Teaching Students How to Communicate Better Through Software Models

Adriana Lopes[1], Edson Oliveira[3], Tayana Conte[1], Clarisse Sieckenius de Souza[2]

[1] USES – Grupo de Usabilidade e Engenharia de Software
Universidade Federal do Amazonas (UFAM) Manaus, AM – Brazil
{adriana, tayana}@ufam.edu.br

[2] Semiotic Engineering Research Group
PUC-Rio, Rio de Janeiro, RJ – Brazil.
clarisse@inf.puc-rio.br

[3] Departamento de T.I.
SEFAZ-AM, Manaus, AM – Brazil.
edson.cesar@sefaz.am.gov.br

USES Technical Report
RT-USES-2019-0001
January, 2019

Institute of Computing (IComp)
Federal University of Amazonas (UFAM)
Manaus, Amazonas 69077-000

ABSTRACT

This technical report presents the material development base on Directives of Communicability (DCs) to teaching software engineering students how to communicate better through software models. The DCs were proposed to improve the quality of communication among software development team members. The development of DCs was based on Grice's Cooperative Principle and Semiotic Engineering, theories that investigate different ways of communication. Based on the DCs, we proposed a Learning Module with the material for training on the use of these directives. The Learning Module includes the following topics: training on software models as a means of communication; (ii) training on the use of the DCs; and (iii) examples of software model problems found when the DCs are not followed. With our proposal, we hope to train software engineering students that reflect on the content of models that will be consumed during the software development process. Furthermore, we present the DCs supporting material.

1. 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].

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 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 incorrect information into other artifacts and generate various defects during the production of software; such as the omission of some necessary information or the vague definition of information, thus allowing multiple interpretations [5]. Communication failures can occur because producers tend to focus only on the content of models, although they should also reflect on how model consumers will interpret them.

One of the causes of miscommunication through software models can be the type of training received by software engineers, in particular the approach used for teaching software models in undergraduate disciplines. Teachers typically emphasize the syntax and representation of the problem domain for modeling. However, students should also learn about which software model contents are appropriate to support model-mediated communication among software development team members.

2. BACKGROUND

2.1 Grice's Cooperative Principle

The Cooperative Principle proposed by Paul Grice [9] to characterize the logic of conversation can be used to characterize the communication between model producers and consumers as well. According to Grice, productive conversation (communication) depends on the observation of reciprocal cooperation, which is established by four maxims:

Quantity - Make your contribution as informative as necessary, and no more than necessary;

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;

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

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. But 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 in order 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.

2.2 Semiotic Engineering

Semiotic Engineering theory [7], characterizes user-system interaction as a particular case of human-mediated systems communication. Systems are considered metacommunication artifacts in Semiotic Engineering, i.e., artifacts that communicate a message from the designer to users about how they can or should communicate with the system to do what they want. The content of the metacommunication message, or metamessage, can be paraphrased in the following template:

"Here is my understanding of who you are, what I've learned you want or need to do, in which preferred ways, and why. This is the system that I have therefore designed for you, and this is the way you can or should use it in order to fulfill a range of purposes that fall within this vision".

Semiotic Engineering extended its original perspective to a Human-Centered Computing (HCC) perspective. HCC is a field of research that aims to understand human behavior by integrating technologies in social and cultural contexts [6]. This contribution is related to the set of conceptual and methodological tools called SigniFYI (Signs For Your Interpretation) [8]. The SigniFYI Suite assists in the investigation of meanings in a software during the development process and in the communication between producers and consumers of software.

3. DIRECTIVES OF COMMUNICABILITY

The DCs were developed with expressions that can characterize the producer's communication to consumers. To support the use of DCs, we based on the Semiotic Engineering metacommunication template to help producers think about consumers before model development. We adapted the original template of this theory to:

"Here is my understanding, as a producer of the model, of who is the consumer (to whom the producer is designing the model), what I've learned about what you need to do in system development (about what should be addressed in the model). This is the solution of the system that I designed for you to carry out your activities".

Based on this template, we developed questions that supports the reflection of the producers on the modeling. Below are the questions developed:

For whom is the model being designed? – *"Can the content of the model be comprehended so that the consumer accomplishes its objectives?"* – to support the students (or producer) to reflect whether the information in the model can be understood by everyone involved, such as developers and managers, or only developers;

What is being addressed in the model? – *"What content should be addressed about the system's problem/solution domain in the model?"* - in order to encourage the producer to reflect on the content that he wishes to be comprehended from the model, such as the tasks that a user can perform on the system. Fig. 1 presents an example about the DCs and the proposal that supports the reflection of the producers on the modeling.

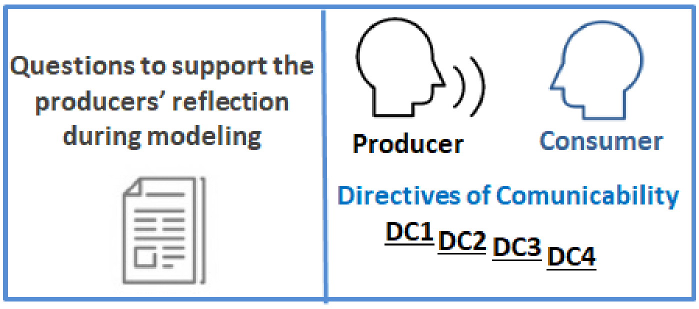

FIG. 1. EXAMPLE OF USE OF THE DCS.

Below we present each DCs. Table 1 shows the process to support the use of the DCs. The DCs are:

• *"Say the truth!"* - DC1: Use true information. Do not use information that affects the quality of the model (maxim of Quality).

• *"Say what is needed and no more than necessary"* - DC2: Use the necessary content in the template. Do not use unnecessary content in the model (maxim of Quantity).

• *"Say it logically"* - DC3: Organize the information in the model consistently (maxim of Relation).

• *"Say it clearly"* - DC4: Organize the information in the model clearly (maxim of Manner).

As suggested in Table 1, students can use the main DCs (DC1, DC2, DC3 and DC4) and combine them. This helps producers to keep previous DCs in mind while going through the list. For instance (for the combination of D2.1), when a producer uses the DC2 for a required information that is "missing" in the diagram, the producer should also be alert so as not to include information that is not true (DC1). This combination of the DCs evolved from discussions with a Semiotic Engineering expert and with the researchers involved in this work. We also proposed the Printed DCs, with the summary of the DCs. The Printed DCs should be used as additional support in the modeling.

TABLE 1. STEPS USING THE DCS.

S#	Description of use
Step 1	DC1 (maxim of Quality).
Step 2	DC2 (maxim of Quantity), with the following combinations: D2.1 – Regarding the information that is necessary and no more than necessary in the model, do not include information that affects the quality of the model (Quantity and Quality).
Step 3	DC3 (maxim of Relation), with the following combinations: D3.1 - In the case where incomplete or extra information is relevant in the model, justify (Relation and Quantity). D3.2 – In the case where false information is relevant, justify (Relation and Quality).
Step 4	DC4 (maxim of Manner), with the following combinations: D4.1 - Keep the conciseness, without the sacrifice of coherence (Manner and Relation). D4.2 - Keep the conciseness, without the sacrifice of what is needed (Manner and Quantity). D4.3 - Keep the conciseness, without sacrificing quality (Manner and Quality).

Teachers and professionals interested in using the DCs can obtain them in [10]. With our proposal, we hope to train software engineering students to reflect on the content of models that are consumed by other stakeholders.

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] 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.

[6] 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.

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

[8] 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.

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

[10] DCs Material (only the materials) Available:

https://drive.google.com/drive/folders/1z9nZgoWOjXm2Co0_IQ8gulYgR0h5M8NM?usp=sharing