice to add your email address; too, since some email programs make it difficult to see who the sender of the message was.

Indicate humor or jokes with a smiley face. :-).

Be calm. You may have misunderstood what was meant. Don't reply while you're still angry (this is called "flaming").

Don't forward emails unless you have the permission of the author. What they wrote may not have been intended for wider distribution, so it's always better to ask.

Do let people know their letter was received. Try to talk about one subject per message only. For another subject, start a new email.

3. Science language is perfect sample for meta communication in Software Engineering

Since has its own language. It tells or passes us information and data by showing and serving some code, figures, charts and graphics also etc. It accepts that if we know these codes than lean to us concepts, thoughts and idea.

Discourse in the science classroom for Software Engineering is framed under situated cognition theory, whereby interactions between individuals are part of the normal culture of the classroom. For Software Engineering knowledge to be adequately constructed by a student, these interactions must be meaningful ones. This is especially important in an online course where typically learning occurs through interactions between the students and the instructor, the students with one another, and within the individual themselves. As part of these online interactions, good reflective practice includes the different forms of feedback and the quality of this feedback. However, even with quality reflective interactions, there are barriers to Computer Science concept construction in an online environment. These barriers are discussed, and future research directions are suggested based on this review.

It is clear that the environment for learning Software Engineering is not limited to the face-to-face classroom, but can be other environments such as online or informal Education environments. How these characteristics of science inquiry look in practice in both the face-to-face and online classrooms has been discussed elsewhere by the authors (Baptiste *et al.*, 2011). Software processes are specified for a number of reasons: to facilitate human understanding,

communication, and coordination; to aid management of software projects; to measure and improve the quality of software products in an efficient manner; to support process improvement; and to provide a basis for automated support of process execution. Enable Effective Communications: modelling employs the application domain vocabulary of the software, a modelling language, and semantic expression (in other words, meaning within context). When used rigorously and systematically, this modelling results in a reporting approach that facilitates effective communication of software information to project stakeholders. Management sponsorship supports development process, product evaluations and the resulting quality findings. Then an improvement program is developed identifying detailed actions and improvement projects to be addressed in a feasible time frame. Management support implies that each improvement project has enough resources to achieve the goal defined for it. Management sponsorship is solicited frequently by implementing proactive communication activities. Different types of reviews and audits are distinguished by their purpose, levels of independence, tools and techniques, roles, and by the subject of the activity. Success of a software engineering endeavor depends upon positive interactions with stakeholders. They should provide support, information, and feedback at all stages of the software life cycle process. Therefore, it is vital to maintain open and productive communication with stakeholders for the duration of the software product's lifetime (SWEBOK 3.0, 2014).

Multicultural environments can have an impact on the dynamics of a group. This is especially true when the group is geographically separated or communication is infrequent, since such separation elevates the importance of each contact. Intercultural communication is even more difficult if the difference in time zones make oral communication less frequent.

More frequent communication, including face-to-face meetings, can help to mitigate geographical and cultural divisions, promote cohesiveness, and raise productivity. Also, being able to communicate with teammates in their native language could be very beneficial. It is vital that a software engineer communicate well, both orally and in reading and writing. Successful attainment of software requirements and deadlines depends on developing clear understanding between the software engineer and customers, supervisors, co-workers, and suppliers. Let's have look deeper to examples from the math course world.

If formula of square' area indicates or shows, it means computing of square' area in any language, even change of the length of the sides does not chance way of computing of square' area. Only numbers change and formula stay in the same body. When we sow formula of square' area, we think and animate in our mind that square' area is equal to one side's square. These formulas are bringing a picture to our mind as automatically (see Fig.1).

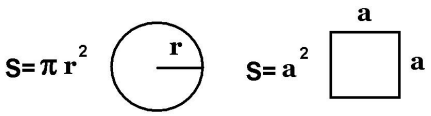

Fig. 1. Pictures for circle and area of square.

We still remember these formulas as certain concept in picture form. It is just like traffic signs. Some formulas are important for life science have the same importance for our daily life, so we do not forget them any time. We use them automatically such as reflex. These are examples from the certain life science which learned at certain education level in our education life. Some graphs are tell us very briefly what is happening in the diagram on some increasing success, increasing producing or decreasing success, decreasing producing etc.

On graphing functions, with examples, try to give detailed info and matched mentioned subjects. The properties of the graphs of linear, quadratic, rational, trigonometric, absolute value, logarithmic, exponential and piecewise functions are analyzed in details. Detailed info and explanations to the examples are included. As seen in these examples we do not need to talk or tell much. Concepts such as asymptotes or colors for graphs of rational, logarithmic and exponential functions are explored numerically. It gives the main idea in general info at initial seeming. They help us to tell very complex results in basic and brief explanation. Asymptotes, colors, legends and charts have their own meanings which are decode in our mind immediately. This decoding tells us correlations and differentiations with each other.

Good practice for Software Engineering is UML usage (UML – Overview, 2016) as a meta-language. UML is a standard language for specifying, visualizing, constructing, and documenting the artefacts of software systems. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

A picture is worth a thousand words, this absolutely fits while discussing about UML.

There are a number of goals for developing UML but the most important is to define some general purpose modelling language which all modelers can use and also it needs to be made simple to understand and use.

UML is popular for its diagrammatic notations. For example, UML class is represented by the diagram divided into four parts:

- The top section is used to name the class.
- The second one is used to show the attributes of the class.
- The third section is used to describe the operations performed by the class.
- The fourth section is optional to show any additional components.

The UML infrastructure is used to provide a reusable meta-language core. This is used to define UML itself.

4. Communication skills for Software Engineer

Optimal problem solving is made possible through the ability to investigate, comprehend, and summarize information. Customer product acceptance and safe product usage depend on the provision of relevant training and documentation. It follows that the software engineer's own career success is affected by the ability to consistently provide oral and written communication effectively and on time.

Software engineers are able to read and understand technical material. Technical material includes reference books, manuals, research papers, and program source code. Reading is not only a primary way of improving skills, but also a way of gathering information necessary for the completion of engineering goals. A software engineer sifts through accumulated information, filtering out the pieces that will be most helpful. Customers may request that a software engineer summarize the results of such information gathering for them, simplifying or explaining it so that they may make the final choice between competing solutions.

Reading and comprehending source code is also a component of information gathering and problem solving. When modifying, extending, or rewriting software, it is critical to understand both its implementation directly derived from the presented code and its design, which must often be inferred.

Software engineers are able to produce written products as required by customer requests or generally accepted practice. These written products may include source code, software project plans, software requirement documents, risk analyses, software design documents, software test plans, user manuals, technical reports and evaluations, justifications, diagrams and charts, and so forth. Writing clearly and concisely is very important because often it is the primary method of communication among relevant parties. In all cases, written software engineering products must be written so that they are accessible, understandable and relevant for their intended audience(s).

Software engineers rely on their presentation skills during software life cycle processes. For example, during the software requirements phase, software engineers may walk customers and teammates through software requirements and conduct formal requirements reviews (see Requirement Reviews in the Software Requirements KA). During and after software design, software construction, and software maintenance, software engineers lead reviews, product walkthroughs (see Review and Audits in the Software Quality KA), and training. All of these require the ability to present technical information to groups and solicit ideas or feedback.

The software engineer's ability to convey concepts effectively in a presentation therefore influences product acceptance, management, and customer support; it also influences the ability of stakeholders to comprehend and assist in the product effort. This knowledge needs to be archived in the form of slides, knowledge write-up, technical whitepapers, and any other material utilized for knowledge creation (SWEBOK 3.0, 2014).

Written communication is also extensively used in IT project activities, such as project and product documentation which included operation manuals, check lists data cards etc. It is a one way communication, the

checklist or documents send the information but it is up to the Software Engineer to interpret the message and then take actions based on their understandings.

5. Team and Group Communication

Effective communication among team and group members is essential to a collaborative software engineering effort. Stakeholders must be consulted, decisions must be made, and plans must be generated. The greater the number of team and group members, the greater the need to communicate. The number of communication paths, however, grows quadratic ally with the addition of each team member. Further, team members are unlikely to communicate with anyone perceived to be removed from them by more than two degrees (levels). This problem can be more serious when software engineering endeavors or organizations are spread across national and continental borders. Some communication can be accomplished in writing. Software documentation is a common substitute for direct interaction. Email is another but, although it is useful, it is not always enough; also, if one sends too many messages, it becomes difficult to identify the important information.

One of the fundamental principles of a good requirements elicitation process is that of effective communication between the various stakeholders. This communication continues through the entire Software Development Life Cycle process with different stakeholders at different points in time. Before development begins, requirements specialists may form the conduit for this communication. They must mediate between the domain of the software users (and other stakeholders) and the technical world of the software engineer. A set of internally consistent models at different levels of abstraction facilitate communications between software users/stakeholders and software engineers.

It is typically necessary to validate the quality of the models developed during analysis. For example, in object models, it is useful to perform a static analysis to verify that communication paths exist between objects that, in the stakeholders' domain, exchange data.

If formal analysis notations are used, it is possible to use formal reasoning to prove specification properties. Applying external or internal development standards during construction helps achieve a project's objectives for efficiency, quality, and cost. Standards that directly affect construction issues include communication methods (for example, standards for document formats and contents) (SWEBOK 3.0, 2014).

Communication tools can assist in providing timely and consistent information to relevant stakeholders involved in a project. These tools can include things like email notifications and broadcasts to team members and stakeholders. They also include communication of minutes from regularly scheduled project meetings, daily stand-up meetings, plus charts showing progress, backlogs, and maintenance request resolutions.

6. Methodology of the meta-model presentation

Effective communication among team and group members is essential to a collaborative software engineering effort. Stakeholders must be consulted, decisions must be made, and plans must be generated. The greater the number of team and group members, the greater the need to communicate.

The methodology recommended for the research of the role of communication and meta-communication in software engineering with relation to human errors is a heuristic application of TOGA (Top-down Object based Goal oriented Approach). TOGA is the goal-oriented knowledge ordering (conceptual modelling) tool for the specification and system/process identification of real-world complex problems (Gadomski, 1997).

In such sense, it can be seen as an initial top/generic and axioms-based meta-model, and subsequently, the methodology of problem decomposition and specialization using available knowledge (see Fig.2).

Top-down means: From most general minimal information on a problem to its detailed specification/identification system/process identification. Such approach enables a control/check of the completeness and congruence of system/process identification in every problem specialization step.

It requires an initial sufficient amount of information, knowledge and preferences related to the problem, their subsequent acquisition during the problem system/process identification, and the additional specialization patterns assembled in TOGA as Knowledge Ontology Conceptualization System (KNOCS). KNOCS includes top: meta-modelling axioms, assumptions and model frames.

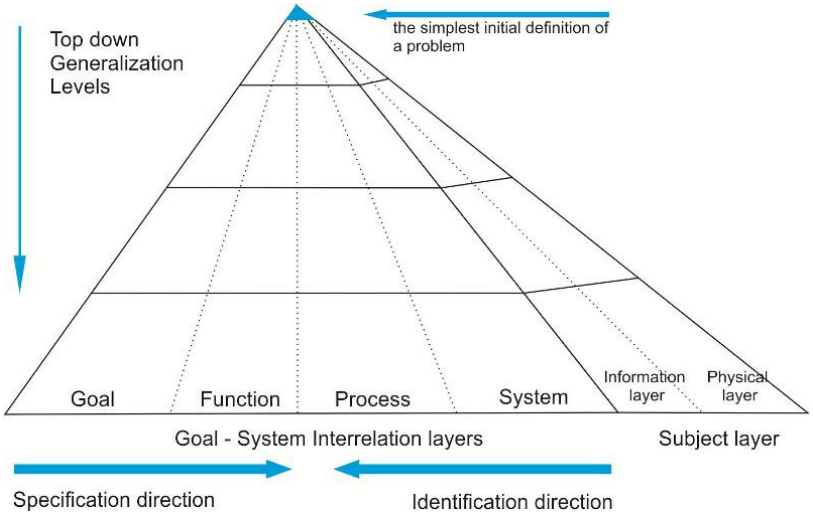

Fig. 2. The methodology of the presentation is a heuristic application of TOGA (Gadomski, 1993).

Object-based indicates a fundamental conceptualization platform of the meta-theory, called the Theory of Abstract Objects (TAO). It is not object-oriented, it assumes that every problem can be represented using the frameworks of world of abstract objects and these worlds universes. The concept of abstract object is defined as everything what can be conceptualized as (<names>, <attributes list>, <values>). Attributes result from objects relations and the change concept. This assumption consists of the primary axiom of the perception of the real world and is considered as the basis of the conscious reasoning of an intelligent entity/agent.

Goal-oriented; it is equivalent to goal-driven, goal-based, goal-directed, teleological and similar approaches, where the methods and methodology applied, in the real-time of the problem solving, confront the attributes of the pre-defined activity/design goal with those which result from candidates on the problem sub-models. The dominating top-goal is defined from the socio-cognitive perspective, and it is always the goal of the human or artificial problem solver, decision-maker or designer.

The goal-oriented and top-down rules of system/process identification are included in the Methodological RUles System (MRUS) - the third TOGA component (Gadomski, 1997). For software engineers, TOGA aims to provide the designer of complex engineering system, an intelligent-agent-based conceptualization with a structured set of methods and rules to allow him to control top-down and goal-oriented conceptual modelling process/activity. It enables to specify formally agent-based systems that can be implemented within an agent-based programming platform.

For such tasks, TOGA also provides a global identification and design methodological framework for human-computer intelligence-based systems. Its level of meta-formalization, top-down and goal-oriented requirements enable together to cope with a symbolic (not a sub-symbolic) design and to develop a general incremental intelligence (an abstract or synthetic intelligence). From the top systemic meta-philosophical perspective, the TOGA computational philosophy is funded on the set of meta-assumptions/meta-axioms leading to the plausible motivations and choices of the TOGA axioms.

Its main reference-point is a subjective perspective of an intelligent; entity, i.e. it assumes that humans acts on the base of always limited available domain-knowledge, therefore every intelligent; agent/entity has his/her/its individual philosophy and it evolves according to their dynamics and different fusions into intelligent aggregates.

Knowledge is found in the minds and bodies of thinking beings (Johnson, 1987). Learning is the construction of knowledge by individuals as sensory data are given meaning in terms of their prior knowledge. It is an interpretive process, involving constructions of individuals and social collaboration (Tobin *et al.*, 1990). Dynamic models of

meta-communication are discussed in the book (Demiray *et al*., 2012). The concepts of the meta-communication model are mainly based on Avatar Manager and Student Reflective Conversations pedagogical theory.

The IT project manager must know the communication processes involved in effective project management. First of all there should be planning to determine what information needs to be communicated to all stakeholders in the project.

Finally, communication with project stakeholders must be managed so that all requirements are met and issues are promptly resolved. Interactions and overlap among the communication processes are inevitable and expected throughout all phases of project management.

This research looks to improve software quality (see Fig.3) in a new way by assuming that human communication error is a key cause of software defects.

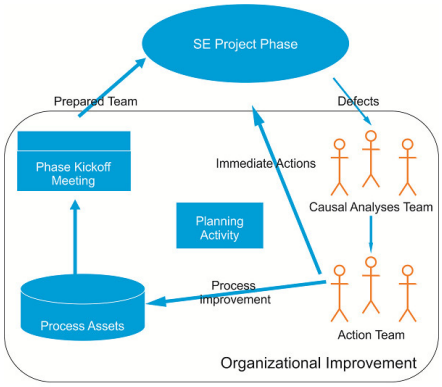

Fig. 3. Defect Prevention Process in Software Engineering.

Research from cognitive psychology is used to develop a deeper understanding of the human errors that occur during the software development process and to develop techniques that detect and prevent those errors early in the software development lifecycle. Early elimination of mistakes will improve software quality and reduce overall development cost. (Carver and Walia, 2014).

Introduction of new intellectual SE tools could completely change the scheme of communication in IT Project. For example, MIT computer scientists (Conner-Simons, 2015) suggested CSAIL's "Helium" tool for bit-rot problem solving. Bit rot is the slow deterioration in the performance and integrity of data stored on storage media.

This new computer program can automatically fix old code so that engineers can focus on more important tasks. CSAIL's "Helium" system revamps and fine-tunes code without ever needing the original source, in a matter of hours or even minutes.

The second case consider the human being (test engineer, developer) to be outside communication process for decision making and bug fix implementation. So we will have two different schemes of communications.

7. Conclusions

Communication is an essential process in the world of IT project management. It is difficult to master, but essential to make a good effort in achieving.

In this paper, we have described a Meta-communication model, which extends the spectrum of earlier discussed approaches to Meta-communication modelling for Software Engineering processes.

Communication in global context remains a challenge and the value-consensus formation nearly impossible in the short run. Suggested model provides a way for systematically and meaningfully structuring and organizing meta-level conversations within IT projects.

Thus, it can be used in several Software Engineering processes, in order to enable effective meta-communication. It is argued that meta-communication, i.e. communication about communication rules, is a general integration methodology that is applicable to the integration of architectures, protocols, and systems.

Efforts towards the development of an automated methodology for meta-communication are discussed. The authors view that meta-communication as a design problem. The developed models can be used for training the IT project management executive staff.

Acknowledgements

The research is part of the project “Implementation of Software Engineering Competence Remote Evaluation for Master Program Graduates (iSECRET)” run by TTI, contract No. 2015-1-LV01-KA203-013439, co-financed by EC ERASMUS+ program.

References

Bateson, G. (1972) Steps to Ecology of Mind: Collected Essays in Anthropology, Psychiatry, Evolution, and Epistemology. University of Chicago Press.

Baptiste, H.P., Neakrase, J.J. and Ryan, A. (2011) Relevant issues that challenge the designing of transformative, liberating on-line science education. In Handbook of research on transformative on-line and liberation: models for social equality, pp. 47–66.

Carver, J.and Walia,G. (2014) Integrating Software Engineering and Human Error Models To Improve Software Quality, 2014. Available at: http://humanerrorinse.org (accessed 24 August 2016).

Demiray, U., Kurubacak, G. and T., Yuzer, V. (2012) Meta-communication for Reflective Online Conversations: Models for Distance Education. Anadolu University, Turkey.

Gadomski, A.M. (1993) Ontology & Knowledge: Meta-Ontological Perspective. Meta-Knowledge Engineering Server. Rome: ENEA, 2002-07. Last updated 2004-07. Available at: http://erg4146.casaccia.enea.it/Ont-know.htm (accessed 24 August 2016).

Gadomski, A.M. (1997) Global TOGA Meta-Theory/ Available at: http://erg4146.casaccia.enea.it/wwwerg26701/Gad-toga.htm (accessed 24 August 2016).

Gibson, E.J. (1977) The performance concept in building. In: Proceedings of the 7th CIB Triennial Congress, Edinburgh, September 1977. London: Construction Research International, pp. 129-136.

E-Mail Etiquette (2014) Available at: http://www.it.cornell.edu/services/guides/email/polite.cfm (accessed 24 August 2016).

Staal, F. (2010) A Theory of Ritual? Available at: https://www.knaw.nl/shared/resources/actueel/bestanden/11102010_Lezing_Frits_Staal_A_theory_of_ritual.pdf (accessed 24 August 2016).

McLean, R.S. (1999) Communication Widgets for Knowledge Building in Distance Education.Computer Support for Collaborative Learning Proceedings of the 1999 conference on Computer support for collaborative learning, 1999, Palo Alto, California December 12–15.

Meandzija, B. (1990) Integration through meta-communication. INFOCOM '90, 9th Annual Joint Conference of the IEEE Computer and Communication Societies.The Multiple Facets of Integration.Proceedings vol.2, IEEE.pp.702-709.

Misnevs, B.and Yatskiv I. (2016) Data Science: Professional Requirements and Competence Evaluation. Baltic Journal of Modern Computing, 4(3), 441–453.

Neakrase J, Prentice Baptiste H., Ryan A. and Villa E., (2011) Science for All through Reflective Interactions: Analyzing Online Instructional Models, Learning Activities and Virtual Resources, In: Meta-Communication for Reflective Online Conversations: Models for Distance Education. Igi Global.

Pennycook, A. (1985) Actions speak louder than words: Paralanguage, communication, and education. TESOL Quarterly, 19, 259–282.

Redmill F. and Rajan J. (1997) Human factors in safety-critical systems.Butterworth-Heinemann.

Spichkova M., Huai Liu, Mohsen Laali, and Heinz W. Schmidt (2015) Human Factors in Software Reliability Engineering. Available at: http://arxiv.org/pdf/1503.03584v1.pdf (accessed 24 August 2016).

SWEBOK 3.0 (2014) Guide to the Software Engineering Body of Knowledge Version 3.0, IEEE. Available at: http://www.swebok.org (accessed 24 August 2016).

Tobin, K., Briscoe, C.and Holman, J.R. (1990) Overcoming constraints to effective elementary science teaching. Science Education, 74(4), 409–420.

Ulrich, W. (2001) A Philosophical Staircase for Information Systems Definition, Design and Development. Journal of Information Technology Theory and Application, 3, 55–84.

UML – Overview. Available at: http://www.tutorialspoint.com/uml/uml_overview.htm (accessed 24 August 2016).

Yetim, F. (2004) A Meta-communication Model for Reflective Practitioners. Available at: http://www.ics.uci.edu/~redmiles/chiworkshop/papers/Yetim.pdf (accessed 24 August 2016).