Journal of Software Engineering Research and Development, 2021, 9:13, doi: 10.5753/jserd.2021.1942

Towards to Transfer the Directives of Communicability to Software Projects: Qualitative Studies

Adriana Lopes Damian [Federal University of Amazonas | adriana@icomp.ufam.edu.br]
Edna Dias Canedo [University of Brasília | ednacanedo@unb.br]
Clarisse Sieckenius de Souza [Pontifical Catholic University of Rio de Janeiro | clarisse@inf.puc-rio.br]
Tayana Conte [Federal University of Amazonas | tayana@icomp.ufam.edu.br]

Abstract

The software artifacts developed in the early stages of the development process describe the proposed solutions for the software. For this reason, these artifacts are commonly used to support communication among members of the development team. Miscommunication through software artifacts occurs because practitioners typically focus on their modeling, without reflecting on how other software development team members interpret them. In this context, we proposed the Directives of Communicability (DCs) to support practitioners analyzing characteristics that affect the artifact's content on communication via artifact. We conducted preliminary studies in a controlled environment with our proposal. However, we noticed that new studies are necessary to evaluate the DCs concerning practitioners' perceptions before transferring them to the industry. In this paper, we present two studies performed aiming to transfer the DCs to the software industry. In the first study, we evaluated the practitioners' perception about the DCs. In the second study, we evaluated the feasibility of the DCs in a software development team. The studies' results indicated that DCs have the potential to support improvements in artifacts' content to reduce miscommunication via artifact. To facilitate the use of our proposal in the software industry, we created procedures that support the adoption of the DCs and checklists for the application of each directive in the software artifacts. We noticed positive perceptions of practitioners about the application of DCs in software artifacts. We hope that our contribution support software development teams that use artifacts in your projects.

Keywords: *Communication via Software Artifacts, Human-Centered Computing, Semiotic Engineering*

1 Introduction

Artifacts developed in the early stages of the software development process, such as the different diagrams of the Unified Modeling Language (UML) (Freire et al., 2018; OMG, 2015), assist practitioners in understanding the problem for which software was required. As proposed solutions for software development are in artifacts, these artifacts also support team communication (Petre, 2013).

Communication is considered an important factor in software development, since miscommunication in software teams causes low productivity and software failures (Käfer, 2017). Miscommunication via artifact occurs, for example, when *consumers* (who take the information they see in the models for the development of another artifact) have different interpretations from the ones intended by the *producers* (who conceive the modeling of the software). As much as consumers know the modeling notation, the way the modeling has been expressed by their producer can affect these practitioners' mutual understanding.

In order to mitigate miscommunication via artifact, we proposed the Directives of Communicability[1] (DCs), presented in Lopes et al. (2019a). The DCs can support reflections to producers about how they can create a software solution via artifacts aimed to get a mutual understanding among development team members.

[1] Communicability in this context refers to the artifact's ability to convey to its consumers the solution conceived by its producers.

Practitioners can use our proposal mainly in the artifacts developed in the initial stages of the development process, such as UML diagrams, mockups and others. We conducted preliminary studies to evaluate our proposal to reduce miscommunication (Lopes et al., 2019a; Lopes et al., 2019b). However, we noticed that new studies are necessary to evaluate the DCs concerning practitioners' perceptions before transferring them to the industry.

Given the context above, we conducted an exploratory study (Lopes et al., 2020) to evaluate practitioners' perceptions of the DCs. Fifteen practitioners participated in this study by modeling UML use case (OMG, 2015) with the support of DCs. The results demonstrated that the UML use cases developed, with the support of DCs, had few risks of miscommunication. Besides, participants' perceptions about the DCs indicate that such directives can support better communication via artifact, contributing to software quality. However, it is also important to evaluate how software engineers apply the DCs in artifacts used in software projects to identify their feasibility.

This paper extends our previous work (Lopes et al., 2020), presenting a study carried out to analyze communication via artifacts in a software development team. We conducted this study in a software team with fourteen practitioners that worked on a cooperation project between the University of Brasilia (UnB) and the Brazilian Army. The results of this study showed the potential of the DCs to indicate improvements in the artifact's content regards communication via artifacts. In addition, we present proposals that facilitate practitioners to adopt the DCs, such as procedures that direct the adoption of the DCs in software artifacts and checklists that support the employ of each directive in common scenarios of two specific artifacts.

Through both studies, we noticed the contribution of the DCs for: (i) few risks of miscommunication via artifacts, allowing better communication via artifact; and (ii) improvements on the quality of artifacts, since miscommunication caused incorrect information in software artifacts. We hope that our contribution helps software development teams reduce miscommunication via artifacts.

2 Theoretical Foundations and Related Works

This section begins by presenting both the Semiotic Engineering theory (de Souza, 2005; de Souza et al., 2016) and the Grice Cooperation Principle (Grice, 1975), which we used to understand communication via artifacts and to propose the DCs. Additionally, we present related works to this type of communication.

2.1 Theoretical Foundations

Semiotic Engineering theory (de Souza, 2005; de Souza et al., 2016) 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 uses the communication space model proposed by Jakobson (1960), that is structured in terms of context, sender, receiver, message, code, and channel, where: "*A sender transmits a message to a receiver through a channel. The message is expressed in code and refers to a context*". Based on the communication space model proposed by Jakobson (1960), we can structure the communication elements via artifact in terms of the problem domain (context), how the artifact is available (channel), informational artifact's content (message) composed of the artifact's notations (code) for the communication between the producer (sender) and consumer (receiver) of an artifact, where: "*A producer transmits the informational content of the artifact to a consumer through a channel. Informational content of the artifact is expressed by the artifact's notations and refers to the problem domain*". Figure 1 presents a characterization of these elements.

Semiotic Engineering proposed evaluation methods to support designer-user communication in order to understand how the user is being receives the metamessage. The principle of categorization of communication failures presented by Semiotic Engineering is related to three categories:

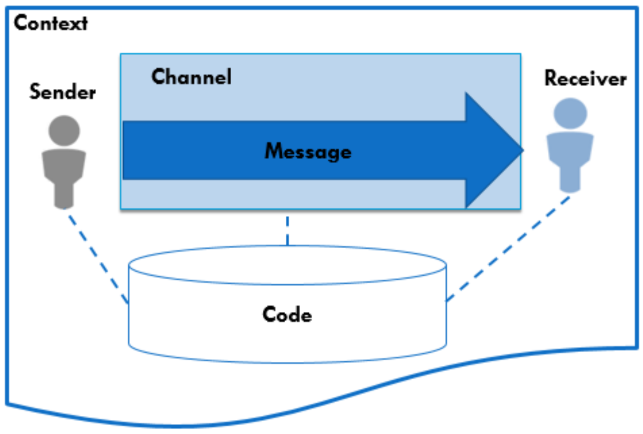

Figure 1. Communication space of Jakobson (1960).

- **Complete failures** - when the intention of the communication and its effect are inconsistent;
- **Partial failures** - when part of the intended effect of the communication is not reached; and
- **Temporary failures** - when in the intention of a communicative act between user and system, the user has momentary difficulty to continue talking with the system.

Semiotic Engineering extended its original perspective to a Human-Centered Computing perspective, a research field that aims to understand human behavior by integrating technologies in social and cultural contexts (Sebe, 2010). This contribution is related to the set of conceptual and methodological tools called SigniFYI (Signs For Your Interpretation) (de Souza et al., 2016). The SigniFYI Suite helps investigate meanings in software during the development process and the communication between software producers and consumers. Among them, the SigniFYI Message tool (SFYI Message) is the operational version of the metacommunication template. This operational version proposes that it can stand on its own as a powerful evaluation resource to identify communicability issues (which refers to the quality of the transmission of the solution designed by producers to consumers).

De Souza et al. (2016) report the use of a principle of reciprocal cooperation related to effective and efficient communication, called Grice's Cooperative Principle (Grice, 1975). This principle is expressed by four maxims. Breaking one or more of these maxims may lead to a communication failure. Grice's four maxims are:

Quality - try to make your contribution a true one. Do not say what you believe to be false and do not say something without adequate evidence. In software development, for example, the software engineer must communicate to the team only information that is related to the problem domain.

Quantity - Make your contribution as informative as is required. Do not make you contribution more informative than is required. Following the previous example, when communicating to the team, the software engineer must try to use only sufficient content to clarify the information they must develop.

Relation – Be relevant, that is, do not introduce points that do not come under discussion. In the case of systems developed in different cycles, each cycle must contain only information relevant to such development.

Manner - be perspicuous, avoiding obscurity of expression and ambiguity, be brief and be orderly. The software engineer must use descriptions that the team easily interprets, avoiding ambiguity.

2.2 Related Works

For the communication to be efficient, the sender must carefully choose an expression for the content he wishes to communicate, using a code that the receiver is able to interpret (de Souza, 2005; de Souza et al., 2016). In this sense, we identified works related to artifacts' comprehensibility, which refers to the receiver's interpretation of what the sender said in his communicative act.

On communication via artifact, Bordin and De Angeli (2016) point out that software engineers stated that documentation keeps a software development team aligned, especially in scenarios of distributed teams or with the introduction of new members in the team.

Schoonewille et al. (2011) present a contribution related to cognitive aspects in the understanding of software design documentation. They investigated, in one study, the participants' ability to extract information from diagrams and texts (grammatically and syntactically correct). The authors noticed that self-assessment could be problematic. They observed that developers were satisfied to "fill in" information missing from the documentation without the same understanding as the documentation producers. This can cause incorrect interpretations regarding the software.

Nakamura et al. (2011) proposed three metrics related to the comprehensibility of UML class diagrams in the following aspects: (1) class structure, (2) package structure and (3) attributes and operations. The authors claim that the metrics help in estimating the cost of time for understanding a class diagram.

Cruz-Lemus et al. (2010) present a predictive model of comprehensibility for UML state machine diagrams, analyzing its structural complexity. The authors' goal was to reduce the impact of understanding this diagram.

Tilley (2009) presents a work that summarizes 15 years of research on the use of graphical notation as documentation for understanding the system. According to the author, the graphical notation can help to understand the system and support communication. However, technical 'communicators' are not usually involved in this process. Still, according to the author, the result is that the engineers, who have the best of intentions, do not have the necessary background to explore the resources of the graphic notation to support end users' tasks. Therefore, the author reports a lesson learned: "we need to know how to talk". Therefore, this highlights the importance of the producer thinking about the consumers.

Lange and Chaudron (2006) present a work that investigated the effects of defects in UML diagrams in relation to different interpretations. They conducted two controlled experiments with a large group of students and practitioners. The two main contributions of this work are investigations on defect detection and different interpretations caused by undetected defects. The authors state that the results are generalizable for modeling with UML diagrams.

These works deal with topics related to communication with the support of artifacts developed in the early stages of software development. Schoonewille et al. (2011) and Tilley (2009) show the importance of artifact producers to reflect on consumers. Thus, it is important to have a proposal for artifacts producers to reflect on the consumers. The contributions of the DCs can help with this, as their goal is to support communication via artifact. This can be achieved when practitioners make improvements to the artifacts to obtain a mutual understanding of team members.

3 Directives of Communicability

For the DCs proposal, we have appropriated the communication space of Jakobson (1960) to communication via artifact, as follows: the artifact is made available with the support of a tool (the channel) with information from the problem domain (context) to support communication between artifacts producers (the emitters) and consumers (the receivers). The producer, in his message, must consider how the content is expressed (the use of the code) in such artifacts. Figure 2 shows such appropriation.

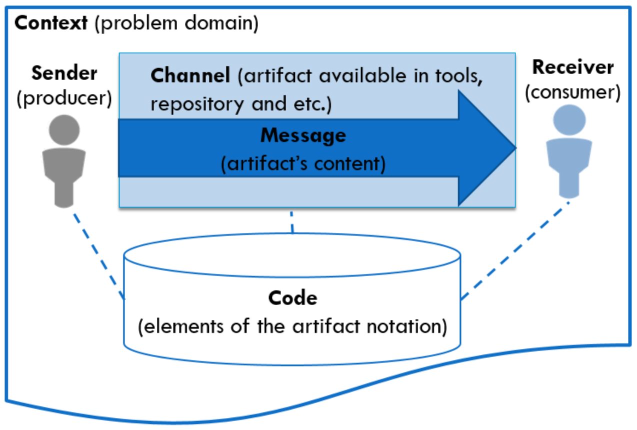
Context (problem domain)
Sender (producer)
Channel (artifact available in tools, repository and etc.)
Receiver (consumer)
Message (artifact's content)
Code (elements of the artifact notation)

Figure 2. Appropriation of the communication space of Jakobson (1960) for communication via artifact.

Besides, we have appropriated of Semiotic Engineering to define the following concepts related to communication via artifact:

- **Communicability of software artifacts** - refers to the artifact's ability to transmit to its consumers the proposed solutions for software development.
- **Communicability issues in software artifacts** – refers to the expressions or features of the artifact that can be directly associated with an incompatibility between meanings associated to them by their producers and consumers.
- **Risks of miscommunication via artifacts** – the

likelihood of a communicability issue to cause communication failures between producers and consumers.

- **Miscommunication via artifacts** – incompatible interpretations by artifacts consumers from the producer perspective for software modeling**.**

3.1 Proposal of the DCs

We elaborated the DCs based on Semiotic Engineering (de Souza, 2005; de Souza et al., 2016) and Grice's Cooperative Principle (Grice, 1975). We adapted the original Semiotic Engineering *metacommunication* template as follows: "*Here is my understanding as a producer of the model, of who you are, as its consumer (to whom the producer is designing the model), what I have 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, we created the following questions to help producers reflect on artifacts consumers:

(i) *Can the consumer understand the artifacts' content? Can the consumer achieve its goals?* – to support producers to reflect on whether everyone involved can understand the information in the model, such as developers and managers, or only developers; and

(ii) *What content should be addressed about the domain of the problem/solution of the system in the artifact?* - In order to encourage the producer to reflect on the content that she wishes to be comprehended from the model, such as the tasks that a user can perform on the system. These questions are used before the use of the DCs.

Regarding the information related to models' content, the DCs use the four maxims of Grice's Cooperative Principle. The directives will allow producers to reflect on the models' content before they send it to the consumer, so that there is mutual comprehension in software development teams. With this, the DCs can improve the model's ability to convey to its consumers the solution conceived by its producers. Below we present each DC, based on Grice's maxims:

"Say the truth!" - DC1: Use true information. Do not use information that affects the content quality in the model (maxim of Quality). In the UML use case diagram, for instance, do not insert use cases that are outside the problem domain:

"Say what is needed and no more than necessary" - DC2: Use the necessary content in the model. Do not use unnecessary content in the model (maxim of Quantity). Analyze, for instance, the amount of information in the specification of all use cases;

"Say it logically" - DC3: Organize the information in the model consistently (maxim of Relation). For example, organize the use cases in the diagram so that they present a logical sequence for the producers;

"Say it clearly" - DC4: Organize the information in the model clearly (maxim of Manner). Describe the names of the use cases so that they are easily understood and differentiated from each other.

3.2 How can software engineers can apply the DCs?

We designed the DCs to be employed by software engineers in artifacts that represent aspects of the software developed from their perspectives, such as UML diagrams, BPMN diagrams, and prototypes. In the study presented in Subsection 4.2, a team adopted UML use cases and prototypes to represent their software development decisions. The DCs can reduce the risks of miscommunication via these artifacts. Figure 3 presents a schematic of how software engineers can apply the DCs into UML use cases.

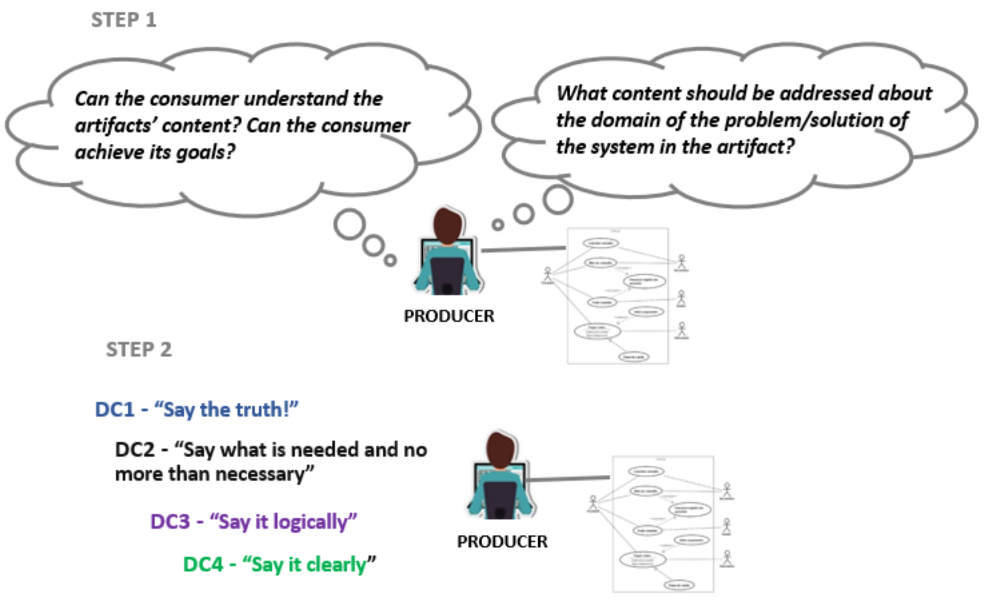
STEP 1
Can the consumer understand the artifacts' content? Can the consumer achieve its goals?
What content should be addressed about the domain of the problem/solution of the system in the artifact?
PRODUCER
STEP 2
DC1 - "Say the truth!"
DC2 - "Say what is needed and no more than necessary"
DC3 - "Say it logically"
DC4 - "Say it clearly"
PRODUCER

Figure 3. Directives of Communicability for software artifacts.

In step 1 of Figure 3, the producer begins his process of reflection on the consumers of the artifact produced based on the proposed questions. In step 2 of Figure 3, the producer is able to obtain a better use of the directives in modeling, so that mutual understanding occurs. For example, consider a practitioner modeling a use case for a system that supports users in the use of medicines. By using DC2, based on the practitioner's reflection, if the producer knows that the consumer can recognize the difference between the 'reminders' and 'notices' elements that will be used in the system, there is no need to detail the difference between them. If the consumer does not know such a difference, it is important that the producer describes this. DC2 will support this producer in producing use case specifications with the amount of information needed for those elements.

Regarding the use of the DCs, producers can use them in digital format, available in a technical report (Lopes et al., 2021), or print them to put on their workstations. We just emphasize that it would be interesting for the producers had access to directives during the development of the artifacts. In Section 5, we present proposals that can help software engineers adopt the DCs in software projects.

About the users of our proposal, we created the DCs to be used by both beginners and experienced software engineers, since they know the modeling notation. We emphasize that the DCs support producers reflect on the artifact's content to achieve a mutual understanding among the members of a software development team and not in modeling errors.

3.3 Preliminary studies with the DCs

In Lopes et al. (2019a), two software engineers, with the same level of experience in modeling, produced artifacts. One of them used the DCs and the other did not. Then, 30 participants were invited to create mockups based on the artifacts produced by the software engineers. We divided the participants into two groups. The experimental group created the mockups based on the artifacts produced with the DCs and the control group used the mockups based on the artifacts developed without the DCs. We noticed that the experimental group had a lower number of miscommunication.

In Lopes et al. (2019b), the DCs were also analyzed to reduce the risk of miscommunication in software artifacts, such as UML class diagrams, BPMN (Business Process Modeling and Notation) diagrams (OMG, 2011) and IFML (Interaction Flow Modeling Language) (Brambilla and Fraternali, 2014). We choose these diagrams for different communication purposes during software development. Twenty-four participants, divided into two groups, based on a modeling scenario, produced such diagrams. The experimental group used the DCs and the control did not use the directives. The experimental group created artifacts with a lower number of risks of miscommunication compared to the control group.

In Lopes et al. (2019a) and Lopes et al. (2019b), we presented studies with quantitative analyzes. However, it is important to carry out qualitative studies on the DCs before transferring them to the industry. For this reason, we planned new studies that aim to analyze practitioners' perceptions about the directives. Figure 4 presents a timeline about the studies carried out and our planning regards the new studies, which to answer the research questions (RQ) below.

RQ1 - Do practitioners perceive the DCs as support in improving the quality of artifacts?

RQ2 - Is the DC application by producers feasible in development teams?

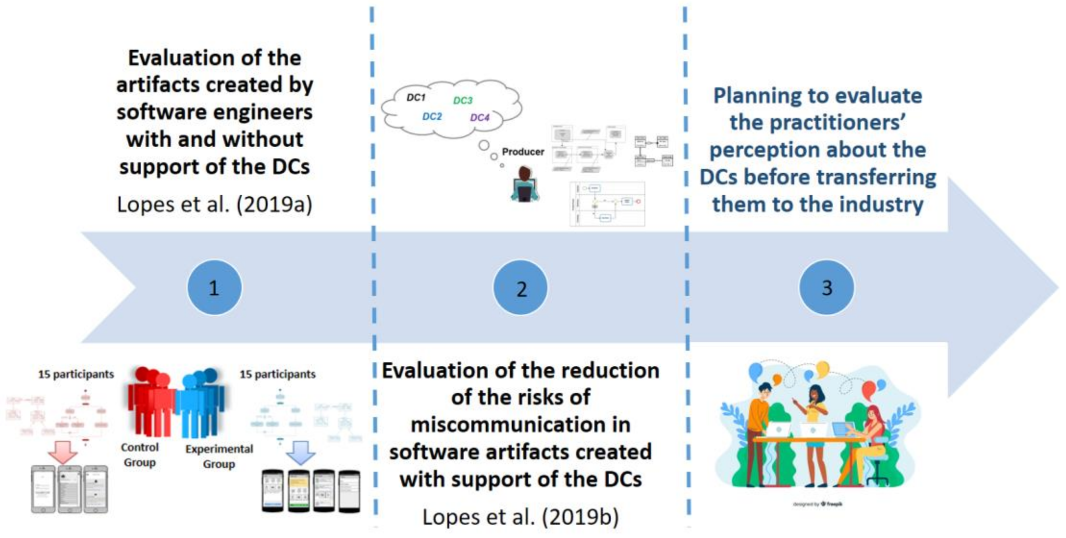
Evaluation of the artifacts created by software engineers with and without support of the DCs
Lopes et al. (2019a)
DC1
DC3
DC2
DC4
Producer
Planning to evaluate the practitioners' perception about the DCs before transferring them to the industry
1
2
3
15 participants
Control Group
Experimental Group
15 participants
Evaluation of the reduction of the risks of miscommunication in software artifacts created with support of the DCs
Lopes et al. (2019b)

Figure 4. Timeline of preliminary studies with the DCs and planning of new studies in the software industry.

4 Experimental Studies

This section presents the studies carried out with practitioners before transferring the DCs to the industry. In the first study (Study 1), fifteen practitioners participated. They created a UML use case with the support of the DCs. Our main goal in this study was to analyze the communication intention by artifacts producers. After the study, the participants provided their perceptions about the DCs through a questionnaire.

In the second study (Study 2), we carry out a study in the context of a software project. We evaluate if the DCs can provide supports to identify risks in software artifacts that caused miscommunication. In addition, producers and consumers provide their perceptions about communication via artifacts through interviews and an online questionnaire.

4.1 Study 1: Evaluation of the DCs from the practitioners' perception

We conducted a first study that evaluates the practitioners' perception of the DCs from 15 practitioners regarding their support during artifacts development (Lopes et al., 2020). In this study, the participants applied the DCs in UML use cases, that is, in the use case diagram and specification. After that, we sent questionnaires to collect the practitioners' perceptions.

As in this study we evaluated the practitioners' perception of the DCs during the modeling of use cases. We did not investigate the communication between producers and consumers. Therefore, the researchers analyzed only the possibility of a risk of miscommunication in the use cases. In addition, we analyzed the impact on the quality caused by the risks of miscommunication and qualitative data obtained about the practitioners' answers.

4.1.1 Study 1: Planning

We selected 15 practitioners to produce UML use cases with the support of the DCs. All practitioners had a college degree and they were taking the Fundamentals of Software Engineering class in Software Engineering postgraduate course at Northern University Center (UNINORTE). Table 1 presents a summary of the participants' experience. Regarding the participants, most of them did not work creating artifacts in software projects related to our research. However, we consider practitioners who are consumers in software projects able to participate in the study, because they can provide their perception into our proposal to communication via artifacts. In this way, we planned training so that participants execute the study activities.

We planned this study to take in a single day, during the morning and afternoon. In the morning, before we carried out the study, the participants received training of approximately two hours for exercising use case modeling. It is noteworthy that all participants had prior knowledge of UML use cases. In the afternoon, we reserved a laboratory for the execution of this study, which had notebooks for the participants to use. We planned to run this study in approximately three hours.

Table 1: Participants' experience in software industry

EXPERIENCE IN THE INDUSTRY	PARTICIPANTS
1 – 3 years	P1 (Developer) P2 (Software Tester Developed) P3 (Software Analyst) P4 (Developer) P6 (Process Engineer) P7 (Developer) P10 (Developer) P12 (Developer) P13 (Developer)
4 – 8 years	P8 (Software Tester Developed) P9 (Developer) P15 (Developer)
more than 9 years	P5 (Developer) P11 (Developer) P14 (Project Manager)

In order to observe the participants' discussion regarding the development of use cases in different modeling scenarios, we randomly defined four groups. Each modeling scenario had simple content, so that the participants complete the study activities in the planned time. We present the description of the modeling scenarios for each group below:

Group 1 scenario – To support students who want private lessons in basic classes such as Mathematics, a system must be developed. The system should provide teachers with private lessons. Additionally, evaluations of these teachers by students/other teachers should be displayed. The system should allow managing the teachers' agendas on the classes, so that students can enroll in them. Thus, it is possible to include and cancel classes.

Group 2 scenario – To support the small events, a system must be developed. In this system, the organizers will be able to create their accounts and, from this, register events such as birthday parties, guest lists, and gift lists. They will also be able to send invitations via e-mail, control expenses, and generate reports for both guests and expenses. The system provides communication among organizers and guests. Guests may or may not confirm their presence at the event and consult the gift list.

Group 3 scenario – To support sales professionals in their orders, such as delivery control, customer management (retailers and wholesalers), a system must be developed. The system will support professionals who want to computerize and innovate the service, minimizing errors and constraints from the lack of systematic control. The system should allow users to register their customers and to manage the stock of their products. After the payment record, the order is sent to the customer with the delivery invoice.

Group 4 scenario – To support residents of the state of Amazonas in Brazil, who have difficulty accessing the information on river routes for purchasing tickets, a system must be developed. The system will support passengers of different vessels, embarking/disembarking times, the vessels' capacity, number of available spaces, price, and information on river routes. Concerning vessels' owners, they will be able to register the number of employees available for passenger assistance.

Before the execution of the study, we planned training with the DCs to be applied in use case modeling. Regarding the study activities planned and Figure 5 summarizes them.

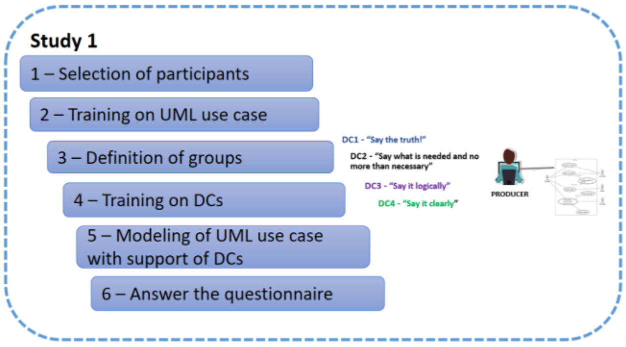
Study 1
1 – Selection of participants
2 – Training on UML use case
3 – Definition of groups
4 – Training on DCs
5 – Modeling of UML use case with support of DCs
6 – Answer the questionnaire
DC1 - "Say the truth!"
DC2 - "Say what is needed and no more than necessary"
DC3 - "Say it logically"
DC4 - "Say it clearly"
PRODUCER

Figure 5. Study 1 activities planned.

To analyze the defects related to the risks of miscommunication, we used the types of defects presented by Granda et al. (2015). Table 2 shows these defects.

Table 2. Types of defects (adapted from Granda et al. (2015))

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.

4.1.2 Study 1: Execution

We asked the participants to position themselves according to the groups defined to carry out the study activity. The participants were in the same laboratory, but the groups were far from each other. After that, we delivered to the groups the modeling scenarios and the printed DCs.

The main researcher, the participants to draw up the use case diagram together, discussing relevant aspects of the system. After that, the researcher requested each participant to specify only one use case. The participants created the use cases from the modeling scenarios. Regarding the use of the DCs, the participants should, for example, create use cases in the context of the problem domain (use of DC1) and analyze the amount of information to understand these use cases correctly (use of DC2). During the study, a researcher a researcher took notes of the directives most used by the participants.

The participants used the Astah[2] tool to model use cases. Table 3 presents the four groups defined with the participants and the objective of each system in the modeling scenarios.

Table 3. Groups defined in this study

GROUPS	PARTICIPANTS
Group 1	P4, P5, P6 e P7
Group 2	P8, P9, P10 e P11
Group 3	P12, P13, P14 e P15
Group 4	P1, P2 e P3

Regarding the use of the DCs, the main researcher informed the participants the directives can be applied according to the most appropriate for them, such as by using the directives during the modeling or after the participants made a modeling proposal. The main researcher noticed that all groups made a modeling proposal and then applied the DCs.

At the end of the study, all participants answered a post-study questionnaire to provide their perceptions about the DCs, including each participants' experience in the industry. Regarding the duration of the study, it was completed ahead of our planning.

4.1.3 Study 1: Results

We analyzed the use cases produced by the groups regarding the risks of miscommunication, which were discussed with the other authors of this paper. The risks of miscommunication identified in the use cases of each group are shown in Table 4, including their total number of occurrences and the description of each risk.

Table 4: Risks of miscommunication in the use cases developed by the groups

GROUPS	DESCRIPTION OF THE RISKS OF MISCOMMUNICATION
Group 1	- Lack of relationship in the use case diagram (1) - Different standards in the organization of the use case specification (3) - Lack of information in business rules (5)
Group 2	- Use case specification inconsistent with the use case diagram (1) - Lack of relationship in the use case diagram (1) - Lack of information in business rules (4)
Group 3	- Lack of information in business rules (2) - Lack of steps in the main flow of the use case specification (2)
Group 4	- Lack of steps in the main flow of the use case specification (2) - Lack of information in business rules (5)

In this analysis, for example, we noticed that the participants in Group 1 did not provide all the necessary information in the business rules, such as the fields in the system for a student to evaluate the teachers. The evaluation of the

[2] https://astah.net/

artifacts produced by the groups showed few risks of miscommunication compared to the number of risks of miscommunication identified in other software artifacts in a preliminary study (Lopes et al., 2019b). However, such risks can cause possible miscommunication.

Regarding the application of the DCs by the participants, based on the researcher's notes during the study, the majority of them used the following directives: DC2 to evaluate the amount of information that should be represented, and DC3 for the organization of information logically in use cases.

Analysis of software defects related to risks of communication failure. We grouped the risks of miscommunication in Table 5, which are related to the groups. Defects related to risks have also been described in this table.

Regarding the risks of miscommunication, we noticed a lack of information for the business rules in the four modeling groups. Besides, there was a lack of information for the relationship between use cases in the diagrams produced by the two groups. There was a lack of specification of steps in the main flow of use cases of two groups. These risks would be mitigated if the participants had reflected better on the amount of information, related to DC2. Regarding the risks related to the lack of standardization of use case specification itself and inconsistency between the use case specifications and the use case diagram, this would be mitigated with DC4 and DC1, respectively.

Table 5. Defects related to the risks of miscommunication in the use cases developed by the groups

GROUPS	DESCRIPTION OF THE RISKS	DEFECTS
Group 1 Group 2	Lack of relationship in the use case diagram	Omission
Group 1	Different standards in the organization of the use case specification	Ambiguity
Group 1 Group 2 Group 3 Group 4	Lack of information in business rules	Omission
Group 2	Use case specification inconsistent with the use case diagram	Inconsistency
Group 3 Group 4	Lack of steps in the main flow of the use case specification	Omission

Regarding the risks related to the lack of information in: (i) the business rules, (ii) main flow steps in the use case specification, and (iii) relationships between use cases in the diagram we considered them to be an 'Omission' defect. Different standards in the specification of use cases may allow different interpretations by consumers, which we considered an 'Ambiguity' defect. Finally, we considered inconsistent information between the use case diagram and the use case specification to be an 'Inconsistency' defect.

Analysis of the Participants' Perception. Regarding the post-study questionnaire, the participants answered the following question: "What is your perception of the directives of communicability?" We defined this question in a general way to collect different opinions of the participants on the DCs.

To analyze the qualitative data obtained in the study according to Strauss and Corbin (1998), researchers can use coding procedures to achieve their research objectives. We used open coding to understand participants' perceptions. With that, we observed the following codes:

DCs contribute to the quality of software artifacts

"The directives help to reflect on what should be developed, avoiding inconsistencies" (P5)

"The directives help to understand the system, support to identify possible errors" (P8)

"Facilitates the identification of problems in modeling" (P13)

DCs promote the organization of information in artifacts

"The directives helps to organize and improve the information required to create a system" (P3)

"DCs assist in organizing ideas together with the development team" (P10)

"The directives help to organize thoughts when designing the system" (P2)

DCs support the understanding of the system

"DCs assist to obtain relevant information for the project" (P4)

"The directives provide great support for the production of the software" (P7)

"Helps to considerably improve the general understanding of a system" (P15)

DCs can promote effective communication via artifact

"DCs are a type of roadmap for organizing ideas in communication through a logical way" (P11)

"They help to think about how to communicate with colleagues" (P6)

"Help in communicating correctly in software development" (P12)

DCs promote the reduction of different interpretations

"The directives help reduce the multiple interpretations of the same idea, as the ideas must be conveyed so that everyone understands" (P14)

Difficulties with the use of the DCs

"It is not easy to understand the directives; it required more of my mental effort" (P2)

"It is not easy to apply the directives; I believe it depends on the user's experience" (P5)

"Directives demand time for understanding" (P6)

Through the participants' perceptions, we observed that the DCs contribute to the improvement of the quality of the artifacts. Such perceptions are represented by the codes 'DCs contribute to the quality of software artifacts' and 'DCs promote the organization of information in artifacts'. Most of the participants' responses showed that they perceived the purpose of the DCs, as we noticed the codes 'DCs can promote effective communication via artifact' and 'DCs promote the reduction of different interpretations'.

Some participants also reported 'Difficulties with the use of the DCs', which may be related to their reflection on whether or not they are correctly applying the main concept of each directive for what each producer wants to communicate. However, this is part of the reflection process by producers regarding their communication through artifacts.

Analysis of acceptance. We applied the Technology Acceptance Model (TAM) to analyze the participants' perception of the DCs in the post-study questionnaire (Venkatesh and Bala, 2008). TAM is one of the most adopted models for collecting information about the decision to accept or reject technologies (Marangunić et al., 2013). This model is basically based on two constructs:

Perceived Ease of Use: degree to which a user believes to use a specific technology with little effort.

Perceived Usefulness: the degree to which a user believes that using a specific technology would improve their performance at work.

The user's behavioral intention to use a technology, the *Intention to Use*, is determined by the perceived ease of use and perceived usefulness. The statements contained in the post-study questionnaire to assess the constructs of ease of use, usefulness, and intention to use the DCs, adapted from [25], are presented below:

PERCEIVED EASE OF USE

E1. My interaction with the Directives of Communicability is clear and understandable.
E2. Interacting with the Directives of Communicability does not require a lot of my mental effort.
E3. I find the Directives of Communicability easy to use.
E4. I find it easy to get the Directives of Communicability to do what I want it to do.

PERCEIVED USEFULNESS

U1. Using the Directives of Communicability improves my performance better for understanding aspects of the software.
U2. Using the Directives of Communicability in my job has improved my productivity, since I will not have to correct information that is not understood by colleagues.
U3. Using the Directives of Communicability enhances my effectiveness on communication with the team based on the artifacts.
U4. I consider the Directives of Communicability useful for software design.

INTENTION TO USE

I1. Assuming I had enough time to design software, I intend to use the Directives of Communicability.
I2. Considering that if I could choose any tool, I predict that I would use the Directives of Communicability.
I3. I plan to use the Directives of Communicability in my next project.

Regarding the adapted TAM statements, participants provided their answers on a seven-point Likert scale (Likert, 1932). The possible answers were "Totally Agree, Strongly Agree, Partially Agree, Neutral, Partially Disagree, Strongly Disagree, and Totally Disagree". The participants answered their degree of agreement on the usefulness, ease of use, and intention to use the DCs in the production of artifacts. Figure 6 summarizes the participants' answers.

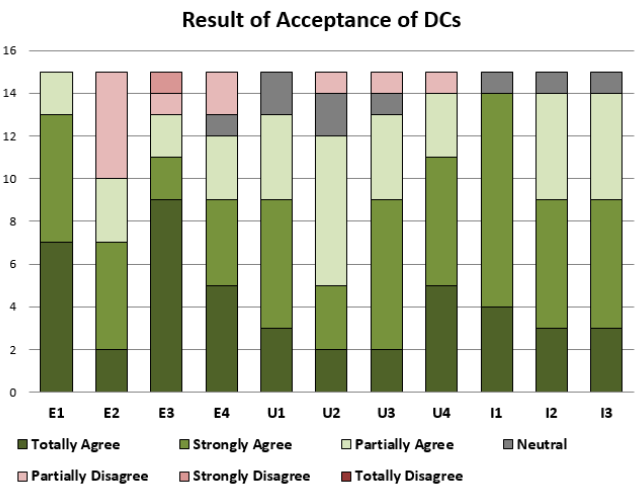

Figure 6. Degree of participants' acceptance regarding the use of the DCs in the production of artifacts

Regarding the disagreements related to E2, E3 and E4 for ease of use, as shown in Figure 6, we noticed five participants answered that, including P2, P5 and P6 that cited it's not easy to employ DCs, represented by the 'Difficulties with the use of the DCs' code in Subsection 5.2. The other participants did not provide answers to explain why they disagreed with E2. In summary, such answers may indicate that it is important to provide material that helps in the producer's reflection based on the DCs.

About the disagreement and neutral answers with the statements that measure usefulness, we noticed that P3, P6, and P11 answered this. However, all participants who consume information from artifacts, i.e. developers, agree that our proposal is useful to communication via artifact. Overall, most of the participants' answers showed agreement regarding ease of use, usefulness, and intention to use the DCs.

With this research (Lopes et al., 2020), we observed that the DCs promoted the participants' reflection on their communication to the others involved in the development of a software. The DCs also made it possible to reduce the introduction of defects, because we perceived consistent mapping between the risks of miscommunication and software defects. Additionally, most of the participants' answers to the DCs were positive about their use. With this, it is possible to infer that the software industry considered the directives useful.

Based on the results obtained in this study, we decided to carry out a feasibility study in a software development team. This study may increase the indications on the transfer of the DCs to the industry.

4.1.4 Threats to Validity of Study 1

In all experimental studies, there are threats that can affect the validity of the results. The threats related to this study are discussed below with the classification of threats to validity presented by Wohlin et al. (2012):

Internal validity. Training effect - it would be interesting if there was no need for training. However, the short training time allowed the DCs to be used by practitioners during the production of UML use cases. In addition, training on use case modeling also enabled participants to execute the study activities, as most of them did not work creating artifacts that represent software decisions in projects. Time used for the study - despite the time considered long for the use case modeling, all participants completed the study activities before the expected time.

External validity. Validity of the artifacts – we carried out only modeling of UML use cases in this study. It is not possible to claim that UML use cases represent all the artifacts that support communication. Besides, the use cases were modeled for four software projects. It is not possible to claim that this artifact represents all types of software.

Construct validity. Indicators for miscommunication - The measures adopted to analyze miscommunication were based on the Semiotic Engineering theory (de Souza, 2005; de Souza et al., 2016), which has different methods to assess communication during the development process.

Conclusion validity. There is a limitation in the representativeness of the results, a known problem in experimental studies of Software Engineering (Fernandez et al., 2012). The results obtained in this study may not be reproduced in other software artifacts that support the understanding of members of a team. Analysis of Artifacts – about the risks of miscommunication in use cases, there is a threat regarding the researcher who carried out such analysis. To mitigate this threat, we added another researcher to discuss this analysis.

4.2 Study 2: Feasibility Study

In study 1, although we perceived positive answers about the DCs, we did not carry out the study in the context of a software project. We carried out another study, our second study, in a software development team. We investigated the use of the DCs in the artifacts used by the team to identify risks that caused miscommunication in the development of the Bulletin System (SISBOL).

SISBOL is a Web System, with client-server architecture, following the standard Representational State Transfer (REST) [1], with the purpose of automating the process of generating newsletters (official) and managing the members' personal historical (changes of the military) of the Brazilian Army (EB). A bulletin represents an instrument by which the commander, chief or director of the EB disseminates the orders of the higher authorities and the facts that must be known by the Military Organizations in which the members participate. SISBOL is composed of entities associated with military, such as qualification, graduation, subunit/division/section, military organization, function and alteration, associated with the bulletin structure (type of bulletin, section, part, general subject, specific subject, note) and associated with system users. Notes are documents proposed by a competent authority to be approved by the commander, chief or director, for publication in its bulletin. The system has a certain degree of configurability, allowing the approval processing workflows for notes and bulletins to be customized for each military organization.

4.2.1 Study 2: Planning

We initially designed the study to analyze how the team conducted its activities and how software artifacts support communication. Then, we planned the analysis of the artifacts with the support of the DCs to identify opportunities for improvement to a better communication. Finally, we planned a collection of the team members' perception of the support of the artifacts.

The team selected for the study was composed of 14 practitioners who developed the SISBOL. Table 6 shows the characterization of the team.

Table 6. Characterization of the team

TEAM	EXPERIENCE
Systems Analyst (Product Owner- PO)	20 years
Designer	9 years
Developer 1	7 years
Developer 2	20 years
Developer 3	5 years
Developer 4	16 years
Developer 5	4 years
Developer 6	4 years
Developer 7	19 years
Developer 8	12 years
Developer 9	12 years
Developer 10	3 years
Developer 11	10 years
Developer 12	3 years

The scope of the new SISBOL involves 30 functionalities, which were divided into Legacy Features (23) and New Features (07). The team used the agile development methodology. The development team used the agile Scrum methodology. The artifacts elaboration process was collaborative and involved different project stakeholders.

The team used UML use cases and prototypes as artifacts that contain the solution designed for software development. Regarding the team selected, the practitioners did not create a domain model, just the use cases and mockups. About the experience of the producers, the system analyst had twenty years of experience with UML and the designer had nine years of experience with prototypes in projects. Regarding the system developed by the team, it was already in its final phase, as the team was only making corrections to some features.

4.2.2 Study 2: Execution

We carried out the following steps in this study:

(i) a meeting between the PO and the main researcher in order to obtain an overview of the activities and the artifacts used as a mean of communication;
(ii) meeting among the main researcher with the teams' producers to analyze the artifacts' content based on the DCs;
(iii) after that, we prepared an electronic questionnaire for producers to answer their perceptions of the artifacts as a support for communication;
(iv) a meeting of the main researcher with an artifacts consumer to understand how they used the artifacts;
(v) we also sent an electronic questionnaire for consumers to answer their perceptions of the artifacts with questions based on the DCs. Due to the unavailability of some participants to participate in individual meetings, this questionnaire facilitated the collection of team members' perceptions.

In relation to step 2, the main researcher should be present to support the producers and to collect their perception about the artifacts based on the DCs. The material used in these steps is available in a technical report (Lopes et al., 2021). Regarding step 3, the participants answered the following questions on the electronic questionnaire:

1. What is your perception about this artifact as a means of communication?

2. Tell us about your perception regarding the communication via artifact.

About the step 5, we used the following questions to collect the consumer's perceptions about the software artifacts:

1. During the software development, did you notice any inconsistent information regards the team knowledge about the software? – based on DC1;

2. About the quantity of information, is there the lack of information or excessive information?- based on DC2;

3. Is all information in the artifacts relevant to software development? Please, tell us your perception – based on DC3;

4. It was difficult to understand any information in the artifacts? – based on DC4;

With the execution of these steps, the DCs can help practitioners to understand aspects that need improvements in the informational content of the artifacts. These improvements can lead a better communication via artifacts.

4.2.3 Study 2: Results

Firstly, the producers analyzed the artifacts with the support of the DCs and we analyzed the types of defects related to risks identified. After this, analyzed the participants' answers.

Analysis of Software Defects Related to Risks of Communication Failure. The main risks of miscommunication are in the use cases. Such risks are related to the lack of updating of some information, identified with the support of DC1, and the excess of information, identified with DC2. Figure 7 presents a characterization of the identified risks.

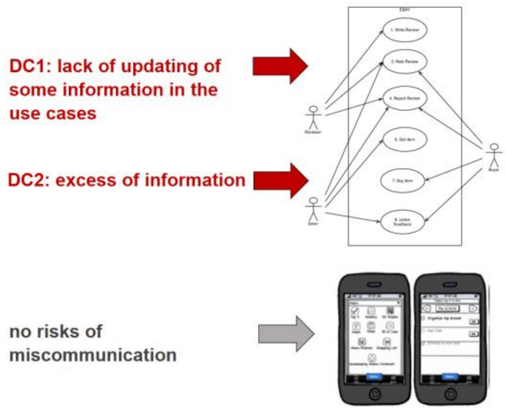
DC1: lack of updating of some information in the use cases
DC2: excess of information
no risks of miscommunication

Figure 7. Analysis of artifacts based on DCs

Regarding defects related to the risks of miscommunication, we have identified:

- **lack of updating of some information in the use cases** – *Inconsistence* defect: the lack of updating led to inconsistent information in the artifact; and *Extraneous Information* defect: information not needed in the artifact.
- **excess of information -** *Ambiguity* defect: the excess of information promotes different interpretations.

Analysis of Communication via Team Artifact. We used open coding (Strauss and Corbin, 1998) to understand the team's communication via artifact and how the DCs can support the improvement of this type of communication. We applied the coding based on the answer of producers and consumers about the artifacts' content.

When analyzing the team's communication through the artifacts, we identified characteristics in the informational content that affected the communication. We noted that consumers had adopted mockups more to support their activities compared to use cases. The team PO, one of the producers of the artifacts, and the consumers mentioned:

"*Perhaps I put more information in the mockups than necessary and it led the team to not consult the use cases* (Systems Analyst)".

"*There was an excess of information in the documentation. So many details generated several differences in the documentation for implementation and other minimal details that did not affect the system's functionality itself... With the use of the mockups, it was easier to understand the user's needs, and so the doubts that I had about the functioning of the system were resolved* (Developer 11)".

"*With the mockups, half of the system's functionalities were well defined, with only the business rules missing, which could not be modeled visually* (Developer 12)".

The DCs indicated that <u>consumers adopted more mockups as support in their activities than the use cases due to the excess of information in the use cases</u> (with the support from DC2), also cited by Developer 11. Additionally, there was an **outdated use case** (with the support from DC1), generating a negative impact on communication via artifact, as observed by one of the consumers:

"*Throughout the development, I believe that the artifacts have become outdated in relation to the needs of users and the implementation of the system* (Developer 4)".

Regarding the communication via this team's artifact, one of the producers reflected on their communication based on the DCs and he believes there was a <u>lack of another artifact that supports the understanding of the user's interaction with the system</u> (DC 2):

"*The artifacts contain the necessary information that the team needs to understand the problem. However, there are some limitations and information that cannot be transmitted in the artifacts. For example, the 'disposable mockup' presents only an idea of what the interface with the possible fields of the system will look like, but it does not present how it will be done, or even the user's interaction with the system* (Designer)".

With the results of this study, we noticed miscommunication via artifact identified with the support of the DCs. The DCs were able to support the producers to make improvements in the artifacts, enabling better communication via artifact.

4.2.4 Threats to Validity of Study 2

The threats related to this study are discussed below with the classification of threats to validity presented by Wohlin et al. (2012):

Internal validity. The main threat to internal validity was the sharing of developers' perceptions of the artifacts. To mitigate this threat, we sent an electronic questionnaire to each participant to answer their perception individually. However, this does not eliminate the possibility of communication between the participants.

External validity. Regarding the artifacts evaluated in this study, it is not possible to state that they represent all the artifacts that support communication. Additionally, these artifacts were modeled for just one software project.

Construct validity. We identified the threat of the participant providing answers that do not reflect reality but rather personal expectations regarding the artifacts. To mitigate this threat, we informed the participants that the experiment did not provide any kind of personal or project assessment but rather as an assessment of the use of artifacts in support of communication.

Conclusion validity. There is a limitation in the representativeness of the results, this being a known problem in experimental studies of Software Engineering (Fernandez et al., 2012). The results obtained in this study may not be reproduced in other software artifacts that support the understanding of those involved in the production of the systems.

4.3 Lessons Learned

These studies helped us to understand different aspects of the DCs from the practitioners' perception. About study 1, we described our lessons learned below.

- *Disagreements about the ease of use of the DCs show the need to create material that supports application of each directive* – although most participants agree that DCs are easy to use, we noticed some disagreements about this. The DCs are general instructions that supports the 'reflection' of producers about their communication via artifact and there are no specific steps for that. However, to support producers employ the directives, a material that indicates some reflection points would be interesting. Such material can be created based on common scenarios noticed in both studies presented in our paper.
- *Perceived usefulness by practitioners who act as consumers indicate that our proposal can support the communication of the artifact* –the usefulness perceived by practitioners who work as developers indicated that our proposal supports mutual understanding between producers and consumers, since such participants may had experienced such a scenario.

About Study 2, we noticed that DCs supported practitioners in the evaluation of artifacts already used by a software team. We had the following lessons learned from our proposal in this study:

- *Consumers' perceptions during evaluation of artifacts improve this type of communication* – Regarding the employ of DCs in the evaluation of artifacts already used by software teams, both producers and consumers can do that, providing a contrast about communication via artifact in a team. Such practice supports continuous improvements in this type of communication.
- *Material that supports producers to adopt the DCs in software projects* – we designed the initial proposal of the DCs to apply them in the production of artifacts, but we noticed the potential of the directives evaluate artifacts already used by teams. Aims to help software engineers to adopt the DCs in their projects, it would be interesting the development of procedures that indicate the main steps to apply the DCs.

The next section presents the proposal of the materials prepared to support software engineers adopt the DCs in their projects. We created such proposals based in our lessons learning.

5 Proposal to Support the Application of the DCs in Software Projects

Regarding the DCs, each directive aims to provide a general indication of how artifacts can be expressed by their producers about their communication, so the risks of miscommunication are mitigated. Regarding the adoption of the DCs to support improving the communicability of a software artifact, we observed two contexts in which artifacts are used: (1) when they are already being used by a team during the execution of a project, and (2) before being used by the team when the project is in its initial stages.

For the context above, we created procedures to facilitate practitioners who wish to adopt the DCs in their projects. Figure 8 presents a procedure to be followed by practitioners who wish to adopt the DCs to identify opportunities for improvements in the artifacts. This procedure is suggested for teams that have started creating artifacts without the support of our proposal, but they would like to adopt it in the artifacts, as noted in the second study presented in this paper, such as:

1. **Communication intent** - practitioners should reflect on their communication intent based on the questions: "*Can the consumer understand the artifacts' content? Can the consumer achieve its goals?*", like developers and testers, and "*What content should be addressed about the domain of the problem/solution of the system in the artifact?*", as the tasks that a user can perform in the system.

2. **Use of the DCs in the artifacts' content** - use of the DCs to identify risks that caused miscommunication. To facilitate the use of the DCs, we prepared a checklist, presented later in the text. At this stage, producers and consumers can carry out the evaluation for a better understanding of the necessary improvements.

3. **Availability of artifacts** - with the improvements made, producers make the artifacts available to consumers, as this also affects communication via artifacts, such as e-mail or repository used by the team.

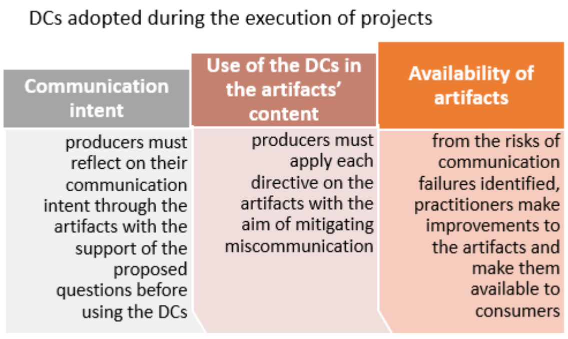
DCs adopted during the execution of projects
Communication intent
producers must reflect on their communication intent through the artifacts with the support of the proposed questions before using the DCs
Use of the DCs in the artifacts' content
producers must apply each directive on the artifacts with the aim of mitigating miscommunication
Availability of artifacts
from the risks of communication failures identified, practitioners make improvements to the artifacts and make them available to consumers

Figure 8. Use of the DCs during execution of projects

For the second context, Figure 9 shows the procedure that practitioners can adopt when using the DCs before the production of the artifacts. Each step to be followed in the procedure is described below. With the DCs applied to the artifacts before their consumption, the risks that caused miscommunication can be reduced.

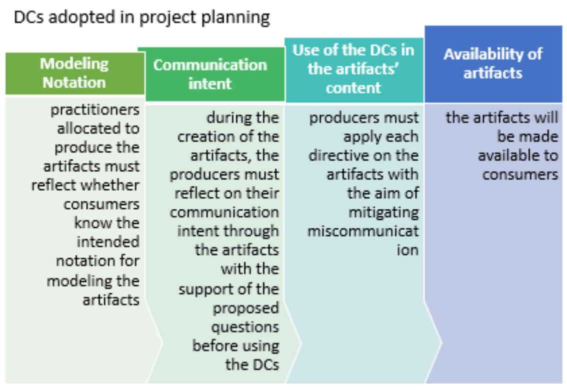
DCs adopted in project planning
Modeling Notation
practitioners allocated to produce the artifacts must reflect whether consumers know the intended notation for modeling the artifacts
Communication intent
during the creation of the artifacts, the producers must reflect on their communication intent through the artifacts with the support of the proposed questions before using the DCs
Use of the DCs in the artifacts' content
producers must apply each directive on the artifacts with the aim of mitigating miscommunicat ion
Availability of artifacts
the artifacts will be made available to consumers

Figure 9. Adoption of the DCs in project planning

1. **Modeling notation** - it is important for producers to reflect on the notation that will be adopted when modeling the artifacts to represent aspects of the software. Additionally, it is important for producers to reflect on whether such notation is known to consumers. This step was not considered in the first context because the team already has the artifacts established to represent the solutions modeled for the software.

2. **Communication intent** - similarly to the first context, practitioners must reflect on their intention to communicate based on the questions proposed to use with the DCs.

3. **Use of the DCs in the artifacts' content** - Use of the DCs to reflect on producers' communication intent. The checklist also supports this reflection.

4. **Availability of artifacts** - producers should reflect on the best means of communication that artifacts should be made available to consumers, as it can affect communication via artifacts, such as e-mail or repository.

In addition to the procedures, we also developed checklists that can facilitate the application of the DCs in the artifacts investigated in our research, such as UML use case and mockups. Table 7 presents the checklist for mockups and Table 8 presents the checklist for UML use case.

Table 7. Checklist based on DCs for Mockups

DCs	ITEM DESCRIPTION
DC1	Is there information in the mockups that are outside the problem domain? If so, remove that information
	Is there outdated information in the mockups? If so, update them
DC2	Are all requirements represented in the mockups? If not, design mockups with such information
	Are all alternative paths represented in the mockups? If not, enter this information in the mockups
	In general, is the amount of information in the mockups sufficient for the team to understand the system? If not, enter the required amount of information
	Is there an excess of information? If so, if this excess is unnecessary for understanding the system, remove it from the mockups
DC3	Is the order of the screens organized in such a way that the team better understands them? If not, arrange this sequence
DC4	Are the screen names clear in relation to their purpose? If not, clarify the names of the screens
	In mockups, are there any terms that are unknown to consumers? If so, please clarify such terms
	In mockups, is there any ambiguous information? If so, please clarify this information
	Is information used to obtain implicit interpretation by the team? If so, reflect on whether such information should be expressed explicitly to avoid multiple interpretations

Table 8. Checklist based on DCs for Use Cases

DCs	ITEM DESCRIPTION
DC1	Is there information in the use cases that are outside the problem domain? If so, remove that information
	Is there outdated information in the use cases? If so, update this information
DC2	Are all relationships between use cases represented in the diagram? If not, enter such relationships
	Are all use cases represented in the diagram? If not, insert such use cases
	In use cases specification, are all actors involved represented? If not, insert such actors
	When specifying a use case, are all flows represented? If not, enter the necessary flows
	When specifying a use case, are all business rules represented? If not, insert the necessary rules
	Is there an excess of information? If so, if this excess is unnecessary for understanding the system, remove it from the mockups
DC3	Are the use cases organized in the diagram logically? If not, organize the use cases
	Are the actors organized concerning the use cases in the diagram? If not, organize the actors in the diagram
	Is the sequence of information in each use case specification logically organized? If not, organize this information
DC4	Are the names of the use cases clear concerning their purpose? If not, clarify the names of the use cases
	Are the names of the actors clear concerning their purpose? If not, clarify the actors
	In the use case specification, are there any terms that are unknown to consumers? If so, please clarify such terms
	When specifying a use case, is there any ambiguous information? If so, please clarify this information

These checklists contain questions based on common artifact scenarios that have risks of miscommunication. However, we emphasize that DCs help practitioners to reflect on the artifacts and checklists support the identification of specific risks. In this way, checklists should be used together with the DCs.

6 Discussion

We carried out studies with the objective of transferring the DCs to the software industry. In the first study, conducted to answer RQ1 (*Do practitioners perceive the DCs as support in improving the quality of artifacts?*), we noticed that the directives supported the participants' reflection on the communication via UML use cases. This allowed reducing possible inconsistencies in the development of the explored artifact. It was possible to obtain evidence that the DCs can contribute to improving the artifacts' quality, since the DCs supported reducing incorrect information. This can reduce costs during software development, as defects discovered during the software development process increase costs due to the correction of such defects.

The second study conducted aims to understanding the feasibility of the DCs to support the improvements in the communicability of software artifacts used by the team to answer RQ2 (*Is the DC application by producers feasible in development teams?*). The use of the DCs showed the main aspects that need improvements, since they negatively affected the communication between producers and consumers of these artifacts. The results of this study demonstrated the benefit of using the DCs, as the problems identified in the informational content of the artifacts can be fixed.

Both studies showed evidence to transfer the DCs to the industry. In addition, such studies help us to obtain insights for the development of proposals that facilitate the adoption of the DCs in software projects.

7 Final Considerations

This paper presented research carried out with the aim of transferring the DCs to the software industry. We explored the DCs concerning the practitioners' perception about the DCs as supports in the reflexion of them as producers, in a first study, and a specific software development team about the risks in software artifacts that caused miscommunication with supports of the DCs, in the second study.

From the first study, the results showed that the DCs supported participants to reflect on the system, reducing possible inconsistencies in the development of the explored artifact, a UML use case. The DCs also promoted the participants' reflection on their communication with the others involved in the software development. The reduction of miscommunication also reduces the introduction of defects, as a consistent mapping between risks of miscommunication and software defects has been perceived.

In the second study, from the risks identified in the artifacts used by the software development team, producers made improvements in the artifacts. With that, software development teams will be able to adopt the DCs in their projects to improve communication via artifact. Additionally, most of the participants' perception about the DCs were positive.

From the studies results, we noticed the need to define some artifacts so that practitioners can use our proposal in their projects. We presented in this paper two procedures that facilitate the use of the DCs in software projects. Besides, for the better use of each directive, we have proposed checklists. We believe that practitioners interested in adopting our proposal can use them. Regarding the use of the DCs in these studies, it is possible to infer that they were considered feasible for the software industry. The new studies in the context of software projects can provide more evidence on the application of the DCs to support producers about their communication, aiming to reduce the risks of miscommunication.

As future work, we intend to carry out an observational study in different software development teams using our proposal. In this future study, the teams will use our artifacts, process and checklists, proposed in this paper, including the evaluation of artifacts developed in the early stages of the software development process not explored in our studies. In addition, we intend to investigate the software engineers perceptions about to include the DCs as part of the company's culture related to creation of artifacts used as means of communication.

Acknowledgements

We are grateful for the financial support from CAPES (financing code 001), CNPq (311494/2017-0 e 204081/2018-1/PDE) and FAPEAM (062.00150/2020).

References

Brambilla, M., & Fraternali, P. (2014). *Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML*. Morgan Kaufmann.

Bordin, S., & De Angeli, A. (2016). Focal Points for a more User-Centered Agile Development. *International Conference on Agile Software Development*, 3-15.

Corbin, J., & Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

De Souza, C. S. (2005). *The semiotic engineering of human-computer interaction*. MIT press.

De Souza, C. S., Cerqueira, R. D. G., Afonso, L. M., Brandão, R. D. M., & Ferreira, J. S. J. (2016). *Software Developers as Users*. Cham: Springer International Publishing.

Freire, E. S. S., Oliveira, G. C., & de Sousa Gomes, M. E. (2018). Analysis of open-source CASE tools for supporting software modeling process with UML. In *Proceedings of the 17th Brazilian Symposium on Software Quality*, 51-60.

Granda, M. F., Condori-Fernández, N., Vos, T. E., & Pastor, O. (2015). What do we know about the defect types detected in conceptual models? In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, 88-99.

Grice, Herbert P. Logic and conversation. *Speech acts*. Brill, 1975. 41-58.

Jakobson, R. (1960). Linguistics and poetics. In *Style in language*. MA: MIT Press, 350-377.

Käfer, V. (2017). Summarizing software engineering communication artifacts from different sources. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 1038-1041.

Likert, R. (1932). A Technique for the Measurement of Attitudes. *Archives of Psychology*, 144 (55), 7-10.

Lopes, A., Oliveira, E., Conte, T., & de Souza, C. S. (2019a). Directives of communicability: towards better communication through software models. In *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 45-48.

Lopes, A., Conte, T., & de Souza, C. S. (2019b). Reducing the risks of communication failures through software models. In *Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems*, 1-10.

Lopes, A., Conte, T., & de Souza, C. S. (2020). Exploring the Directives of Communicability for Improving the Quality of Software Artifacts. In *Proceedings of the XIX Brazilian Symposium on Software Quality* (SBQS'20), 10 pages.

Lopes, A., Conte, T., & de Souza, C. S. (2021). Directives of Communicability: Towards Software Development Teams. USES Research Group Technical Report, TR-USES-2021-01. https://doi.org/10.6084/m9.figshare.15057984.v2

Marangunić, N., & Granić, A. (2015). Technology acceptance model: a literature review from 1986 to 2013. *Universal access in the information society*, 14(1), 81-95.

OMG. (2011). Business process model and notation (BPMN) version 2.0. *Object Management Group*, *1*(4).

OMG. (2015). Unified Modeling Language TM (UML) Version 2.5.

Petre, M. (2013). UML in practice. In *Proceedings of the 2013 International Conference on Software Engineering* (ICSE 2013), 722-731.

Khare, R., & Taylor, R. N. (2004). Extending the representational state transfer (rest) architectural style for decentralized systems. In *Proceedings of the 26th International Conference on Software Engineering*, 428-437.

Sebe, N. (2010). Human-centered computing. In *Handbook of ambient intelligence and smart environments*, Springer, Boston, MA, 349-370.

Schoonewille, H. H., Heijstek, W., Chaudron, M. R., & Kühne, T. (2011). A cognitive perspective on developer comprehension of software design documentation. In *Proceedings of the 29th ACM international conference on Design of communication*, 211-218.

Tilley, S. (2009). Documenting software systems with views VI: lessons learned from 15 years of research & practice. In *Proceedings of the 27th ACM international conference on Design of communication*, 239-244.

Venkatesh, V., & Bala, H. (2008). Technology acceptance model 3 and a research agenda on interventions. *Decision sciences*, *39*(2), 273-315.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.