ComD2: Family of Techniques for Inspecting Defects in Models that Affect Team Communication

Adriana Lopes, Ursula Campos and Tayana Conte
USES Research Group
Instituto de Computação, Universidade Federal do Amazonas (UFAM)
Manaus, AM - Brazil
{adriana, usc, tayana}@icomp.ufam.edu.br

Clarisse Sieckenius de Souza
Semiotic Engineering Research Group
Departamento de Informática, PUC-Rio
Rio de Janeiro, RJ - Brazil
clarisse@inf.puc-rio.br

***Abstract* — Communication failures in software development teams can compromise the software quality. Therefore, identifying and mitigating risks for effective team communication are important activities in software development. Software models are one of the means of communication in development teams, because it communicates other members of the development team about the software. Thus, our research focuses on inspection techniques for identifying defects that affect the team communication through the software models. This paper presents a family of techniques for inspecting defects that affect team communication, called ComD2 (Communication between Designers and Developers). The ComD2 family was developed based on theories that investigate different ways of communication. For the time being, the ComD2 family has three specific inspection techniques for UML models, such as class diagrams, activity diagrams, and state machine diagrams. We performed a feasibility study and the results showed that the ComD2 family was considered useful for the identification of defects that affect the team communication through the software models.**

Keywords - inspection technique, communication artifact, UML diagrams, human-centered computing; software engineering;

I. INTRODUCTION

According to Reed and Knight [1], effective communication is one of the most critical components of working in software teams. In software development, the communication is carried out through face-to-face discussions in co-located or distributed teams [2], besides the support offered by tools [3]. Software models are also used as means of communication in software development teams [4]. In this paper, we explore the communication of software development teams through software models.

Software models that support the communication in different domains can be considered boundary objects. Boundary objects are used for different purposes and in different domains while maintaining their authenticity [5]. The term boundary object comes from the use of objects that facilitate the sharing of information across linguistic, cultural, or knowledge boundaries, such as the communication between a software development team and its client.

DOI reference number: 10.18293/SEKE2018-155

Communication failures from software models can come from information that is not clearly expressed by their producers (people who created the models). Thus, other members of the software development team (i.e. consumers, who comprehend the models for the creation of other artifacts) may have different interpretations of the ones intended by the producers. Different interpretations can introduce defects during the production of software; such as the omission of some necessary information or the vague definition of information, thus allowing multiple interpretations [6].

The propagation of defects associated with model representations can be costly, especially in large or complex system development projects. Can defects be detected and remedied by model consumers down the communication line? How do they affect communication in these and other (undetected) cases? Is there a way to prevent or at least minimize such defects? Our research aims to contribute to answer these questions and starts with the following interrogation: *What defects in software models can affect the communication of software development teams?* To answer this question, we have developed a family of techniques called ComD2 (Communication between Designer and Developers). The purpose of the ComD2 family is to support the identification of defects that affect the team communication, i.e. the communication of the designer[1] to the developers at development time. The collaboration between designer and developers is one of the factors for the success of software development [8]. We have initially developed three specific inspection techniques for UML class diagrams, activity diagrams, and state machine diagrams. The techniques were developed for these models because they are among the most frequently used in the industry [9].

The ComD2 family of techniques was developed based on theories related to Human-Centered Computing (HCC), a field of research that integrates theories and methodologies in research on machines, human and application domains [10]. In particular, the Semiotic Engineering, a theory originally proposed for Human-Computer Interaction [11][12], which has

[1] We use the term *designer* for the Software Designer, also called Information Architect, i.e., the professionals involved in designing the software solution.

been extended to account for HCC, investigating different forms of communication through and about software, both during development (between producers and consumers of software development artifacts) and at use time (between software producers and end users, through systems interfaces). De Souza et al. [12] propose that Grice's Cooperative Principle [13] can be used to assess the effectiveness and efficiency of communication achieved through software products (or representations thereof). Thus, we also adopted this Gricean principle as a theoretical basis for the development of the ComD2 family.

The verification items of the ComD2 family help practitioners classify different defects that are found in software models [14] and detect which dimension(s) of model representation associated with the defect can lie at the origin of potential miscommunication. The employed dimensions of representation are Syntactic (relationship between model and the modeling language), Semantic (relationship of the model with the problem domain) and Pragmatic (relationship of the model with the stakeholders) [15].

We conducted a feasibility study with the initial version of the ComD2 family in an academic environment. This study was performed with 30 participants who had knowledge on class diagrams, activity diagrams and state machine diagrams. The results showed that defects in the Semantic and Pragmatic dimensions may affect the effective communication between producers and consumers of the models. With the support of ComD2 family, we have indications about the different defects that impact the team communication. Therefore, Grice's Cooperative Principle and the different defects associated with the dimensions of representation are important concepts for reducing team communication failures using the models.

The remainder of this paper is organized as follows: Section 2 presents the theoretical background and related work. Section 3 presents the ComD2 family of techniques. Section 4 presents the feasibility study. Section 5 presents the discussion of the obtained results. Finally, Section 6 presents the final considerations and future work perspectives.

II. BACKGROUND AND RELATED WORK

This section presents the concepts used in the definition of the ComD2 family of techniques. We also present the main works related to the concepts adopted in the techniques.

A. *Grice's Cooperative Principle*

Grice's Cooperative Principle assists in the expression of essential characteristics of effective and efficient communication [13]. According to Grice, productive conversation (communication) depends on the observation of reciprocal cooperation, which is established by four maxims:

(i) **Quantity** - Make your contribution as informative as necessary, and no more than necessary;

(ii) **Quality** - Try to make your contribution true. Do not say what you believe to be false and do not say something that you do not have adequate evidence of;

(iii) **Relation** - Be relevant, that is, do not introduce issues that do not come to the case under discussion; and

(iv) **Manner** - Be clear, brief and organized with your input. Avoid obscurity of expression, ambiguity.

Breaking one or more of these maxims may lead to communication failure. However, an adequate use of Grice's maxims involves the concept of *implicature*, that is, information that can be inferred from statements. Conventional implicatures can be inferred from the conventional meaning of word. There are also conversational implicatures, that is, inferences that can be drawn from participants of a given conversational context in order to fulfill certain gaps and omissions to (re)establish coherence and consistency in communication. Therefore, unlike conventional implicatures, conversational implicatures cannot be resolved by invoking the usual meaning of information represented in communication and require different kinds of inferences.

Grice's maxims have been previously used by Santana et al. [16] to analyze interaction diagrams modeled with MoLIC (Modeling Language for Interaction as a Conversation). MoLIC diagrams, which are based on Semiotic Engineering, allow us to represent the interaction of the user with the system as a communication process. The results showed that Grice's maxims can indeed help detect human-computer interaction problems in MoLIC diagrams. We thus extend the object of inspection and use Grice's maxims to assess effective communication between producers and consumers of other kinds of software models.

B. *Defect Inspection in Software Models*

Software defects may be related to an inappropriate comprehension of the information within the models. Granda et al. [14] present a classification for defects that are commonly found in UML models, which are presented in Table 1.

TABLE I. TYPES OF DEFECTS, ADAPTED FROM [14].

Type	Description
Omission	The required information has been omitted.
Incorrect Fact	Some information in the model contradicts the list of requirements or general knowledge of the system domain.
Inconsistency	Information in one part of the model is inconsistent with information in other parts in the model
Ambiguity	The information in the model is ambiguous. This can lead to different interpretations of information.
Extraneous Information	The information that is provided is not required in the model.
Redundant	Information is repeated in the model.

Inspection is a method used to identify defects with lower cost during the development process [17]. According to Qazi et al. [18], the main purpose of an inspection is to identify defects to reduce costs and improve software quality.

Travassos et al. [19] developed a family of seven techniques for inspection, called OORTs (Object Oriented Reading Techniques). The techniques of the OORTS family can be used to inspect object-oriented models with regards to: (i) the different models used, such as use case diagrams and class diagrams, ensuring consistency among such models; and (ii) the requirements and models, ensuring traceability within a domain in order to find defects between them. However, we

did not find inspection techniques with the purpose of supporting the defects identification that affects the communication of the team through the use of models. For this reason, we proposed inspection techniques to identify defects that impact the communication of the team through the use of models. We have developed our techniques in the context of HCC and related theories that study different ways of communication.

III. THE COMD2 FAMILY

ComD2 offers specific techniques for inspecting defects that affect the team communication through the software models. The purpose of the ComD2 family is to assist experienced and novice practitioners by checking for defects in models that impair the communication of the team. The theoretical basis of ComD2 family is presented in the following paragraphs. From this theoretical basis, it is possible to develop specific techniques for other software models.

Using Grice's Cooperative Principle [13], ComD2 family uses the four maxims. We created verification items to support the identification of discrepancies (these discrepancies may be defects or not) based on these four maxims. Based on the maxim of Quantity, we developed verification items for the necessary content (and no more than necessary), in the models e.g., for the class diagram: "*Are all necessary classes of the problem domain in the diagram?".* Based on the maxim of Quality, we developed verification items for the identification of false information in the models, e.g. for the class diagram, we have: "*Do the classes have content that affects the quality of the model?"* Based on the maxim of Relation, we developed verification items for the identification of information that is not relevant to the models, e.g.: *"Are classes relevant to system modeling?".* Based on the maxim of Manner, we developed verification items for the identification of information that is not clear in the model, e.g.,: "*Are there classes and relationships with descriptions that are not clear?".*

We observed that when the maxims are not respected in the models, they could cause defects in software. This occurs due to the lack of understanding of the consumers on the intention of the producers regarding the model. Thus, for each verification item, we used the defect classification that is presented by Granda et al. [14] (see Table 1). From the verification items, it is possible to classify the defects. For instance, for the verification item that is based on the maxim of Quantity, we relate this item as follows: *Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy.*

Each verification item presents the dimensions of representation that was affected by the defects in the models, being these Syntactic, Semantic and Pragmatic [15]. These dimensions can help in comprehending the defects that cause the communication failures from software models. Defects related to the *form* of representation are associated with the syntactic dimension, whereas the defects related to the *content* of information are associated with the semantic and pragmatic dimensions. We highlight in the verification items the different dimensions of representation related to the defects. The following verification item shows the dimension of representation affected by the defect, which refers to the *content* of system information in the modeling: *Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy (Semantic).*

Verification items can be proposed for each one of the models. At this initial state, we have developed specific techniques for inspecting three UML models: class diagrams, activities diagrams, and state machine diagrams. These three models are commonly used in software development [9]. There are verification items for unique elements in the models, such as the following item for the Association element in the class diagram: *According to the problem domain, are all Association relationships established among classes? If not, this may be an Omission discrepancy (Semantic and Pragmatic).* In some cases, there are verification items for all elements of a model, such as the following item for the class diagram: *Are there elements with descriptions that are not clear? If so, there are probably Ambiguity discrepancy (Pragmatic).*

The Fig.1 presents some verification items for class diagrams. We use the following structure in the ComD2 family techniques: verification items for the elements of the respective models, which are related to the questions based on Grice's maxims. Each verification item suggests the identification of one or more defects and support the classification of the information representation dimension that such defects affect in the models.

Class Diagrams Technique	
Is the necessary information, and no more than necessary, present in the model?	
Class	Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy (Semantic).
...	...
Does the information in the model contain statements that are not true?	
Class	Do the classes have content that affects the quality of the model? If so, there are probably Inconsistency and/or Extraneous Information discrepancies (Semantic).
...	...
Is the information relevant to system modeling?	
Class	Are classes relevant to system modeling? If not, there are probably Extraneous Information and/or Redundant discrepancies (Semantic).
...	...
Is the information difficult to understand in the model?	
Class	Are there classes and relationships with descriptions that are not clear? If so, there are probably Ambiguity discrepancy (Pragmatic).
...	...

Figure 1. Extracts of the ComD2 techniques for class diagrams.

The verification items are divided into categories corresponding to the Grice's maxims (highlighted in gray in Fig. 1.), such as: **Is the necessary information, and no more than necessary, present in the model?** (verification items related to the maxim of Quantity); **Does the information in the model contain statements that are not true?** (verification items related to the maxim of Quality); **Is the information relevant to system modeling?** (verification items related to the maxim of Relation); **Is the information difficult to understand in the model?** (verification items related to the maxim of Manner). Despite the structure used, we do not define the order for the use of each category. The inspection techniques for the activity and state machine diagrams use the same structure presented in Fig.1. These techniques are available in [20].

IV. FEASIBILITY STUDY WITH THE COMD2 FAMILY

In order to evaluate the initial techniques of the ComD2 family, we conducted a feasibility study in an academic environment. In this study, we analyzed the effectiveness (ratio between the number of detected defects and the total number of defects) and efficiency (ratio between the number of defects per inspection time) of each participant for the different techniques. The adopted measures of efficiency and effectiveness are often used in studies investigating inspection techniques [21] [22]. We also evaluated the participants' perceptions on the techniques.

A. Planning and Execution of the Feasibility Study

In the planning stage, we selected 30 participants for the study. The participants are undergraduate students and have a basic dimension of knowledge about software modeling with class diagrams, activity diagrams and state machine diagrams. We selected the UML models of a real web and mobile development project. In addition, we prepared all necessary artifacts, such as forms for the participants to report the identified discrepancies and post-study questionnaires. The package with the artifacts used also available in [20].

In the execution stage, we first gave lectures on the techniques of the ComD2 family. Then, the participants performed the inspection of the UML models individually. After the inspection, we applied the post-study questionnaires. During the study, two of the researchers took notes for later analysis.

B. Results of the Feasibility Study

After the execution of the study, we verified whether the technique achieved the goal of detecting defects. The oracle of defects contained a total of 25 defects in the three diagrams (10 defects in the class diagrams, 8 defects in the activity diagrams and 7 defects in the state machine diagrams).

Table 2 presents the participants (column P#), number of defects found by each participant (CD column for the class diagram, AD column for the activity diagrams and SD column for state machine diagrams), inspection time (in hours) and the effectiveness of the participants (EfC column for the class diagrams technique, EfA column for the activity diagrams technique, EfS column for state machine diagrams technique) and the average effectiveness of each technique (in the last line in the Table 2). As the participants performed the inspection of the created models at the same time, we did not evaluate the individual efficiency of each participant with the techniques.

Analyzing the effectiveness indicator, we noticed that the inspectors were able to identify an average of 53% of the defects with the class diagrams technique, 35.8% with the activity diagrams technique and 25.8% with the state machine. This is a positive result in terms of effectiveness when compared to the indicators achieved by other inspection techniques for models [22]. The results showed that ComD2 can support the detection of defects. Regarding efficiency, as the participants used three techniques in the inspection of the models, we analyzed the efficiency of the entire ComD2 family. The participants found an average of 10.94 defects per hour with the techniques. However, as the number of defects is directly dependent on the inspected models, is not suitable to compare the results of efficiency from this study with the results of other techniques.

TABLE II. RESULTS PER PARTICIPANTS WITH THE COMD2 FAMILY.

P#	CD	AD	SD	time (hours)	EfC (%)	EfA (%)	EfS (%)	EfT (%)
P1	1	3	2	1,33	10	37,5	25	24,1
P2	6	4	3	1,21	60	50	37,5	49,1
P3	4	2	1	0,95	40	25	12,5	25,8
P4	5	3	1	0,81	50	37,5	12,5	33,3
P5	3	2	1	1,26	30	25	12,5	22,5
P6	4	3	4	1	40	37,5	50	42,5
P7	4	4	3	1,16	40	50	37,5	42,5
P8	8	2	3	0,63	80	25	37,5	47,5
P9	7	3	3	1,48	70	37,5	37,5	48,3
P10	7	1	1	0,83	70	12,5	12,5	31,6
P11	5	2	1	0,81	50	25	12,5	29,1
P12	9	2	1	0,81	90	25	12,5	42,5
P13	6	4	1	1,3	60	50	12,5	40,8
P14	9	3	3	0,95	90	37,5	37,5	55
P15	8	4	1	1	80	50	12,5	47,5
P16	4	4	2	1,01	40	50	25	38,3
P17	3	4	3	1,13	30	50	37,5	39,1
P18	4	4	4	0,96	40	50	50	46,6
P19	8	0	3	0,26	80	0	37,5	39,1
P20	5	2	1	1,16	50	25	12,5	29,1
P21	6	2	3	1,33	60	25	37,5	40,8
P22	2	3	1	1,06	20	37,5	12,5	23,3
P23	8	2	3	1,05	80	25	37,5	47,5
P24	4	1	2	1,06	40	12,5	25	25,8
P25	5	7	3	1,01	50	87,5	37,5	58,3
P26	4	3	2	1,05	40	37,5	25	34,1
P27	3	4	1	1,06	30	50	12,5	30,8
P28	7	2	2	1,21	70	25	25	40
P29	5	3	1	1,21	50	37,5	12,5	33,3
P30	5	3	2	1,21	50	37,5	25	37,5
Average Effectiveness					53	35,8	25,8	-

We analyzed the post-study questionnaires to understand participants' perceptions. The questionnaire had three open questions, the first question being: *What is your perception with the use of the techniques*? We analyzed the responses of participants P2, P12 and P13, who had more than 40% effectiveness in detecting defects with the ComD2 family (considering the effectiveness of the three techniques):

"The techniques are very practical. The dimensions of representation facilitate the review and help understand the intent of the artifact's author. With this classification, it is also possible to correct defects more easily" (P12)

"The techniques show an intermediate dimension of representation between the requirements and implementation. Therefore, it is a great way to analyze the team's understanding of the requirements" (P21)

Other participants reported perceptions that could be used to improve the techniques, such as joining some verification items that were considered repetitive. Some quotations from the participants were:

"The techniques help to ensure the reliability of the model, but some points are repetitive" (P12)

"The techniques can be more unified in the description of the problems, since some errors are the same in different models" (P22)

In spite of perceived issues, while answering the first question our participants considered the information representation dimensions useful, because they help gain better understanding of information that the model communicates. We believe this may improve the identification of defects that undermine the consumers' understanding of the intention of the producers of the models.

The second question posed to participants was: *Do the information representation dimensions help to understand the defects that could undermine the understanding of the model?* Some participants reported the following:

"Yes, especially the defects in the semantic and pragmatic dimensions that can compromise software development" (P3)

"Through the dimensions of representation, it is possible to see if we should change only something in the syntax or redesign parts of the system" (P13)

The participant's utterances showed that the defects in the semantic and pragmatic dimensions are the types of defects that may most affect the understanding of a model; since the syntactic dimension defects may not compromise both the understanding of the model language.

To understand if participants had difficulties with the ComD2 family, the post-study questionnaire included a third question: *What are the difficulties with using the techniques?* The following are excerpts from some of the answers.

"Although the techniques help in the classification of the representation dimensions, I had doubts with the classification of defects in Semantic and Pragmatic dimensions" (P6)

"Certain defects fall into more than one dimension, so it is necessary to evaluate and interpret each case in order to avoid misunderstanding" (P29)

Regarding the citations of participants P6 and P29, we noticed that there are difficulties with understanding the dimensions of representation and related defects. Although the techniques help in the classification of the dimension of representation associated to the defects, we can make improvements in the techniques with regards to the examples of the different dimensions of representation.

After the participants ended the study activities, we asked them which types of information in models could affect team communication. We noticed that some participants considered that unnecessary, irrelevant, ambiguous and false information affect the communication of the team when using these models. This type of information violates Grice's four maxims and indicates that this theory is adequate for analyzing communication between producers and consumers. Moreover, based on the results of this feasibility study, we define a prioritization of the categories in ComD2 family. The proposed prioritization follows this order: (1st) Quality, (2nd) Quantity, (3rd) Relation and (4th) Manner. This prioritization order should be followed in the next applications of the ComD2 family.

V. Discussion

The results of the study provided initial evidence to the feasibility of ComD2 family to inspect defects that may impact the communication between producers and consumers. Regarding our research question - *What defects in software models can affect the communication of development teams?* - the results obtained with the techniques showed that the defects in the Syntactic dimension do not always affect the consumers' understanding, since they are related to the syntax of the language used for modeling. Defects at the Semantic (relationship of the model with the problem domain) and Pragmatic (relationship of the model with the stakeholders) dimensions can affect the communication of the team. Regarding defects in the Semantic dimension, communication failures occur because the consumers infer explicit content inconsistent with the problem domain (conventional implicature of the explicit content in the model). However, if the consumers have knowledge about the problem domain, these defects can be perceived and not propagated to other artifacts (e.g. when the consumer uses the class diagram for system coding and it perceives the lack of a domain class, then this class could be added). Defects in the Pragmatic dimension may not be perceived by the consumers due to lack of information context (conversational implicature of the implied content in the model). In this case, communication failures occur because consumers may not understand the intention of producers. However, the ComD2 family can help reduce risks for effective team communication through models.

In the feasibility study of the ComD2 family, there are limitations, such as the fact that the participants are undergraduate students and that the study is conducted in an academic environment. Regarding this limitation, Fernandez et al. [8] state that undergraduate students who do not have experience in the industry may have similar skills to less experienced practitioners; and one of the goals of the techniques is to assist practitioners with no experience in the inspection process of models. Another limitation is that the inspected models were from a development project, since it is not possible to state that these models represent all types of class diagrams, activity diagrams and state machine diagrams. Therefore, we intend to carry out new studies with the set of techniques for different models. Regarding the indicators of effectiveness and efficiency that were adopted, they are often used in studies investigating inspection techniques [22].

VI. Concluding Remarks and Future Work

The purpose of this paper was to answer the following research question: "*What defects in software models can impact the communication of development teams?".* To do so, we developed a family of techniques called ComD2 that helps

practitioners identify defects that affect software team communication. We initially proposed and evaluated specific techniques of the ComD2 family for inspecting class diagrams, activity diagrams, and state machine diagrams. The results of the evaluation provided initial evidence to the feasibility of ComD2 family.

As future work, we intend to improve the ComD2 techniques and perform an empirical study in comparison with other specific techniques for inspecting class diagrams, activity diagrams and state machine diagrams. Furthermore, we intend to carry out a longitudinal study with the ComD2 family to evaluate the identification of defects that affect the team communication through the models employed during the software development.

ACKNOWLEDGMENT

We thank the undergraduate students for their participation in the feasibility study. We would like to thank the financial support granted by UFAM, CNPq through processes numbers 423149/2016-4, 311494/2017-0 and 304224/2017-0, and CAPES through process number 175956/2013.

REFERENCES

[1] A. H. Reed and L.V. Knight, "Effect of a virtual project team environment on communication-related project risk", International Journal of Project Management, vol. 28 (5), 2010, pp. 422–427.

[2] E. Diel, S. Marczak, D. S. Cruzes, "Communication Challenges and Strategies in Distributed DevOps", Proceedings of the 11th International Conference on Global Software Engineering (ICGSE 2016), 2016, pp. 24-28.

[3] V. Käfer, "Summarizing software engineering communication artifacts from different sources", Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017), 2017, pp. 1038-1041.

[4] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, "The impact of agile practices on communication in software development", Empirical Software Engineering, vol. 13 (3), 2008, pp. 303-337.

[5] P. Ralph, M. Chiasson and H. Kelley, "Social theory for software engineering research", Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16), 2016, pp. 44-55.

[6] R. M. de Mello, E. N. Teixeira, M. Schots, C. M. L. Werner and G. H. Travassos, "Verification of software product line artefacts: a checklist to support feature model inspections", Journal of Universal Computer Science, vol. 20(5), 2014, pp. 720-745.

[7] A. M. Qazi, S. Shahzadi and M. Humayun, "A comparative study of software inspection techniques for quality perspective", International Journal of Modern Education and Computer Science, vol. 8 (10), 2016, pp. 9-16.

[8] J. M. Brown, G. Lindgaard, and R. Biddle, "Collaborative events and shared artefacts: Agile interaction designers and developers working toward common aims," Proceedings - 2011 Agile Conference, Agile 2011, pp. 87–96.

[9] G. Reggio, M. Leotta, F. Ricca, and D. Clerissi, "What are the used activity diagram constructs? A survey", Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2014), 2014, pp. 87–98.

[10] Sebe, N. Human-centered computing. In Nakashima, H., Aghajan, H., & Augusto, J (Eds.), Handbook of ambient intelligence and smart environments, pp. 349–370, 2010. DOI: 10.1007/978-0-387-93808-0_13.

[11] C. S. De Souza, The Semiotic Engineering of Human-Computer Interaction (Acting with Technology). The MIT Press, 2005.

[12] Clarisse Sieckenius de Souza, Renato Fontoura de Gusmão Cerqueira, Luiz Marques Afonso, Rafael Rossi de Mello Brandão and Juliana Soares Jansen Ferreira. 2016. Software Developers as Users: Semiotic Investigations in Human-Centered Software Development. In Springer International Publishing Switzerland. DOI 10.1007/978-3-319-42831-4.

[13] H. P. Grice, "Logic and conversation". Syntax and Semantics 3: Speech arts, ed. Peter Cole and Jerry Morgan, 1975, pp. 41–58.

[14] M. F Granda, N. Condori-fernández, T. E. J. Vos, O. Pastor, "What do we know about the defect types detected in conceptual models?", Proceedings of the IEEE 9th Int. Conference on Research Challenges in Information Science (RCIS 2015), 2015, pp. 96–107.

[15] M. Priyanka and R. Phalnikar, "Generating UML diagrams from natural language specifications", International Journal of Applied Information Systems, vol. 1(8), 2012, pp. 19-23.

[16] B. S. Silva, V. C. O. Aureliano, S. D. J. Barbosa, "Extreme designing: binding sketching to an interaction model in a streamlined HCI design approach", Proceedings of the VII Brazilian Symposium on Human Factors in Computer Systems, 2006, pp. 101 – 109.

[17] P. C. Rigby and C. Bird., "Convergent contemporary software peer review practices", Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013), 2013, pp. 202–212.

[18] A. M. Qazi, S. Shahzadi and M. A Humayun, "Comparative study of software inspection techniques for quality perspective", International Journal of Modern Education and Computer Science, vol. 10 (8), 2016, pp. 9-16, DOI: 10.5815/ijmecs.2016.10.02.

[19] G. Travassos, F. Shull, M. Fredericks, V. Basili, "Detecting defects in object-oriented designs: using reading techniques to increase software quality", Proceedings of XIV ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, And Applications, 1999, pp. 47-56.

[20] A. Lopes, T. Conte, C. S. de Souza. 2018. ComD2 (Communication between Designer and Developers): A Family of Techniques for Inspecting Defects that Affect Communication from Models. USES Research Group Technical Report. TR-USES-2018-0003. Available: http://uses.icomp.ufam.edu.br/wp-content/uploads/2018/03/TR-USES-2018-0003.pdf

[21] A. Fernandez, S. Abrahão, E. Insfran, and M. Matera, "Further analysis on the validation of a usability inspection method for model-driven web development", International Symposium on Empirical Software Engineering and Measurement (ESEM 2012), 2012, pp. 153-156.

[22] N. M. C. Valentim, J. Rabelo, A. C. Oran, S. Marczak, T. Conte, "A Controlled Experiment with Usability Inspection Techniques Applied to Use Case Specifications: Comparing the MIT 1 and the UCE Techniques", Proceedings of the 18th International Conference on Model Driven Engineering Languages and Systems, (MODELS 2015), 2015, pp. 206-215.