Federal University of Minas Gerais
Computer Science Department

Final Report for *Projeto Orientado em Computação II*

Development of a mobile application to support the use of the MoLVERIC technique for MoLIC diagram inspection

Technological Research

por Isabel Elise Silva Duque

Advisor: Raquel Oliveira Prates
Co-advisor: Angélica Beatriz Castro Guimarães

Belo Horizonte
Minas Gerais
27 Jan 2025

Federal University of Minas Gerais
Computer Science Department

Isabel Elise Silva Duque

Development of a mobile application to support the use of the MoLVERIC technique for MoLIC diagram inspection

Final Report for the subject *Projeto Orientado em Computação II* to conclude the Computer Science course at Federal University of Minas Gerais. Prepared under the guidance of Professors Raquel Prates and Angélica Guimarães

Belo Horizonte
Minas Gerais
27 Jan 2025

Abstract

This technical report describes the development and operation of a mobile application designed to support the use of the MoLVERIC technique, a gamified method for inspection of MoLIC diagrams that model interaction between users and software interfaces. The system goal is to enhance user experience by incorporating additional gamification elements and making its use more practical for students and professionals in the Human-Computer Interaction field.

Contents

1 Introduction

Students of the *Interção Humano-Computador* subject at the University of Minas Gerais are presented with a series of concepts and activities relevant to the field of Human-Computer Interaction. During the course, they are asked to conceptualize and implement an interactive prototype for an innovative application on a specific theme that varies each semester. To achieve this goal, they undergo a series of steps while learning and practicing several theories that support the structured approach to software product design.

After committing to a concise idea for the application, the students must construct a MoLIC diagram illustrating all the possible routes that a user may take while interacting with the application. MoLIC (Modeling Language for Interaction as Conversation) [1] corresponds to a language used for interaction modeling based on the theory of Semiotic Engineering, which studies focus on communication between designers and users in the moment of interaction [2].

To check if the MoLIC notations were correctly used and the interaction scenario was presented consistently, the students are then required to apply the MoLVERIC technique to assess the amount of errors present in their MoLIC diagrams. This method works by employing a set of cards that refer to elements that participate in the structure of MoLIC diagrams [3]. Each card describes a list of defects that may or may not be present in the diagram, along with a score granted to the inspector by correctly identifying a defect of that card. Given a set of physical cards, the students inspect each one, manually annotating the defects located and registering the cumulative score.

The goal of the system presented in this report is to assist HCI students, as of the subject *Interção Humano-Computador* at Federal University of Minas Gerais, along with professionals in the field, in the process of inspecting MoLIC diagrams through the MoLEVRIC technique.

The result consists of a mobile application where users have access to all 19 MoLVERIC cards by two different inspection methods, a guided and a loose one, tied to an interface designed to provide an easy and practical use of the technique. Along with a progress bar, with which users can follow the progress in the inspection, it provides an automatic calculation of the inspection score and generation of the inspection report that can be viewed in two different structures and exported into a PDF file.

2 Methods

The application was implemented using React Native, a framework for building native applications using React [4]. The tool was chosen taking into account the author's experience with React in developing web interfaces. As

recommended by the official React Native website [4], the Expo framework was also used, along with its provided libraries as Expo Router, for example, to support page navigation.

Coding was carried out using Typescript along with the Visual Studio Code IDE (VSCode). The task was made easier thanks to IntelliSense, a resource provided by VSCode that shows code completion, hover information, and signature help, allowing code to be written in a quicker and more correct manner [5].

Following the work developed in *Projeto Orientado em Computação I* [6], the interface and functionality implementation were based on the ones simulated by the interactive prototype previously built for the application.

The adopted strategy was to implement the core components first, and then build all the ones for which necessity was being observed. That way, a bottom-up approach was utilized, as the application pages' layout and navigation between them were one of the last segments implemented when most of their content was already resolved.

2.1 Device emulation

In order to view what was being developed and check if the app would function as expected, an Android device was emulated using Android Emulator, a device emulation tool that can be used to debug and test applications in an Android runtime [7].

Figure 1 shows the emulated device, named MEDIUM_PHONE_API_35, with screen dimensions 1800×2400.

Since React Native is able to translate React components into code that runs on different platforms, some of the app testing was done in the browser. Although some minor interface differences between platforms were observed, the web execution presented itself as lighter and faster, making it a good option for quick verifications.

2.2 Local data storage

Proper application functioning required data to be persistent upon closure and re-opening. In order to achieve that, methods were employed to assure the inspection progress could be recovered if the user decided to stop using the app and later decides to continue where they left of.

The react_native_mmkv [8] framework was used to store key-value pairs when application is run on a mobile device. For the browser environment, the localStorage property was employed, which can be used to access the current origin's local storage space [9].

Stored values include information and state for marked and unmarked defects, the cards already inspected in the loose inspection, the current point

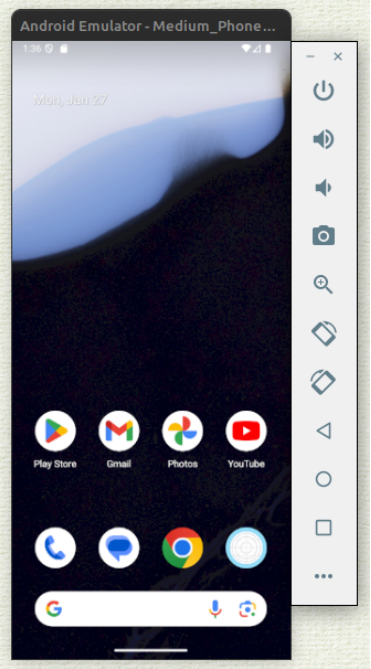

Figure 1: Android emulator running

in the guided inspection order, and the inspection state (whether there is a inspection in progress or not).

2.3 User experience survey

A small and informal user survey was conducted in parallel with the system development. The goal was to collect the opinion of *Interação Humano-Computador* students about the non-functional prototype in order to identify usability issues that could be addressed in a first functional version of the application as well as in future related works.

A succinct google forms questionnaire was designed and released in a message group with members being students or former students that already had contact with MoLIC and MoLVERIC concepts.

3 Result

Figure 2 shows the general page and navigation structure for the application. The code, along with its commit history, may be viewed in its entirety at `https://github.com/isabel-elise/molveric-app`.

The quantity of respondents to the user experience questionnaire was insufficient to warrant a robust statistical analysis. Therefore, the concluding subsection succinctly presets the outcomes of the survey.

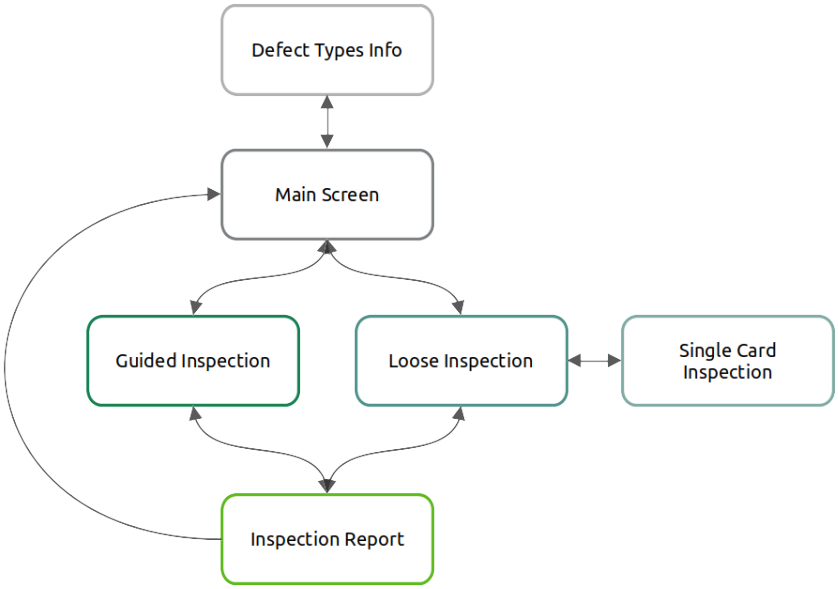

Figure 2: Page and navigation structure

3.1 App interaction flow

The following subsections describe functionalities and interaction flow a user may experience when using the system, highlighting the purpose of each segment in the goal of MoLIC diagram inspection.

Main Screen

The app starts on the Main Screen. As shown in the first screenshot in figure 3, there are different options presented in the form of 4 buttons.

The first button, *Inspeção Guiada*, takes the user to the Guided Inspection screen, which is described in detail in section 3.1. The second one, *Inspeção Livre*, takes the user to the Loose Inspection screen, described in detail in section 3.1. The third button is not available the first time the app is booted, but becomes available as soon as one of the inspection methods are selected.

The inspection progress is saved locally on the device (or browser, if the application is being executed on the web), which means its preserved even if the app is closed and then re-opened in a later moment. Because of that, the third button, *Nova Inspeção*, allows the user to request that the current progress be deleted, clearing all marked defects and inspected cards in the current inspection. Before performing the progress erasure, a dialog box

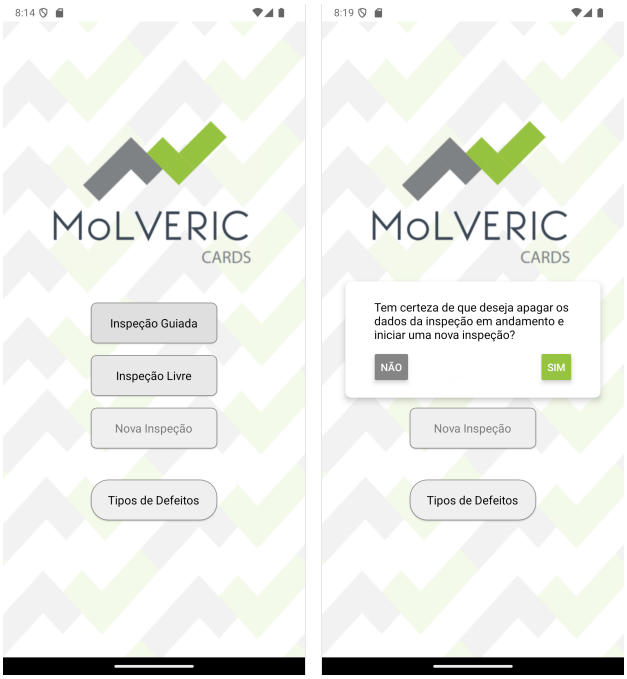

Figure 3: Screenshots of the app's home screen running on the emulated Android device

appears, as shown in the second screenshot of figure 3, asking the user for confirmation.

The fourth button, *Tipos de Defeitos*, takes the user to the Defect Types Info screen, which content in described in section 3.1.

Defect Types Info

Defect Types Info is an informative screen with no possible interactions other than returning to the Main Screen. It explains all of 5 MoLVERIC's defect types that each defect may be classified as.

As depicted in figure 4, it shows the defect types name, icon, and a short paragraph describing their meaning.

Guided Inspection

In guided inspection mode, the cards are presented, one at a time, following a specific order. As shown in figure 5, the screen layout consists of a card occupying most of the visible space and a progress bar at the bottom.

The user may choose to go forward or backward in the inspection by accessing the next or previous card in the order by tapping one of the arrows on the side. The progress bar updates accordingly, reflecting how many cards the user must inspect before reaching the end of inspection.

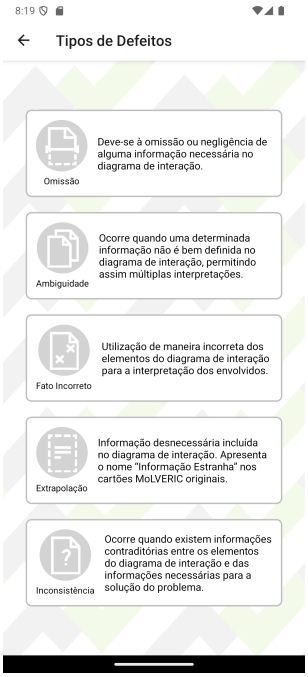

Figure 4: Screenshot of the app's Defect Type Info screen running on the emulated Android device

The user may go back to the Main Screen, but must view all the available cards in order to reach the screen in which the inspection report is generated, along with the calculated total score.

The in-card interactions are described in section 3.1 and allow for the marking of defects found on the current MoLIC diagram being inspected.

Loose Inspection

The loose inspection workflow consists of two separated screens, the one presented in figure 6 and the one described in section 3.1.

In the Loose Inspection screen, all the cards are displayed in a summarized visualization, showing only the referenced MoLIC element, its unique code, and the types of defects contained in the card. The user can then select the card they want to inspect by tapping its shortened form, prompting the Single Card Inspection Screen to appear.

The displayed cards may already be inspected or not. Only inspected cards highlight the types of the defects that were marked in that card.

The layout also provides a progress bar, as in the Guided Inspection screen, and a button for viewing the inspection report. In this inspection method, there's no need to view all the cards before accessing the screen in which the report in generated. If the user taps the button to check the

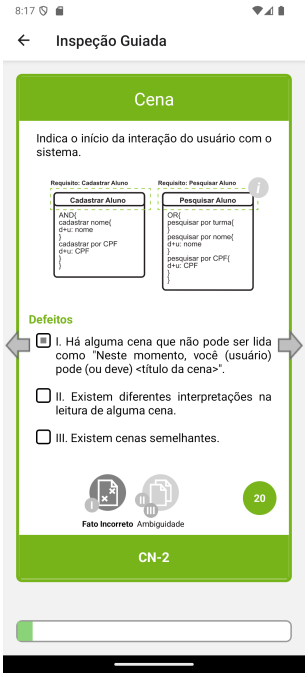

Figure 5: Screenshot of the app's Guided Inpection screen running on the emulated Android device

report without having inspected all cards, a dialog box will appear, as shown in the second screenshot of figure 6, warning that there are cards that were not inspected.

If the users confirms that they want to view the inspection report, they'll be taken to the Inspection Report screen, described in detail in section 3.1.

Single Card Inspection

In the Single Card Inspection screen, as shown in the first screenshot of figure 7, the user is presented with the MoLVERIC card selected in the previous Loose Inspection screen, a button for marking the card as inspected and the progress bar.

The card content includes its referenced MoLIC diagram element, a short description for the element (along with a i button that is present on some cards that show further information), an example picture of the card diagram structure focus, a list of related defects that may or may not be present in the diagram, and the types of the defects listed in the card displayed by name and icon.

Each listed defect is accompanied by a selection box that indicates if it is currently selected or not. In case an unselected defect is tapped, a dialog box appears, as shown in the second screenshot of figure 6, asking the user

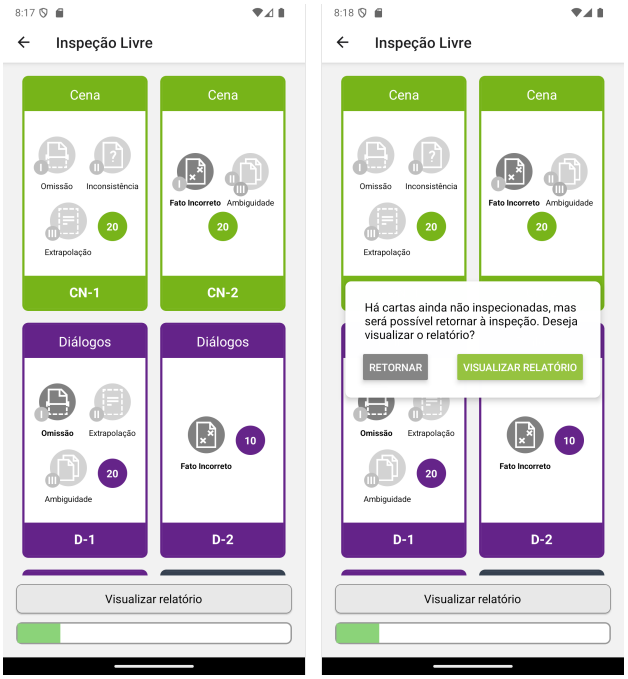

Figure 6: Screenshots of the app's Loose Inspection screen running on the emulated Android device

to specify the defect location on the diagram and provide an explanation for the marking, if necessary. As a defect is marked, its corresponding type on the bottom of the card darkens, indicating that a defect of that type is currently selected in that card.

After hitting *Concluído*, the defect marking information is stored temporarily in the current execution. Only when *Marcar carta como inspecionada* in pressed that the card state is stored locally and persists upon app closure. On Guided Inpection, however, all markings are directly stored, without the need to explicitly mark each card as inspected.

When a previously uninspected card is indicated as inspected, the progress bar reacts, reflecting an advance in the loose inspection progress.

Inspection Report

The Inspection Report screen shows all the defects marked during the inspection along with options for returning to the inspection, exporting the report, and returning to Main Screen, as depicted in the screenshots of figure 8.

The defects may be presented in two different report structures, one organized by each card element, shown in the first screenshot of figure 8, and the other one by defect type, shown in the third screenshot of the same

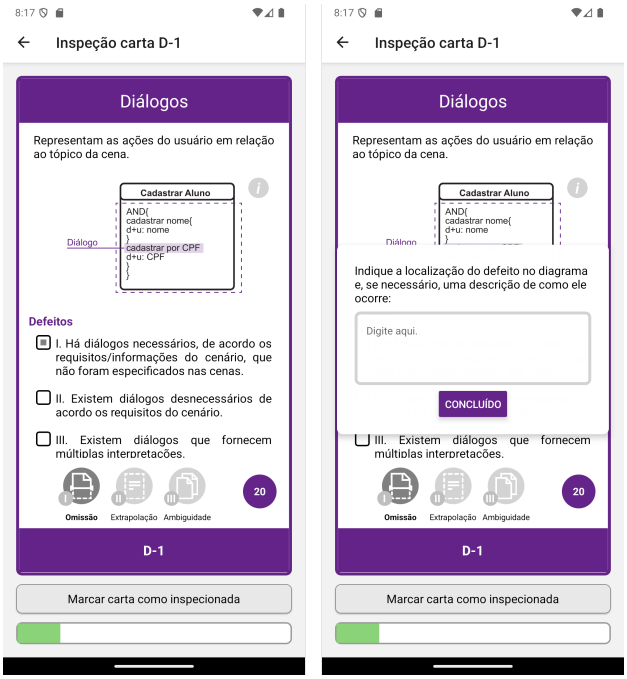

Figure 7: Screenshots of the app's Single Card Inspection screen running on the emulated Android device

figure. Each marked defect is shown with its description, respective card and type, and an icon that, when pressed, shows the defect location and/or explanation provided when the defect was marked, as shown in the second screenshot of figure 8.

By pressing the *Exportar relatório* button, a PDF file is generated considering the current selected report structure. Examples of generated inspection report documents are presented in appendices 1 and 2.

3.2 Survey outcome

Even with a low number of participants, it was possible to extract some interesting results of the survey, some of then being about ideas for enhancing the user experience by the addition of features on the app.

One participant suggested that a broader view of all cards could be implemented to allow for the inspection to occur without having to access each card at a time. Another suggestion was about helping the user to decide which inspection method they should select when they first boot the app.

In general, the students that took part on the survey found the application easy to operate for MoLIC diagram inspection using MoLVERIC and conveyed it would be productive to use in class.

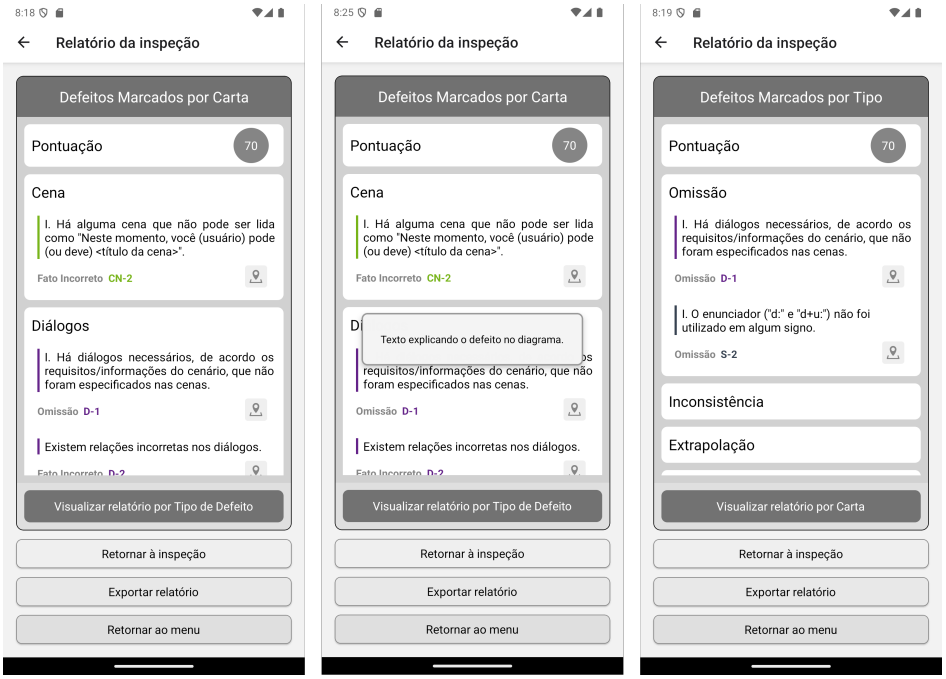

Figure 8: Screenshot of the app's Inspection Report screen running on the emulated Android device

4 Conclusion

The work developed for this project achieved the proposed goal, resulting in a mobile application designed for the MoLVERIC technique. Employing additional gamification features and enhancing the experience of the traditional MoLIC diagram inspection, the system provides the key features simulated by its former non-functional prototype created in *Projeto Orientado in Computação I*.

There is potential for students to be more involved in studying MoLIC diagrams by bringing the MoLEVRIC technique to a digital environment of daily use along with gamification elements. For HCI professionals, the system could serve as a practical tool for checking whether their interaction models comply with the MoLIC notation and respective interaction scenario requirements. The significance of these potentials lies in facilitating designers' utilization of a technique to detect errors early in development, thereby enhancing system quality and reducing production cost.

Future work may include adding more functionality to the app, such as a way to store multiple inspection progresses and an embedded view of the MoLIC diagram being inspected. The system quality would also benefit from a more formal user experience survey to identify usability problems to be solved in upcoming versions.

Bibliography

[1] Barbosa, Simone and Maíra Paula: *Designing and evaluating interaction as conversation: A modeling language based on semiotic engineering.* pages 16–33, June 2003, ISBN 978-3-540-20159-5.

[2] Souza, Clarisse de: *The Semiotic Engineering of Human-Computer Interaction*. February 2005, ISBN 9780262271363.

[3] Damian, Adriana, Anna Beatriz Marques, Tayana Conte, and Simone Barbosa: *Molveric: An inspection technique for molic diagrams.* July 2015.

[4] *React native*. `https://github.com/facebook/react-native`. Accessed 26 Jan. 2025.

[5] *Typescript in visual studio code*. `https://code.visualstudio.com/docs/languages/typescript`. Accessed 26 Jan. 2025.

[6] Duque, Isabel Elise Silva: *Protótipo de sistema para aplicação da técnica molveric para inspeção de diagramas molic*, 2024. `https://monografias.dcc.ufmg.br/monografia/prototipo-de-sistema-para-aplicacao-da-tecnica-molveric-para-inspecao-de-diagramas-molic/`, visited on 27 Jan. 2025.

[7] *Ferramentas do sdk*. `https://developer.android.com/tools?hl=pt-br#tools-emulator`. Accessed 27 Jan. 2025.

[8] *React native mmkv*. `https://github.com/mrousavy/react-native-mmkv`. Accessed 27 Jan. 2025.

[9] *Window: localstorage property*. `https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage`. Accessed 27 Jan. 2025.

1 Report exported by card element

Relatório da Inspeção

Defeitos marcados por Carta

Pontuação total: 60

Cena

CN-1 **Omissão**

I. Há algum objetivo do usuário não representado nos tópicos das cenas.

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.

Diálogos

D-1 **Ambiguidade**

III. Existem diálogos que fornecem múltiplas interpretações.

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.

Signos

Ponto de Abertura

Ponto de Encerramento

Acesso Ubíquo

Processo do Sistema

Fala de Transição e Recuperação da Ruptura

FTR-3 **Omissão**

I. Há falas que não utilizam o enunciador "u:" ou "d:".

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.

2 Report exported by defect type

Relatório da Inspeção

Defeitos marcados por Tipo

Pontuação total: 60

Omissão

CN-1 **Omissão**

I. Há algum objetivo do usuário não representado nos tópicos das cenas.

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.

FTR-3 **Omissão**

I. Há falas que não utilizam o enunciador "u:" ou "d:".

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.

Inconsistência

Extrapolação

Fato Incorreto

Ambiguidade

D-1 **Ambiguidade**

III. Existem diálogos que fornecem múltiplas interpretações.

Localização / Explicação do defeito

Texto de exemplo descrevendo a localização do defeito no diagrama.