UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Enhancing Decision Support Systems development with the EUL Framework applied in the Agricultural Sustainability Assessment domain

Wanner Martins de Menezes

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: ______________________

Wanner Martins de Menezes

Enhancing Decision Support Systems development with the EUL Framework applied in the Agricultural Sustainability Assessment domain

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Dilvan de Abreu Moreira

USP – São Carlos
December 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

M543e Menezes, Wanner Martins de
Enhancing Decision Support Systems development with the EUL Framework applied in the Agricultural Sustainability Assessment domain / Wanner Martins de Menezes; orientador Dilvan de Abreu Moreira. -- São Carlos, 2023.
87 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Decision Support System. 2. User Interface Generation. 3. User Interface Evaluation. 4. Ontology. 5. Domain Specific Language. I. Moreira, Dilvan de Abreu, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Wanner Martins de Menezes

Aprimorando o Desenvolvimento de Sistemas de Apoio à Decisão com o EUL Framework aplicado no domínio de Avaliação de Sustentabilidade Agrícola

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Dilvan de Abreu Moreira

USP – São Carlos
Dezembro de 2023

This work is dedicated to all kids from Brazilian public schools that will never stop believing in a better future.

ACKNOWLEDGEMENTS

This work would not be completed if I was not surrounded with spectacular people who helped me in lots of ways. I would like to thank my advisor Dilvan Moreira who guided me through this whole time. I also thank Kamila Rios, who contributed a lot to this research. I thank my mom Maria, my dad Paulo, and my sister Wanessa for being my support. And I specially thank Caio Oliveira, Mariany Morais, Thiago Rubio, Alex Pinto, Gustavo França, Leonardo Yves, and Jonathas Alves, people who helped to make this process a lot easier.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. And also by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

ABSTRACT

MENEZES, W. M. **Enhancing Decision Support Systems development with the EUL Framework applied in the Agricultural Sustainability Assessment domain**. 2023. 87 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Decision Support Systems (DSSs) play a vital role in organizing and processing data and information to facilitate decision-making in specific domains within organizations. The development of DSSs traditionally requires close collaboration between software developers and domain experts. However, maintaining a dedicated development team can be financially burdensome for many projects, resulting in slow and costly development and maintenance processes that hinder the agile development of DSS. To address this challenge, our group created the EUL framework, which proposes the utilization of ontologies and scripts based on Domain-Specific Languages (DSL) to automatically generate web-based DSS. The ontologies serve to describe the knowledge domain, while the scripts define the operations to be executed on user-supplied domain data. Developers and experts are only jointly concerned with these 2 artifacts. From them, the EUL framework automates the generation of the DSS, including its User Interface (UI).

This research analyses the quality of DSS generated using the EUL framework through its application in the agricultural domain, the SustenAgro DSS. SustenAgro creates sustainability assessments of sugarcane agricultural production systems located in south-central Brazil. In this work, the SustenAgro DSS was evaluated to validate its usability. This evaluation was conducted with different techniques. First, the Think Aloud method was used with four end users who were tasked with performing a set of predefined tasks. Additionally, these users provided feedback on system usability through the System Usability Scale (SUS). The results of the usability tests indicate that the generated UI for the DSS is well-structured, self-explanatory, and easy to use, demonstrating satisfactory functionality. Second, a qualitative evaluation was performed with a sustainability domain expert. He was able to modify the parameters of the DSS and generate new versions.

The findings highlight the successful application of the framework in the agricultural domain and validate its usability through user and expert evaluations. The automatic generation of DSS, using ontologies and DSL scripts, offers the potential for efficient and cost-effective development, enabling greater agility in the deployment and maintenance of DSS across various domains.

Keywords: Decision Support System, User Interface Evaluation, User Interface Generation, Domain Specific Language, Ontology, framework, automation.

RESUMO

MENEZES, W. M. **Aprimorando o Desenvolvimento de Sistemas de Apoio à Decisão com o EUL Framework aplicado no domínio de Avaliação de Sustentabilidade Agrícola**. 2023. 87 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Os Sistemas de Apoio à Decisão (DSS) desempenham um papel vital na organização e processamento de dados e informações para facilitar a tomada de decisões em domínios específicos dentro das organizações. O desenvolvimento de DSS tradicionalmente requer estreita colaboração entre desenvolvedores de software e especialistas de domínio. No entanto, manter uma equipe de desenvolvimento dedicada pode ser oneroso financeiramente para muitos projetos, resultando em processos de desenvolvimento e manutenção lentos e caros que dificultam o desenvolvimento ágil de DSS. Para enfrentar esse desafio, nosso grupo criou o framework EUL, que propõe a utilização de ontologias e scripts baseados em Domain-Specific Languages (DSL) para geração automática de DSS baseados na web. As ontologias servem para descrever o domínio do conhecimento, enquanto os scripts definem as operações a serem executadas nos dados do domínio fornecidos pelo usuário. Desenvolvedores e especialistas estão preocupados apenas com esses dois artefatos. A partir deles, o framework EUL automatiza a geração do DSS, incluindo sua User Interface (UI).

Esta pesquisa analisa a qualidade dos DSS gerados usando a estrutura EUL por meio de sua aplicação no domínio agrícola, o SustenAgro DSS. SustenAgro cria avaliações de sustentabilidade de sistemas de produção agrícola de cana-de-açúcar localizados no centro-sul do Brasil. Neste trabalho, o DSS SustenAgro foi avaliado para validar sua usabilidade. Essa avaliação foi realizada com diferentes técnicas. Primeiro, o método Think Aloud foi usado com quatro usuários finais encarregados de executar um conjunto de tarefas predefinidas. Além disso, esses usuários forneceram feedback sobre a usabilidade do sistema por meio da Escala de Usabilidade do Sistema (SUS). Os resultados dos testes de usabilidade indicam que a UI gerada para o DSS é bem estruturada, autoexplicativa e de fácil utilização, demonstrando funcionalidade satisfatória. Em segundo lugar, uma avaliação qualitativa foi realizada com um especialista no domínio da sustentabilidade. Ele foi capaz de modificar os parâmetros do DSS e gerar novas versões.

As descobertas destacam a aplicação bem-sucedida da estrutura no domínio agrícola e validam sua usabilidade por meio de avaliações de usuários e especialistas. A geração automática de DSS, usando ontologias e scripts DSL, oferece potencial para desenvolvimento eficiente e econômico, permitindo maior agilidade na implantação e manutenção de DSS em vários domínios.

Palavras-chave: Sistema de Apoio à Decisão, Avaliação de Interface do Usuário, Geração de Interface do Usuário, Linguagem Específica de Domínio, Ontologia, *framework*, automação.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

APTA	São Paulo Agency of Agribusiness Technology
DSL	Domain Specific Language
DSS	Decision Support System
Embrapa	Brazilian Agricultural Research Corporation Environmental Unity
EUL	End User Language
HCI	Human-Computer Interaction
HTML	Hypertext Markup Language
ICMC	Institute of Mathematical Sciences and Computation
IDE	Integrated Development Environment
IFML	Interaction Flow Modeling Language
IT	Information Technology
JSON	JavaScript Object Notation
MDA	Model-Driven Architecture
MDD	Model-Driven Development
NLP	Natural Language Processing
OWL	Web Ontology Language
SPA	Single Page Applications
SUS	System Usability Scale
SWA	Semantic Web Application
UI	User Interface
UML	Unified Modeling Language
USP	Universidade de São Paulo
UX	User Experience
W3C	World Wide Web Consortium
WebMedia	Brazilian Symposium on Multimedia and Web Systems

CONTENTS

CHAPTER 1

INTRODUCTION

Decision Support Systems (DSS) organize and process data and information to support decisions, judgments, and actions in an organization or a business in a specific domain. They use expert knowledge from their domains. Therefore, DSS allow less knowledgeable individuals to leverage seasoned experts' experience in their field.

However, creating a DSS for a technical domain can be difficult. Software developers usually do not understand the technical field if it is not related to Computer Science, and seasoned experts cannot formalize their domain knowledge in a computable model that can be automatically integrated into the system. The developers must carry out the knowledge modeling process with immediate assistance from the domain experts with requirement-gathering techniques for a correct system implementation (SCHIUMA; GAVRILOVA; ANDREEVA, 2012). This results in a slow and expensive process, hindering the agile development and maintenance of a DSS.

1.1 The SustenAgro Method

The sugarcane production sector is extremely important to the state of São Paulo and the Brazilian economies because it is one of the main cultures produced in the country (JESUS *et al.*, 2019). For this reason, experts from the Brazilian Agricultural Research Corporation Environmental Unity (Embrapa) developed the SustenAgro method to assess sustainability in sugarcane production systems in the central-south region of Brazil. Sustainability experts validated this method (JESUS *et al.*, 2019). The method was validated, but it did not have a software implementation.

Through cooperation between our research group at the University of São Paulo (USP) and Embrapa Meio Ambiente, a web-based DSS prototype implementing the SustenAgro Method was developed and tested. This prototype was based on the Grails framework, using HTML forms, and implemented the basic functionalities needed to apply the method. It used an ontology to describe SustenAgro knowledge and a DSL to implement the method's operations. But it was a prototype specific to the SustenAgro method.

1.2 The EUL Framework

The SustenAgro early prototype led to the idea of developing a framework to automatically create web-based DSS from ontologies in the Web Ontology Language (OWL) and scripts in a Domain Specific Language (DSL). Ontologies are traditionally used in many domains, including in agriculture (ROUSSEY *et al.*, 2010), and can be used by experts to specify the knowledge needed by a DSS in a computer-readable format. More procedural aspects of the DSS can be implemented using DSL scripts. DSLs are easy-to-learn languages tailored to a specific domain.

The developed framework was named EUL, an acronym for End User Language, . It uses the Vue framework to implement DSS as Single-Page Applications (SPA). SPAs are web applications that interact with users by dynamically rewriting the current web page with data from the server. That makes the application feels more like a native app.

With EUL, domain experts still need the help of developers to set up the basic DSS. However, this task is much easier since most of the software is already written, and the domain knowledge is segregated into ontologies and scripts. Once the system runs, the domain experts can do basic system maintenance by themselves, editing the ontology and scripts to implement small changes. That can be done in a safe trial-and-error environment. Users can edit the ontology and scripts online and immediately generate a new DSS version. If they do not like the results, they can easily undo their changes and try again without the risk of breaking the system.

Domain experts will need the help of developers if they need more complex changes. For instance, if they need to refocus the DSS on a new domain or if, for some reason, they are not comfortable doing even small changes in the DSS's files. In both cases, developers will only need to understand and change the ontology and script files, reducing the work needed to implement changes.

SustenAgro was the first DSS written using the EUL framework. The Figure 44 shows the introduction page for the system created. The framework goal is to allow the creation of knowledge-based DSS where functionality is determined by and segregated in ontologies and script files to reduce the amount of code needed to implement and maintain each DSS. It is easy to show that SustenAgro's ontology and script files are much smaller than the many program files used by traditional DSS. But there is no gain if the smaller files lead to a bad DSS. How to show that the generated DSS, SustenAgro, was good enough to be used by its intended audience?

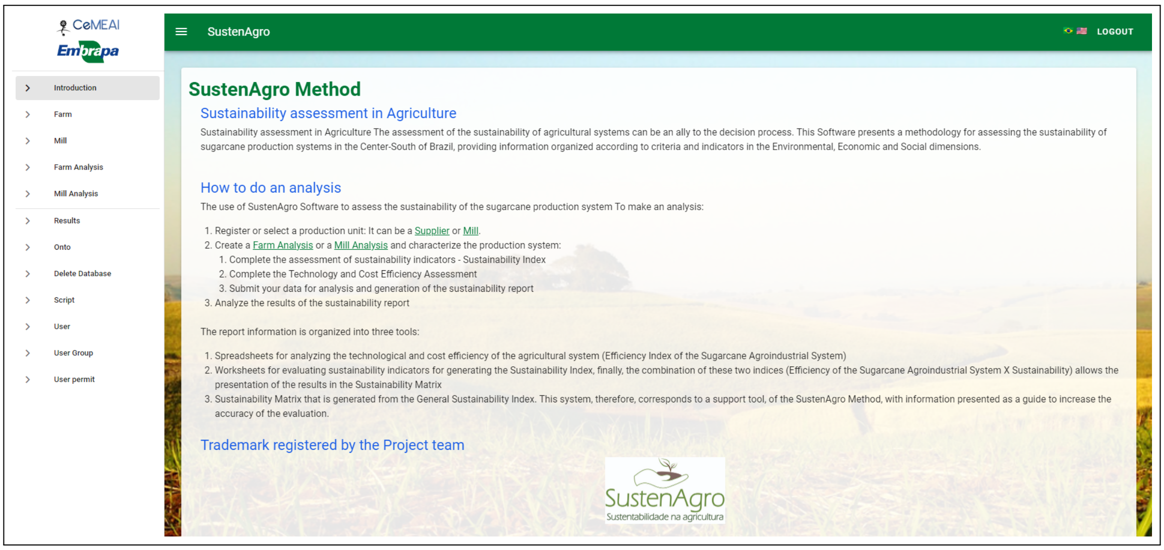


Figure 1 – Introduction page for the SustenAgro DSS.

1.3 Research Goals

This work is part of a wider effort to create a framework (EUL) that allows domain experts to generate data-oriented DSS using just an ontology and a small script file that experts can edit online.

To help with this broader goal, this work focused on testing the applicability of the framework, using a specific DSS (SustenAgro), to be able to answer the following research questions:

- Does the automatically generated EUL-based SustenAgro DSS have enough usability that final users could use it to provide information and execute the tasks needed for the DSS decision-making process to work?
- Using the EUL framework, will domain experts be able to modify the SustenAgro DSS, or parts of it, using ontologies and DSL as tools?

With these questions in mind, we get to the following hypothesis:

- The EUL Framework uses the data format, described by an ontology, to generate user interfaces to create, update, delete, and read these data. Given that SustenAgro, and many other DSS types are is data-oriented, the generated interface must have a satisfying usability.
- The EUL Framework uses an ontology and script file to generate a DSS automatically. Both artifacts can be easily edited online, and the DSS can be regenerated. Specific DSS knowledge is segregated into only 2 files, easing the task of maintaining the DSS. Some Domain experts can even edit some basic features of their DSS.

With all that in mind, the main goals proposed in this work were to fine-tune the last EUL-based Sustenagro DSS version, test its usability, and evaluate its interfaces to investigate if the EUL Framework can generate usable DSS. And as side goals, analyze if domain experts can modify the DSS themselves, and also fix small problems discovered during tests.

1.4 Results

Prior to assessing the usability of the Sustenagro DSS, it was imperative to rectify several issues identified during preliminary internal testing. Further enhancements were also made to improve the overall aesthetic appeal of the system.

Subsequently, a series of comprehensive experiments were executed with the DSS. The initial round of tests involved end users, individuals possessing knowledge in sustainable agriculture, as well as a domain expert specializing in this field. This was followed by a heuristic evaluation conducted by a panel of four User Interface (UI) experts.

Four technicians from Embrapa, with a keen interest in sustainability issues, were recruited as the DSS end users. The Think Aloud methodology, as described by Nielsen (NIELSEN, 1994), was initially used. This approach encourages users to verbalize their thoughts during their interaction with the system while performing a predefined set of tasks. This was followed by the application of the System Usability Scale (SUS) (BROOKE, 1996), aimed at quantifying the users' satisfaction regarding their system interaction. Additional feedback pertaining to the overall DSS was also solicited. As an outcome of this experiment, all 4 end users were successful in completing the tasks and provided favorable reviews about their interaction with the system. The SUS application yielded a score of 63.1 points, showing that improvements are needed, since the ideal range is of 68-70 points.

Apart from the end user tests, a qualitative usability test was carried out with a domain expert associated with the SustenAgro project. The methodologies employed with the end users, namely the Think Aloud and SUS, were replicated, albeit with tasks associated with system modifications. Following this, an interview was conducted to gather his views about the quality of interaction with SustenAgro and the automation of UI generation. The domain expert was successful in completing all tasks and was able to modify the system independently.

These user evaluations yielded constructive feedback regarding the project, highlighting the positive aspects of the generated UI, along with areas requiring usability enhancements, as stated by both domain experts and end users.

Lastly, a heuristic evaluation was performed by four UI experts to identify potential violations of heuristic guidelines. They enumerated all breached Nielsen's heuristics, noting their respective severity values, and proposed corrective measures. This information was then collated into a comprehensive table detailing all evaluation findings, identified issues, their descriptions, violated guidelines, violation severity, and remedial suggestions. It was observed that the majority of heuristic violations, a total of 58,92% were of low severity.

The findings of this investigation have been submitted to a prestigious symposiums in the fields of Human-Computer Interaction (HCI) and Web: the 29th Brazilian Symposium on Multimedia and Web Systems (WebMedia 2023). At the time of this dissertation submission, we are awaiting their decision regarding acceptance and subsequent publication.

Next, in Chapter 2, this work contextualizes the fundamental concepts of DSS, the model of developemnt used in this type of DSS, UI evaluation methods, and introduces the semantic web techniques used in the EUL framework. Chapter 3 shows the related work in automatic UI generation using formal languages. In Chapter 4, the EUL Framework is presented with the new SustenAgro DSS approach. In Chapter 5, we have the usability tests of the DSS UI. And finally, in Chapter 6, we conclude this work with a discussion of the results achieved and suggestions for further research.

CHAPTER

2

FUNDAMENTAL CONCEPTS

This section introduces the theoretical framework and fundamental concepts that will be used to develop this work. The definitions of Decision Support Systems, a methodology for user interface generation called Model-Driven Development (MDD), User Interface and its evaluations, ontologies from the Semantic Web field, and Domain-Specific Languages are discussed here.

2.1 Decision Support Systems

The primary goal of Decision Support Systems is to improve the decision-making process for individuals or groups. They integrate the knowledge of experienced experts with the data provided by their users. Mathematical models and methods are applied to these data to reveal relevant information and advice to support the decision-making process (TWEEDALE; PHILLIPS-WREN; JAIN, 2016).

The development of DSS is a challenging research field due to the complexities of modeling and understanding the various domain areas targeted by these systems. It requires continuous interaction between domain experts and software developers and extensive testing by experts.

A DSS consists of four main subsystems: Data Management, Model Management, Knowledge-Based, and User Interface. The Data Management subsystem manages the data used as information to allow making decisions in the Knowledge-Based subsystem. The Model Management component consists of models that assist decision-making. Knowledge-Based is the system's heart and manages the problem-solving process to generate the final solution. The User Interface allows the user to interact with the system to obtain information (FENU; MALLOCI, 2020).

In this work, we use ontologies to represent the Knowledge-Based subsystem and extract all data needed for Data Management. Ontologies are used to model aspects of the real world. They introduce vocabulary describing various parts of a domain being modeled and provide an explicit specification of the intended meaning of that vocabulary (HORROCKS, 2008).

Written in a DSL, scripts compose most parts of the Model Management that evaluate the data and assist in decision-making. Combined with the ontology, they automatically generate the User Interfaces to get the necessary data from users and show them the results.

The approach can validate the development time since that Ruiz, Serral and Snoeck (2019) state that UI development represents around 50% of the total application development time.

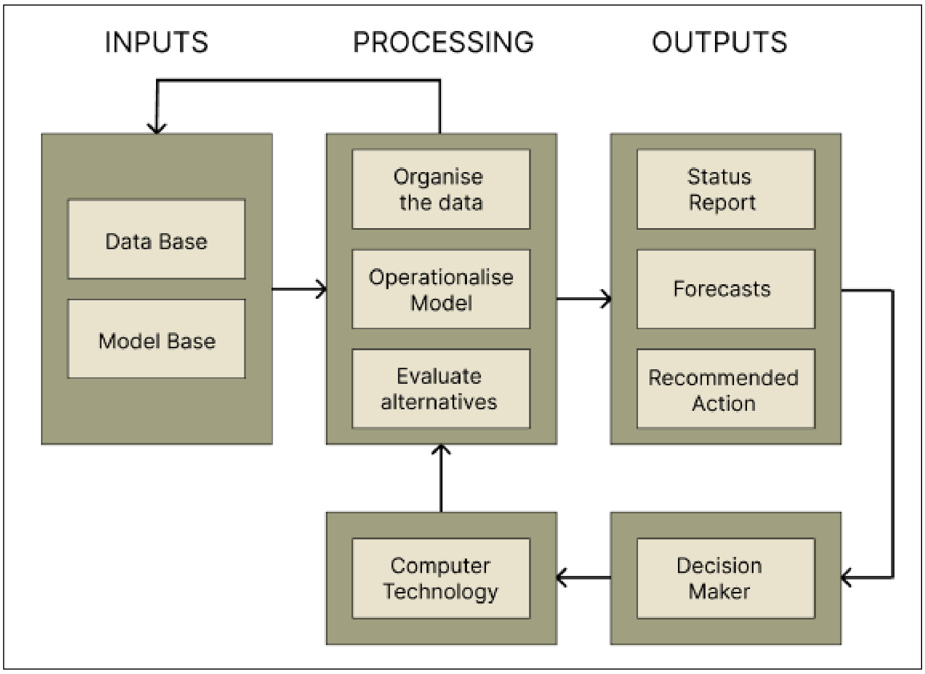
In DSS, the critical knowledge of domain experts must be designed in a model to allow communication and collaboration between the domain experts and the developers (EVANS, 2004). To do that, the DSS uses the architerure described in the following sub-items.

2.1.1 Decision Support System Architecture

The architecture provides information about how the components of the system are related and explains the external connections between them (SUAREZ, 2017a).

Based on Tweedale, Phillips-Wren and Jain (2016), the architecture can be represented by the components presented in Figure 2. It illustrates the process executed by the DSS, in which they receive input, realize the operation of this input, and then return the results analyzed by the decision-maker using computational technology.

Figure 2 – DSS architecture



Source: Adapted from Tweedale, Phillips-Wren and Jain (2016).

Inputs: composed of the data that will be processed and the knowledge models of the experts. The data are stored in a database, and the models are integrated inside the DSS. In this case they are the modeling of the ontologies and the scripts of the DSL.

Processing: composed of the methods and organizational methods, it realizes the operations of the data and generates the alternatives for evaluation for the domain experts, which provides the results of the system. They are the processing of the ontologies and the scripts to generate the outputs.

Outputs: results of the processing of the inputs, in which it is possible to compare the alternatives to make the decision. These outputs usually are in the form of reports, forecasts, and recommendations exhibited through a UI, enabling users to understand and interact with them.

Given the explanation of how a DSS is architected, they are also divided into different taxonomies.

2.1.2 Decision Support System Taxonomy

Power (2002) classified the DSS into five main categories, using as a criterion for this classification the main functionality of each system. These categories are as follows.

Data-Driven DSS: it has the processing and organization of a large amount of structured data as its goal.

Model-Driven DSS: it can manage models that show aspects of the real world. It uses tools, statistics, and analysis, making changing the model possible.

Knowledge-Driven DSS: They specialize in resolving problems through rules, facts, procedures, or domain knowledge that can allow solutions to decision-makers.

Document-Driven DSS: its specialty is to help the decision-makers to obtain, recover, classify, and organize non-structured documents.

Communications-Driven DSS: it is specialized in supporting communications, collaboration, and decision inside a group of people.

A DSS with this Knowledge-Driven approach was developed in this project, which is characterized by containing knowledge of a particular domain. They use knowledge bases that usually have rules to support concepts, knowledge management, and inferences. The SustenAgro is an ontology-based DSS that represents the knowledge of domain experts, and it is possible to determine, categorize, relate, and infer the knowledge to help the experts make decisions.

The Knowledge-Driven DSS developed in this work is generated using the MDD methodology. This way, we could satisfy all the needs and requirements of the system, and the goal was that the experts could maintain and create parts of their own system, managing the knowledge and all logic inside the tool.

2.2 Model-Driven Development

According to Kleppe *et al.* (2003), Model-Driven Development systems have the goal of helping solve some problems in the development of solutions. Some of them are modeling, knowledge reuse, productivity, maintenance and documentation, validation and optimization, portability, and interoperability.

In short, MDD aims to simplify and standardize the software development life cycle to automate various activities and tasks that comprehend this whole process (HAILPERN; TARR, 2006). The main idea of MDD is to use models in the creation process of a system and not only to serve as a guide to development tools and maintenance. They become part of the full software.

The MDD proposal aims to ensure no manual interaction directly with the system's source code. The development concentrates on a higher level of modeling, reducing the complexity of implementation. A tool is responsible for the code generation from models that are part of the system and the source code. (LUCRÉDIO, 2009).

In the case of the SustenAgro DSS, the EUL framework is the tool responsible for the code generation from models, in which we have the ontologies to do this work and the DSL scripts to implement the behavior of the system, both combine to generate the DSS UI. A core functionality of the EUL framework is the automatic generation of the DSS UI using the techniques mentioned in this approach.

2.3 User Interface

According to Nichols and Faulring (2005), interface generation has been used to separate the user interface from the application logic and also allows specific information about the user and the current situation to be incorporated into the design of the user interface.

Still following Nichols and Faulring (2005), model-based systems attempt to formally describe the tasks, data, and users that an application will have, and then use these formal models to guide the generation of the user interface.

Given that this approach presented on this work leads us to generate the web user interfaces, this work will present the evaluation of the quality of these automatic generated user interfaces.

2.3.1 Usability Evaluation

Usability as defined by Nielsen (1994), applies to every aspects of a system that someone interacts with. And to get to know the quality of the interaction. Nielsen (1994) also defines Usability Tests to determine the satisfaction of the experience with the UI.

To evaluate the DSS usability, there are some methods like Think Aloud, as proposed by Nielsen (1994). The Think Aloud method is a widely recognized usability testing technique that aims to understand how users interact with a product or system.

This method requires users to perform tasks while concurrently verbalizing their thoughts and actions, enabling researchers to comprehend the user's thought processes, decision-making strategies, and any challenges encountered during system interaction. By applying the Think Aloud method, researchers can gain valuable insights into the user experience, which can subsequently inform design and development decisions. With Tink Aloud, we can extract the most of feedback from the system, even with a lean set of users (NIELSEN, 1994).

In addition to the Think Aloud method, there is also the System Usability Scale form to be a complementary step to user interface evaluation. The SUS is a standardized questionnaire designed to measure the perceived usability of a product, system, or service (BROOKE, 1996).

The questionnaire comprises statements that participants rate on a 5-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree". These statements are specifically formulated to evaluate the overall satisfaction and ease of use associated with the product being tested (BROOKE, 1996).

SUS scores have demonstrated a high correlation with more comprehensive usability measures, and the scale is often employed in conjunction with other methods, such as heuristic evaluations (BROOKE, 1996). By incorporating both the Think Aloud method and SUS form, this study aimed to provide a robust and comprehensive assessment of the DSS usability for end users.

Furthermore, we conducted a heuristic evaluation following Nielsen's methodology (NIELSEN; MOLICH, 1990). It uses three to five User Interface and User Experience (UX) specialists to analyze a system's user interface using the 10 Nielsen Heuristics, a set of 10 general principles for user interface design:

- Visibility of system status.
- Match between the system and the real world.
- User control and freedom.
- Consistency and standards.
- Error prevention.

- Recognition rather than recall.
- Flexibility and efficiency of use.
- Aesthetic and minimalist design.
- Help users recognize, diagnose, and recover from errors.
- Help and documentation.

These heuristics are a set of guidelines for evaluating the usability of a user interface and became a widely recognized standard in the field of human-computer interaction.

If a specialist thinks the UI does not follow some of these guidelines, he determines a severity rating, on a scale from 0 to 4, for each guideline with problems. He also suggests a solution for each problem found. Afterward, a report is created as a consolidation file with all problem evaluations, containing the problem's location, its description, its severity rating, and suggestions for improvements from the specialists.

2.4 Ontologies

In computer science, ontologies began to be used at the beginning of the 90s as a form to organize a large basis of knowledge. They were used to construct large interoperable databases with a better structure (MOREIRA; ALVARENGA; OLIVEIRA, 2004).

Ontologies on Semantic Web describe a domain's concepts and their relation. The W3C (World Wide Web Consortium) define that ontologies should provide descriptions of concepts in various domains of interest, the relations between these concepts, and the properties they might have (BERNERS-LEE *et al.*, 2001).

The Semantic Web is an alternative to manage knowledge on the Internet in a semantic way, with a readable format understandable to humans and machines (BERNERS-LEE *et al.*, 2001). Its main objective is to structure data and information available online so that the data can be processed meaningfully, making a space where users and developers can work together. The W3C, as the leading organization for standardization, is responsible for defining the protocols that govern the Semantic Web.

According to Patel-Schneider (2005), the representation of an ontology is done through a logic of predicates and descriptive logic, using standards adopted by the community, such as the OWL. Standardizing the representation of ontologies is one of the main contributions of the Semantic Web.

2.4.1 Web Ontology Language

The Web Ontology Language is a Semantic Web language developed to provide a rich and more complex representation of things, groups of things, and relations between things (W3C, 2012).

The W3C recommends OWL for the online representation and sharing of ontologies. This language was planned for applications that process information content rather than organize information in nodes (MCGUINNESS, 2007).

In this work, ontologies in OWL describe the domain knowledge, in this case, the agricultural sustainability domain, and the script, in a DSL, describes the operations the DSS performs in the data to evaluate it in the context of the decision-making process. The data description, in the ontologies, guides the creation of the DSS's web UI.

2.5 Domain-Specific Language

A Domain Specific Language is a computational language used in a particular domain to perform specific tasks (FOWLER, 2005). A DSL is usually declarative and focuses only on the domain of a particular problem (DEURSEN; KLINT; VISSER, 2000).

A DSL's purpose is to simplify application development for a specific domain. However, it demands a more extensive comprehension of this domain (PIERONI *et al.*, 2014).

According to Ghosh (2011), the DSL have different classifications and usages. They can be classified as Domain-Specific Markup Languages, like HTML, that mark data with tags in the web pages domain. Domain-Specific Modeling Languages, like UML (Unified Modeling Language) , allow modeling to specify or model domains. And Domain-Specific Programming Languages, like the R Language, allow high-level programming in the statistics domain.

The DSL scripts used in the EUL framework that generates the SustenAgro DSS is a Domain-Specific Programming Language. In the context of this work, the DSL serves to customize the DSS's behavior, using ontologies to organize the domain knowledge.

A DSL can facilitate the definition of a DSS's behavior, enabling the experts to provide a solution compatible with the terms they use. This allows them to specify the DSS's behavior with a high degree of detail, sufficient to reduce or eliminate the involvement of software developers. In this way, domain experts can act as the developers of their own DSS.

In this chapter the theoretical base to the understanding of concepts and techniques of what this work will approach were shown.

In the next section, we bring the related work in automatic systems generation using formal languages.

CHAPTER

3

RELATED WORK

Decision Support Systems are vital applications that are usually targeted toward specific problems, making them more niche than general-use applications. Consequently, it is not surprising that a broad literature review yielded no specific works on the automatic generation of DSS systems or their interfaces. In light of this, a more general approach was adopted for this work, which focuses on the evaluation of SustenAgro DSS' generated UI and its usability. To this end, the SustenAgro DSS was compared to programs designed to generate systems of any kind or their interfaces from formal specifications.

To relate published works relevant to the topic of this research, a bibliographic search was carried out in Digital Library databases. Among them are IEEE Xplore, Springer, Scopus and Association for Computing Machinery (ACM). The search was guided by specific strings, such as:

- ("Automatic Generating") AND ("web UI") AND ("DSL")
- ("model-driven") AND ("Automatic Generating") AND ("UI")
- ("owl") AND ("automatic") AND ("GUI") AND ("generator")
- ("ontology based") AND ("user interface") AND ("automatic generator")

Besides that, to enrich more the results, the snowballing technique. This technique is based exclusively in the relations of citation and referencing by not depending neither exclusively in the keywords nor an specific database. This way, applying snowballing to the initial results, it is possible to discover relevant researches that could be omitted in conventional search. This approach guarantee a wider and robuster bibliographic review (SILVA, 2017).

Molina (2018) developed a web-based tool called Quid that defines UI composition abstractly using a DSL with minimal complexity. The UI specification is platform-independent, and Quid uses model transformations and code generation to create artifacts such as native web components or Angular code. However, the user needs some knowledge of UI design to personalize the UI the way they want, as well as a specification of the component type required.

It is distinguished from EUL because Quid uses a DSL to generate web components with no knowledge base or database. It generates only UI parts. The user needs some notion of UI design to personalize it the way he wants, and a specification of the component type is required. With EUL, the web components are automatically generated, with their specific types, based on what is set in the ontology, which serves as the knowledge base for the system. In summary, it requires that users have some knowledge of UI design. The EUL-based DSS does not require any such knowledge, the components are generated and organized using a pattern defined by the DSL and the ontology.

Laaz and Mbarki (2019) present OntoIFML, a Model-Driven Architecture (MDA) based approach that can be used as a plugin based on the Interaction Flow Modeling Language (IFML) to facilitate the development of web applications relying on the Semantic web paradigm. They explore the possibilities for automating the generation of domain ontologies and interfaces annotated by semantic vocabularies.

The OntoIFML focused on generating domain ontologies and annotated web pages, using the IFML tool to generate them. It is a different way to generate web interfaces with a semantic web approach.

Using the OntoIFML tool, the user interfaces of Semantic Web Applications (SWA) can be easily generated without having to know all the technical specifications of the execution platform. However, the work is restricted to one feature of the SWAs, which is semantic annotations.

On the other hand, the EUL framework is a more capable tool being able to generate DSS that are more than just web interfaces with semantic annotations. This work is restricted to semantic annotations, while the EUL uses the semantic web paradigm to represent the domain knowledge to generate the system.

Kolthoff (2019) developed a methodology based on Natural Language Processing (NLP) for supporting UI prototyping. It automatically translates natural language requirements into a formal DSL that describes the UI and its navigation schema. The DSL can be translated into prototypes in a target platform for user inspection. This work also shows how to develop UI with a DSL approach but cannot generate a complete knowledge base system as EUL can.

Soto, Mora and Riascos (2022) aimed to generate enough material for training and exploring a Machine Learning approach for automating web design and development. They presented the Web Generator, a software designed to provide web pages, designs, and content based on the Bootstrap frontend framework. The software can generate markup codes, screenshots, and labels for web elements.

Web Generator is built to generate web graphics UI datasets. It comprises images, section descriptions in JSON (JavaScript Object Notation) format, and HTML source with Bootstrap framework classes, allowing the generation of synthetic web UI and content quickly to consolidate datasets that can be used to train different algorithms.

The Web Generator is an excellent approach to creating general content that does not need any specific context. However, this is not the case when we need to develop a DSS based on particular knowledge to implement a decision-making process (as EUL does).

All these works, in one way or another, focus on UI specification aspects. In the EUL framework, users do not have to specify the UI. With EUL, the ontology and scripts define only the domain knowledge and operations. Then, the UI is generated to fit the data types present in the model from interactive or static web components.

Another approach to generating systems is Low-code development platforms. Many low-code platforms today focus on data-intensive web systems, such as DSS. They allow professional developers to use less code to deliver applications quickly. To do that, they simplify development by offering drag-and-drop reusable software components to be used in some applications. However, they do not generate UI automatically. Users without UI design knowledge will have a hard time projecting them.

Some examples are Appian[1] (Application Deployment/Integration Quickbase), Quickbase[2] - creation of workflows, and Mendix[3] - tooling for testing, building and deploying applications.

Sahay *et al.* (2020) compares 8 low-code development platforms and concludes that they are suitable for organizations with limited IT (Information Technology) resources and budget because they can deliver a full product quickly. However, the development possibilities depend on the functionalities provided by the modules available in the platform, and users might need to accommodate their initial requirements depending on the options offered by the employed platform. Also, the management and maintenance of the developed application can be hampered depending on each platform's capabilities.

The EUL framework is different from these low-code platforms as it automatically creates the DSS' UI. Users do not have to design them, like in the other platforms. The scope of functionalities is defined by the domain expert on the scripts, so they are not tied to the pre-defined functionalities of the low-code platforms, and it is also possible to manage, maintain and change the DSS at any time.

In short, the EUL framework used to generate a DSS is not a custom solution, as it can be used in many possible ways and fields.

[1] https://appian.com/
[2] https://www.quickbase.com/product/product-overview
[3] https://www.mendix.com/

3.1 Sustenagro early prototype

The work presented here has its roots in an earlier SustenAgro prototype from Suarez (2017b). It used the Decisioner framework, based on the Groovy Grails[4] framework. It helped the development of SustenAgro using ontologies to represent domain experts' knowledge and a DSL to model the DSS behavior. It automatically generates the SustenAgro interface and behavior from the ontologies and DSL script. However, Decisioner is specific for the SustenAgro DSS and uses a traditional form-based interface, not an interactive one.

The EUL framework allows a more interactive re-implementation of the SustenAgro DSS. The following chapter explains the framework.

[4] https://grails.org/

CHAPTER

4

EUL FRAMEWORK

The End-User Language (EUL) framework can automatically generate a web-based DSS from an ontology, describing the objects being analyzed and the kinds of analyses possible, and scripts in the EUL DSL, describing the DSS behavior. The data description, in the ontology, guides the creation of the DSS's web UI.

In a continuous collaboration with Embrapa researchers, the EUL was employed to develop the SustenAgro DSS in the sugarcane agricultural sustainability domain. In this context, the ontology describes the agricultural sustainability domain, while the DSL script describes the operations the DSS performs on the data to generate a sustainability report for a production unit, such as a farm or ethanol mill. To create this report, the DSS calculates the production unit sustainability and efficiency indexes from the data provided by users. Utilizing these indices, the system offers suggestions for enhancing the sustainability of the production units, thereby assisting in the decision-making process.

The structure of a EUL-based system is depicted in Figure 3. The DSL Module utilizes the DSS's Specific Ontology to compose a UI employing Widgets tailored to particular data types, such as dates, strings, and numbers. Given a particular ontology type, for example, a Farm, the DSL Module composes an interactive interface using Widgets to modify all its fields (e.g., farm name, plantation age, city, etc.). This process is repeated for all pertinent data types (Farms, Ethanol Mills, etc.). The generated UI will gather end user data, which is subsequently processed by the DSL. Upon data input completion, the DSL script calculates the indices and generates the final report using HTML widgets (Web Components). Users can interact with the report results and either print or save them.

The architecture of a DSS created using the EUL Framework is modularized into elements with defined functionalities:

The **Framework's Ontology** contains the main concepts that support the evaluation process,

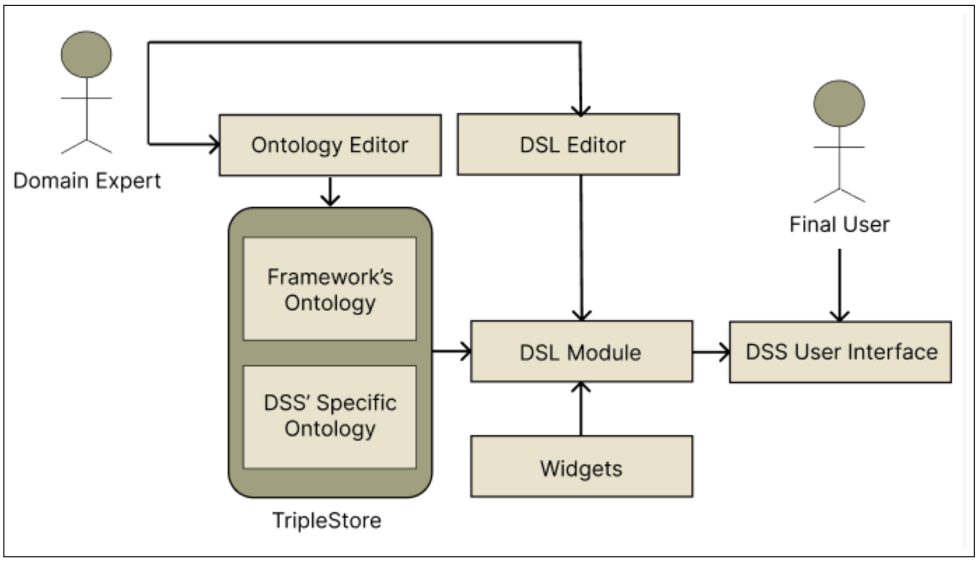


Figure 3 – EUL-based DSS architecture.

provide the mapping of data types and support the creation of graphical user interfaces.

The **DSS' Specific Ontology**: in this case, the SustenAgro ontology is developed with sustainability experts' help; it establishes the fundamental concepts used by the system, such as indicators, indicators' components, indexes, sustainability dimensions, and recommendations.

The target users for the EUL-based DSS include **domain experts**, who leverage the ontology and DSL to define the knowledge (i.e., the questions posed to the final users, the evaluation methodology, and the report format of each analysis) and the **final users**, who utilize the system to evaluate the sustainability of their production units.

Domain expert users employ the ontology and script editors to modify the DSS, as illustrated in Figure 3:

Ontology editor: a text editor for the DSS ontology. These ontologies are written using a YAML[5] based simplified format, a significant reduction from standard ontology languages like OWL. This format exploits the fact that OWL model expressivity is not required for a vast array of problems, offering a simpler language that domain experts can comprehend and utilize to describe their knowledge. An example of an ontology description is illustrated in Figure 4.

DSL editor: a text editor for the scripts written in the DSS' DSL. These scripts can apply specific transformations and analysis to users' data to assist decision-making. With them, domain experts can define the dynamic behavior of the DSS.

[5] https://yaml.org/

```
UseOfMoreResistantStrainsToTheEthanolConcentration:
  is_a: TechnologicalEfficiencyInTheIndustrial
  label:
    - Use of more resistant strains to the ethanol concentration @en
    - Utilização de cepas mais resistentes a concentração do etanol @pt
  xvalue:
    - label: [Alcohol content below 7GL @en, Grau alcoólico abaixo de 7GL @pt]
      asNumber: -1.0
    - label: [7 to 9GL @en, 7 a 9GL @pt]
      asNumber: 0.0
    - label: [Above 9 GL @en, Acima de 9 GL @pt]
      asNumber: 1.0

ReductionInWaterConsumption:
  is_a: TechnologicalEfficiencyInTheIndustrial
  label:
    - Consumo de água @pt
    - Water consumption @en
  xvalue:
    - label: [ maior do que 4 m3/ton @pt, more than 4 m3/ton @en]
      asNumber: -2.0
    - label: [2.0 até 4.0 m3/ton @pt, 2.0 to 4.0 m3/ton @en]
      asNumber: -1.0
    - label: [de 1.0 a 2.0 m3/ton @pt, from 1.0 to 2.0 m3/ton @en]
      asNumber: 0.0
    - label: [0.7 m3/ton até 1.0 m3/ton @pt, 0.7 m3/ton to 1.0 m3/ton @en]
      asNumber: 1.0
    - label: [menor do que 0.7 m3/ton @pt, less than 0.7 m3/ton @en]
      asNumber: 2.0

DevelopmentOfNewTechniquesForReduingWaterConsumption:
  is_a: TechnologicalEfficiencyInTheIndustrial
  label:
    - Desenvolvimento de novas técnicas para redução do consumo de água @pt
    - Development of new techniques for reduing water consumption @en
  xvalue:
    - label: [ não @pt, no @en]
      asNumber: -1.0
    - label: [ sim @pt, yes @en]
      asNumber: 1.0
```

Figure 4 – An example of the ontology description.

The DSS scripts are written as Groovy[6] language DSL. Groovy is a dynamic language similar to Java that supports the creation of a wide range of DSL. In the context of Groovy code, a DSL creates APIs that leverage Groovy's function closures to create an easy way to build complex functionality and data. Examples of widely used Groovy DSL languages are Gradle[7], build automation tool for Java applications, and Spock, testing, and specification framework for Java applications.

Each DSL, specific to individual DSS, is created by software developers (with assistance from domain experts). As the language is tailored to the domain of the experts, it should be more comprehensible for them. Domain experts can implement scripts using the DSS-specific DSL to define their DSS' behavior. A sample of the SustenAgro DSL is displayed in Figure 5.

Ontology and DSL editors can make real-time changes to the DSS.

A DSS, created using the EUL framework system, is divided into:

- A client part, running in a web browser as an interactive Single Page Application;

[6] https://groovy-lang.org/
[7] https://gradle.org/

```
report {

    def environment = weightedSum(sa.EnvironmentalIndicator)
    def economic = weightedSum(sa.EconomicIndicator)
    def social = weightedSum(sa.SocialIndicator)
    def sustainability = (environment + social + economic) / 3
    def costProductionEfficiency = sum(sa.ProductionEfficiency)
    def techEfficiencyInTheField = 0.8 * weightedSum(sa.TechnologicalEfficiencyInTheField)
    def techEfficiencyInTheIndustrial = 0.2 * weightedSum(sa.TechnologicalEfficiencyInTheIndustrial)
    def efficiency = abs(costProductionEfficiency) * (techEfficiencyInTheField + techEfficiencyInTheIndustrial)

    save('efficiency', efficiency)
    save('sustainability', sustainability)

    cmp 'show-report', [instance: instance()]

    cmp 'show-object', [
            class: 'show-report-tab',
            object: instance(ui.hasObject),
            description: [geo.municipality,
                          sa.partnershipsForResearchOrImprovementOfTheSystem,
                          sa.innovationDevelopmentProjects,
                          sa.financing,
                          sa.harvestYear,
                          sa.beginningOfPlantingDate,
                          sa.finishOfPlantingDate,
                          sa.beginningOfHarvestDate,
                          sa.finishOfHarvestDate]]
```

Figure 5 – A piece of the DSL.

- A server backend consisting of web services that communicate with user data, ontology entities, and DSL scripts.

The client side is developed using the JavaScript[8]-based framework VueJS[9] and incorporates the UI look-and-feel specified by Google's Material Design[10] framework.

The Google's Material Design was chosen for being an approach of design of a vast familiarity for the users, since it is present on systems like the Android, and numerous of other applications. Also, for a more agile development, to do not have the job of creating entire new web components styles from scratch.

Figure 6 exemplifies the appearance of a web component.

The server side saves user data and ontology entities in a triplestore, currently, Blazegraph[11], a graph database that stores data and reasons over semantic facts. It supports the system's storage and information recovery. The server employs the Groovy language to create and execute DSL.

A EUL-based DSS is defined by two artifacts (created by domain experts):

- An ontology describing the class (type) of each domain instance;

- Scripts, using a DSL, can apply domain-specific analyses to user data (represented as ontology instances).

[8] https://www.javascript.com/
[9] https://vuejs.org/
[10] https://material.io/
[11] https://blazegraph.com/

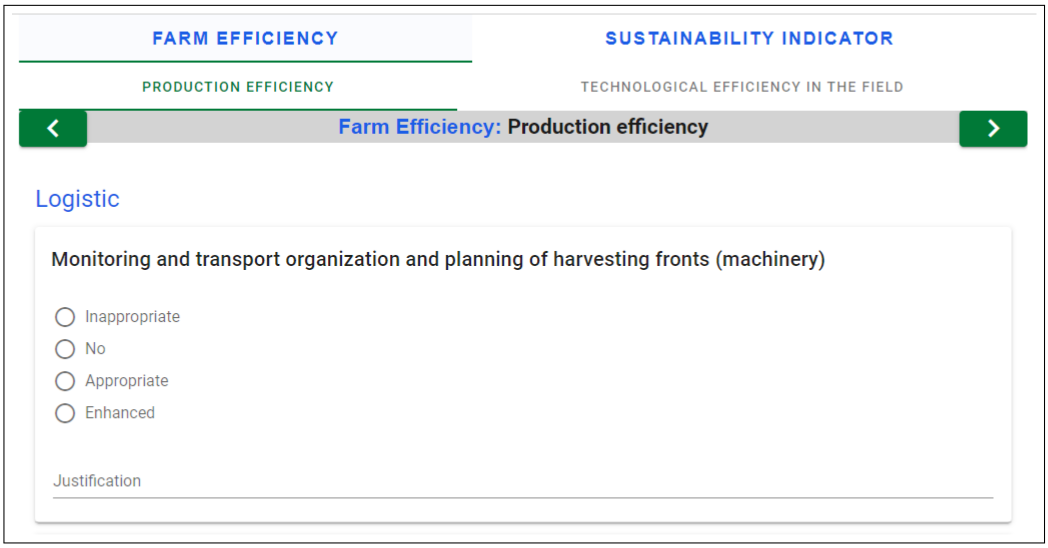


Figure 6 – Web Component appearance.

DSS created by EUL have online editors that permit domain experts to modify ontologies and DSL scripts. Following the editing process, users can regenerate the DSS, allowing them to view their changes' results immediately in the DSS UI. If users are unsatisfied with the results, they can reverse the changes. This facilitates a rapid, interactive trial-and-error development cycle. Users can experiment with various commands until the desired outcome is achieved. Error messages are displayed if something malfunctions.

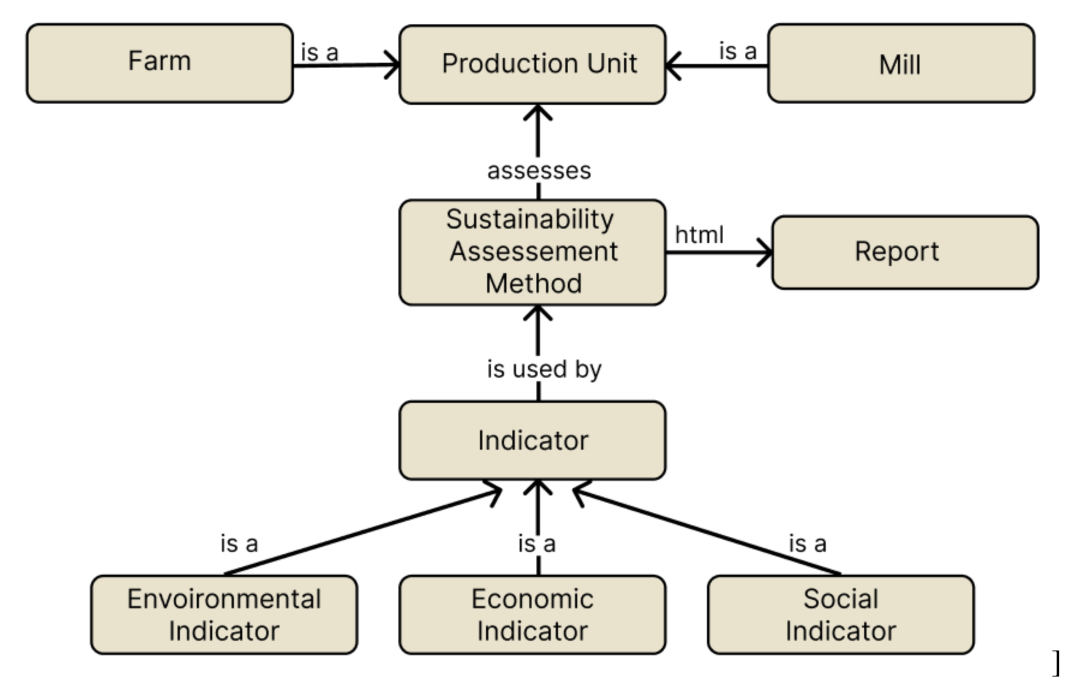
However, the current state of the online editors renders them unsuitable for domain experts to create an entirely new DSS from scratch. They function merely as text editors with buttons for saving and running the DSS when ontologies and scripts change. This online development system is only suitable for essential system maintenance rather than full-scale development.

Despite these shortcomings, a domain expert from Embrapa could make basic modifications using the system. For the first time, he was able to modify parts of a DSS himself (without the intervention of developers). In an upcoming session, we will present his evaluation of the process.

Additionally, the framework supports the creation and management of user groups. Different user groups may have varying access to different classes of instances. Users in the administrator group can use the UI to edit users, user groups, ontologies, and scripts; manage DSS data; and generate new DSS interfaces.

Each implementation of a DSS using the EUL framework can have its specific appearance. Its interface can be customized at the levels of colors and backgrounds and can use customized web components.

The following section presents the SustenAgro DSS, developed using the EUL framework.



]

Figure 7 – Conceptual map of the SustenAgro DSS.

4.1 SustenAgro DSS

The SustenAgro Decision Support System (CARDOSO, 2013) is a tool used to evaluate the sustainability of sugarcane production systems and provide recommendations on various aspects of production in farms and mills.

SustenAgro is available at http://sustenagro.icmc.usp.br/. And published by Embrapa at https://sustenagro.cnpma.embrapa.br. Its whole set of screens generated are also available at the appendix of this dissertation.

From the ontology, the framework obtains all the necessary information to assemble the web UI components that will Create, Read, Update, and Delete (CRUD) ontology instances. The SustenAgro ontology was modeled to represent the knowledge of experts, as per Allemang and Hendler (2011). The conceptual map of the SustenAgro DSS, as illustrated in Figure 7, shows the main concepts modeled and their relationships. The arrows indicate the relationship between two concepts.

The primary ontology components are the indicators of sustainability. The Indicator class represents them. Figure 8, taken from the Protegé[12] ontology editor, shows the hierarchy from the Indicator classes, which is divided into two classes: Efficiency Indicator and Sustainability Indicator. Figure 8 also shows the Adequacy of the Boilers indicator and its properties.

The SustenAgro DSS has three main instance classes (or types), namely Farm, Mill, and Analysis. Farm and Mill describe sugarcane production unities, while Analysis describes

[12] https://protege.stanford.edu/

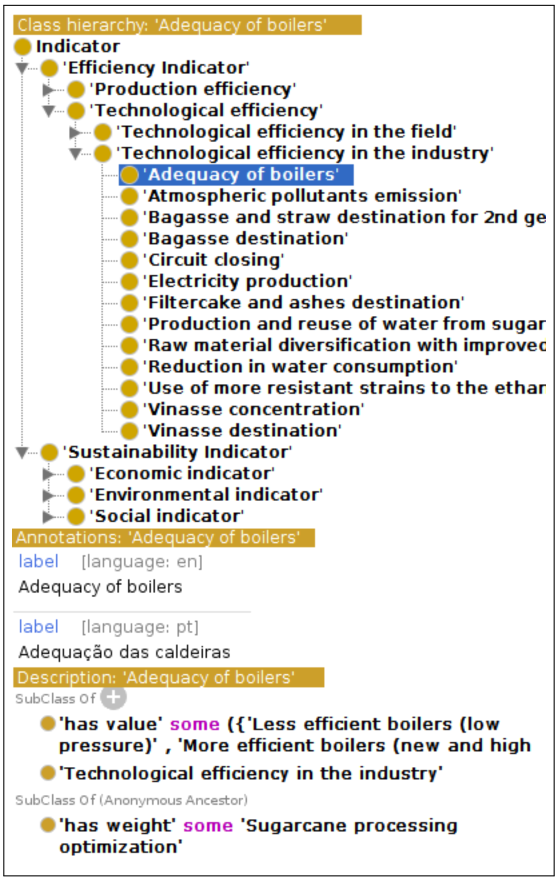


Figure 8 – Indicators' modeling.

the production features from each unity, such as whether the unit production harvest is wholly or partially mechanized. Each field (property) of these classes is defined, including the type of information each can hold, meta information such as minimum and maximum cardinalities, predefined values, etc. These types are defined in the DSS ontology created by a team of agricultural sustainability experts.

If only ontologies are defined (without scripts), the DSS can only create and edit domain instances. The scripts in the SustenAgro DSS are associated with specific class instances. When these instances are created or edited, they are called and apply domain experts' defined operations to add new information to the instances.

In the SustenAgro DSS, when Analysis instances are edited, a script is called to apply the SustenAgro method equations and add information to these instances. This added information can be in the form of new field values and HTML. SustenAgro's scripts calculate and add two values to an Analysis instance: its efficiency and sustainability indexes. It also adds web components to the instance's HTML field. When the EUL framework displays an instance with an HTML field, it only shows this field (instead of all instance fields).

SustenAgro's Analysis instances are presented as a report with text fields and graphics for the Sustainability Matrix and Semaphore. Users can use the framework edit option to modify an

Analysis instance field to generate new reports. Farm and Mill instances do not have associated scripts and are not modified.

The Sustainability Matrix, depicted in Figure 9, is a critical component of the SustenAgro DSS. It is a graphical representation of the system's evaluation of the sustainability of sugarcane production. The matrix comprises two axes corresponding to the efficiency and sustainability indexes, represented by the Y and X-axis, respectively. The indexes are divided into segments that allow the area to be partitioned into twelve quadrants of sustainability. Each evaluation generated by the SustenAgro method produces two indices that locate the sugarcane production system in one of the twelve matrix quadrants. Each quadrant is associated with a specific recommendation from the Sustenagro methodology.

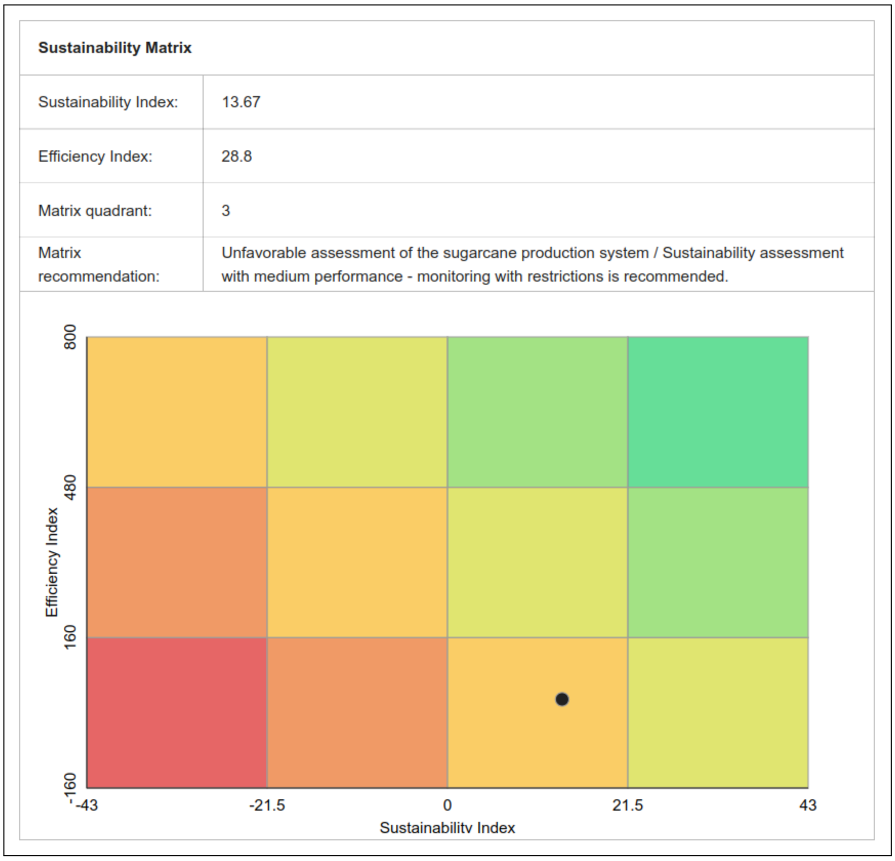


Figure 9 – Sustainability Matrix.

The Semaphore, depicted in Figure 10, is another graphical representation of sugarcane production sustainability generated by the SustenAgro DSS. The Semaphore has an axis that quantifies the value of sustainability normalized from -100 to +100, divided into five segments corresponding to sustainability categories.

The HTML fields in the SustenAgro DSS employ web components to display results. Although a predefined set of static web components is available, each DSS can add its own components to display domain-specific graphics. For example, the SustenAgro DSS adds the Sustainability Matrix web component to exhibit the efficiency and sustainability indexes in a 3 x

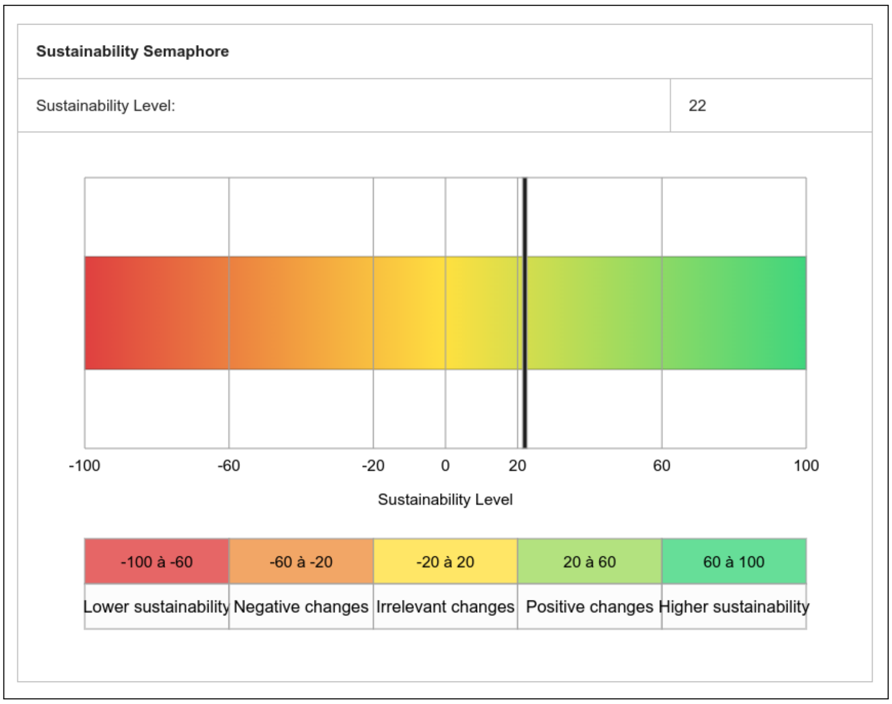


Figure 10 – Sustainability Semaphore.

4 grid. These web components are added as JavaScript files in a folder specific to each DSS.

The EUL Framework is used to build the SustenAgro DSS. The system is generated by converging all the knowledge described in the specific ontology created by the Embrapa domain experts. This ontology provides the necessary information, combined with the Frameworks' ontology, to create widgets that the users interact with to provide inputs to the DSS. These inputs are sent to the DSL module, which is also developed by domain experts and contains all metrics and indexes to calculate the results. These DSL scripts process the input given and generate the web components that display the results in a report, which includes information about the Farm/Mill, and the Sustainability Matrix and Semaphore.

Embrapa's domain experts use the results generated by the SustenAgro DSS to aid in their analysis and decision-making when assessing the sustainability of sugarcane production in Brazil's center-south region. The domain experts themselves can modify the Sustenagro DSS. This aspect and the system's usage and behavior underscore the need to evaluate this DSS generated with the EUL Framework. Such an evaluation is essential to determine whether the framework is adequate and the system is genuinely suitable for its target audience. The next chapter is dedicated to the usability evaluations of the Sustenagro DSS.

CHAPTER

5

USABILITY EVALUATION

One of the core goals of this work is to evaluate the usability of the DSS generated by the EUL framework. Specifically, we evaluated the SustenAgro DSS usability for its final users, i.e. agricultural technicians who use the DSS to evaluate the agricultural sustainability of sugarcane production processes in farms and mills. These users possess technical backgrounds in agricultural techniques used in the sugarcane growing and processing industries, such as agronomists or environmental technicians. To achieve this goal, we conducted a Usability Test (NIELSEN, 1994) with four Embrapa employees having this profile. Since Embrapa as our partners in this project did not want this work to be public they only made available this amount of users to test the tool.

Additionally, we performed a qualitative usability test of EUL's tools for ontology and DSL scripts creation and edition, using SustenAgro's second user group - administrators, who are domain experts in agricultural sugarcane sustainability. However, evaluating these tools is more challenging as EUL's ontology and script edition tools are still primitive, since it does not have much more functionalities than a basic text editor. Nevertheless, users can perform simple, useful tasks using these tools. For this test, we engaged an agricultural sustainability expert from the São Paulo Agency of Agribusiness Technology (APTA) who worked on the SustenAgro implementation. For the same reason as before, we did not have access to more domain experts that could add more feedback on this test.

Furthermore, we conducted a heuristic evaluation (NIELSEN; MOLICH, 1990) with three experts on User Interfaces and Experience, from Intermidia Lab, affiliated to the Institute of Mathematical Sciences and Computation (ICMC) , on USP, São Carlos, to focus on more traditional usability issues.

The evaluation experiments were approved by the Brazilian Ethics Committee under CAAE n^{o} 60435022.3.0000.5504.

All supplementary material from these evaluations is available on the following Google

Docs Repository [13].

It is important to note that we can only evaluate the DSS's usability as a whole, which includes issues related to the framework itself, such as the UI elements it generates, and issues related to the modeling and methods of the DSS being created, such as how well-defined its concepts are. However, these two kinds of issues were separated in the discussions.

5.1 Usability Tests with End Users

The steps applied to test the DSS usability for end users, was the use of Think Aloud, then, the application of the SUS form to measure the satisfaction of the use.

Think aloud is great to extract great information from people, and we can extract the most of feedback with few users (NIELSEN, 1994), since we had available 4 users for this step of testing.

The Think Aloud data was collected with the users sharing their screen and saying comments while they were performing the tasks, this way we could record the sessions and extract all data we needed.

In addition to Think Aloud, the test utilized the SUS form, in which the participants rate their experience and the satisfaction with the system on a 5-point scale, ranging from "Strongly Disagree" to "Strongly Agree" (BROOKE, 1996).

SUS scores have demonstrated a high correlation with more comprehensive usability measures, and the scale is often employed in conjunction with other methods, such as heuristic evaluations (BROOKE, 1996). By incorporating both the Think Aloud method and SUS form, this study aimed to provide a robust and comprehensive assessment of the DSS usability for end users.

5.1.1 Participants

The evaluation of this specialized system necessitates participants with prior knowledge of specific terms and processes. Consequently, identifying a sufficient number of domain experts with the required background proved challenging due to lack of availability of this type of users that the partnership provided us.

For this evaluation, four participants with the following profile were chosen as final users: they were, on average, 40 years old, everyone had at least a graduate degree (Master's or Ph.D.) in agriculture, and worked as researchers in the agricultural domain.

[13] https://drive.google.com/drive/folders/1cGXOfvk8O6qEGhuV$_x j49Scdxu0U4oHc?usp = sharing$

5.1.2 Methods

A videoconference, utilizing the Google Meet platform, was employed to conduct the usability test. At the beginning of the conference, the participants were briefed on the test's purpose, their tasks, and the expectations for their performance during the session. Subsequently, the participants were asked for their consent to record the session and to perform the test.

The participants were instructed to execute the following tasks using the system:

- Task 1 - Register as a user, log in to the system, and navigate to the introduction page;
- Task 2 - Create a farm record (instance);
- Task 3 - Conduct a sustainability evaluation for a farm and visualize the results;
- Task 4 - Edit the sustainability evaluation for the farm and visualize the results;
- Task 5 - Create a mill record and edit it;
- Task 6 - Conduct a sustainability evaluation for a mill and visualize the results;
- Task 7 - Navigate to the results page;
- Task 8 - Remove a farm or a mill record;
- Task 9 - Edit user's data;
- Task 10 - Utilize one or more system functions deemed most important by the user.

Participants were instructed to verbalize their thoughts as they completed each task. Researchers periodically reminded them to express their thoughts and feelings throughout the process. Upon completion of the tasks, the SUS was administered to assess participants' satisfaction with the system. A debriefing session was conducted afterward, allowing users to provide additional comments.

In addition to the qualitative data collected, the following metrics were employed to evaluate the system: (1) time taken to complete tasks; (2) positive and negative comments extracted through speech analysis; and (3) whether or not the user was able to accomplish the tasks.

The best time to complete each task was estimated during a pilot test, wherein a trained user with prior knowledge of the system and method executed the tasks. This user was a partner researcher with previous experience working with the system and method. This way it is possible to compare and estimate how these users can complete the tasks again when they have familiarity with the system.

Tasks	**Best time (m:s)**	**Users average (m:s)**
Task 1	1:30s	2:47s
Task 2	2:25s	4:32s
Task 3	12:28s	22:10s
Task 4	1:45s	2:12s
Task 5	2:37s	2:56s
Task 6	12:48s	16:44s
Task 7	32s	1:20s
Task 8	11s	19s
Task 9	43s	1:52s
Task 10	–	–

Table 1 – Comparison between best and average time taken by users to complete tasks.

The time taken by users to complete all tasks, along with the best time for each task, is presented in Table 1.

When analyzing the time results, it is important to consider that this was the first time the participants interacted with the system, and the Think Aloud method can influence the speed at which users complete tasks.

As such, the best time should be viewed more as a lower limit rather than a goal for task execution times. More crucially, all users were able to successfully complete every task.

Time was not measured for Task 10, as the task's objective allowed each user to select a different task to perform in order to debug the system.

After completing the tasks, participants were asked to provide feedback on their experience using the system. This enabled the extraction of positive and negative comments from their experiences.

Some positive comments extracted from users, using speech analysis during the Think Aloud application, included:

- The system is self-explained.
- The system is well-structured.
- The graphics are impressive.
- The user interface is smooth.
- The system is easy-to-use.

Regarding negative comments, it is noteworthy that most participants referred to issues with the SustenAgro method itself, rather than the interface. For example:

- It took too long to finish the tasks related to the analysis of Farms and Mills because of the number of questions;
- They did not understand some explanations for terminologies used on the questions.

Domain experts from Embrapa can address these negative aspects by improving the method and making the decision-making data collection process more direct or concise. The results were shared with Embrapa research partners for consideration in future versions of the system.

The only negative comments related to the framework were:

- Users found the icons for saving and submitting instances confusing.
- The controls for changing pages were unclear in one of the interfaces, and its internal scrolling was mixed up with the browser page's scrolling.

Modifications were made to the generated UI to address these negative comments. These changes will be revisited in the discussion of the heuristic analysis.

5.1.2.1 System Usability Scale

To gather feedback on user satisfaction with their interaction with the system, the SUS was administered. All data collected to calculate the SUS score is displayed in Table 2. The Answers column indicates the number of users who responded with each value. For instance, for the first question, no participant answered 1, one answered 2, one answered 3, and so on.

System Usability Scale metrics	Answers				
Scale from 1 to 5	**1**	**2**	**3**	**4**	**5**
I think I'd like to use this system regularly	-	1	1	1	1
I think this system unnecessary complex	1	2	-	1	-
I think this system is easy to use	-	2	-	2	-
I think I'd need help from someone with technical knowledge to use this system	-	2	-	2	-
I think that the various functions of the system are well integrated	-	-	1	3	-
I think the system has inconsistency	-	1	3	-	-
I think people will learn how to use this system	-	-	1	2	1
I found the system goofy to use	-	2	1	1	-
I felt confident while using the system	-	-	1	2	1
I needed to learn lots of things while using the system	1	2	-	1	-

Table 2 – SUS results from end users evaluation.

During the Think Aloud process, users completed tasks without significant errors or intervention from the interviewers, we can see this in the SUS results (see Figures 11 and 12). In

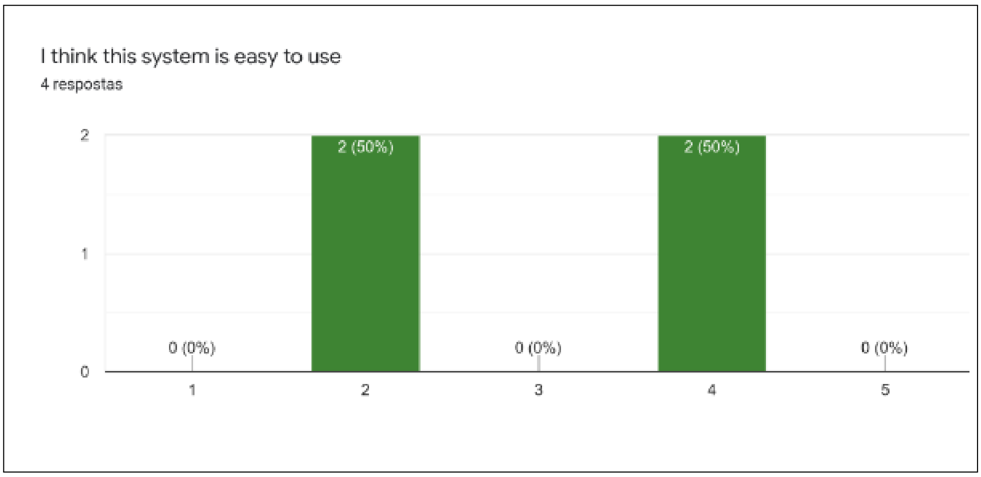


Figure 11 – Answer from System Usability Scale 1.

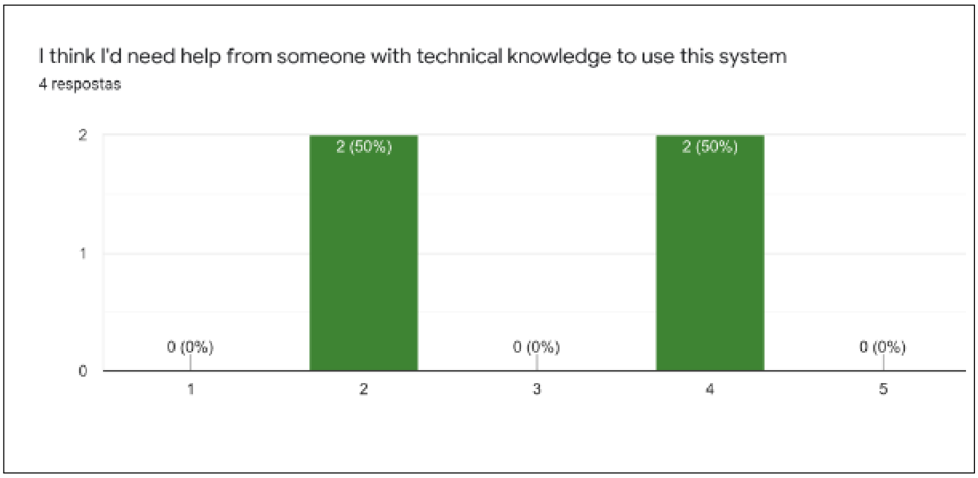


Figure 12 – Answer from System Usability Scale 2.

which users judge that the system is easy to use and if they will need techinical knowledge to use the system. The results are displayed on a scale from 1 to 5, 1 means strongly disagree, and 5 means strongly agree.

Two out of four participants agree that the system is easy-to-use (4); while the other two disagree (2). The same result emerged when asked if users needed help from someone with technical knowledge to use the system. That is primarily connected to the issues identified regarding the SustenAgro method and its terminology used in the questions.

During task execution, users had a fluid interaction with the system. However, when applying the method and engaging in activities that involved sustainability evaluations, users took more time to complete them. This, along with unclear terminology in the questions, were aspects users complained about during the evaluation sessions. This may explain why 50 percent of users did not find the system easy to use, even after having a good interaction with it.

Three out of four users agreed that the system was well integrated, and one remained neutral regarding the system structure and functions. Conversely, three out of four were neutral

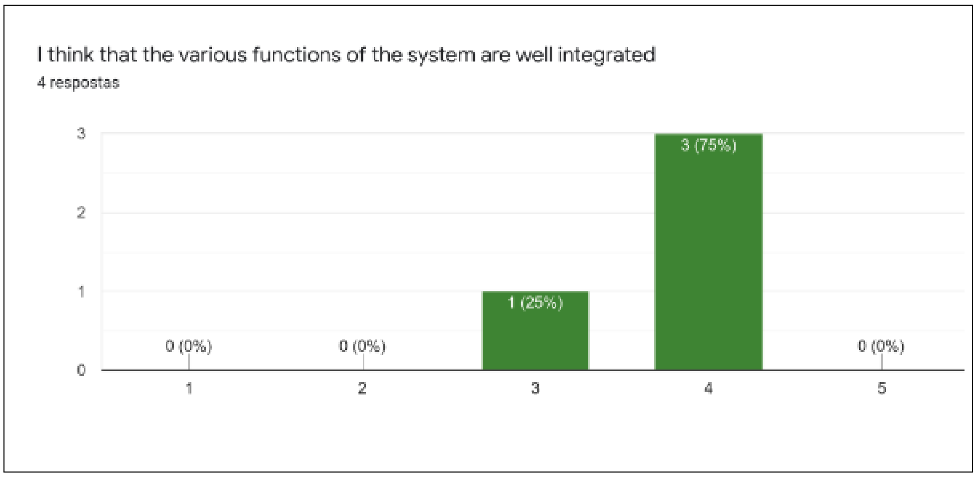


Figure 13 – Answer from System Usability Scale 3

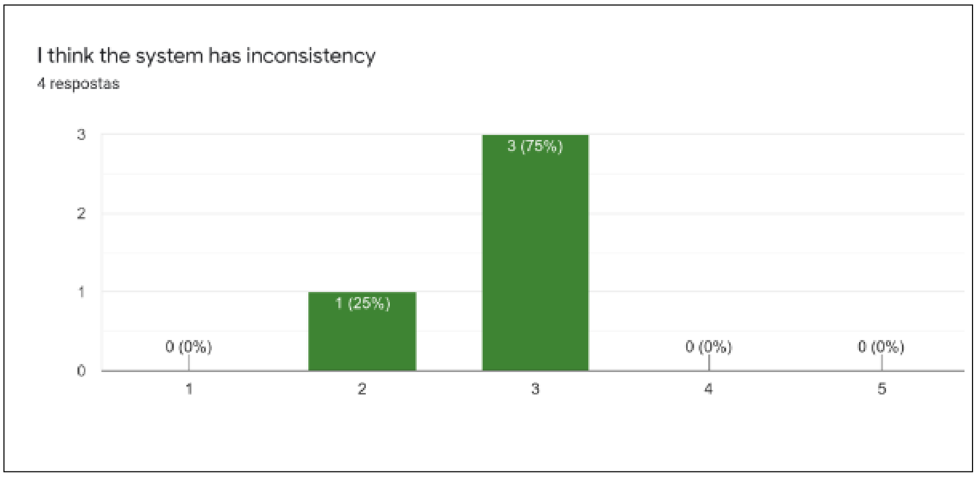


Figure 14 – Answer from System Usability Scale 4

when asked if the system presented inconsistency, and one disagreed, because found the system consistent enough. Regarding confidence, while using the system, one of four was neutral, two of four agreed that they felt confident, and the last of the four strongly agreed. These results are illustrated in Figure 13 and Figure 14.

The UI-related issue participants most frequently complained about pertained to the set of pages containing questions related to the production system features. Users found the pagination and scrollbars on these pages confusing while answering questions. When users reached the last page of questions, the system automatically returned to the first one, leaving users unsure when the questionnaire concluded. Additionally, the form scrollbar and browser scrollbar became intertwined, occasionally requiring users to use both scrollbars to access the bottom of the questions. These problems are depicted in Figure 15, and we revisit these issues in the Heuristic Evaluation section (5.3).

Other problems related to the dependency of the method itself, like time-consuming to finish some tasks in the system because of the amount of questions, were also addressed to the

Embrapa team, as previously mentioned.

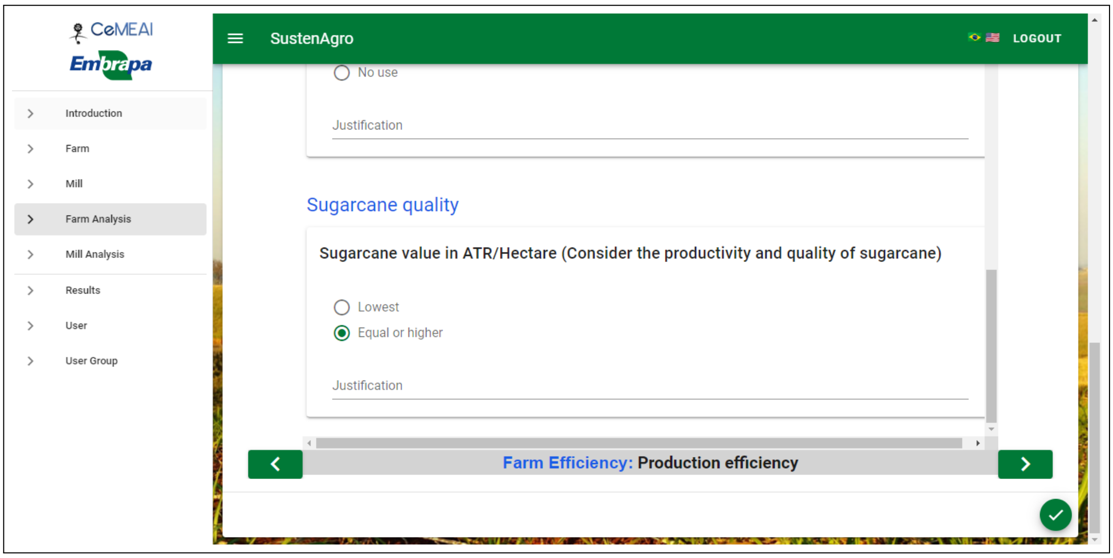


Figure 15 – Bottom of the Analysis Page.

Lastly, the SUS score attained an overall average of 63.1 points. In practice, the ideal average for SUS score ranges from 68-70, which renders the 63.1 value, showing that improvements are needed to reach the ideal range of user satisfaction. Combining both efforts from domain experts to solve the time-consuming part of the tasks with a better UI generation.

5.2 Domain Expert Usability Test

Due to the limited availability of domain experts and constraints in EUL's tools for ontology and DSL script creation/edition, we opted for a qualitative usability test. The aim was to gain insight into how a domain expert would use the system.

The test assessed the quality of domain experts' interactions with the system. We employed the same steps for the final users, using the Think Aloud, and the application of the SUS, and an interview to obtain at least one domain expert's perspective. Unlike the end-user interface, the domain expert interface necessitated basic training. The selected participant was a domain expert involved in the SustenAgro DSS implementation and had prior interaction with the system. We wanted to understand if the system met their needs.

5.2.1 Participant

The participant is a domain expert in the agricultural sustainability domain. He is over 50 years old and has a master's degree.

5.2.2 Methods

We followed the same procedure used for the end-user think-aloud tests, utilizing the Google Meeting tool. The participant was asked for consent for the recording of the session and to execute the following tasks using the system:

- Task 1 - Log in to the system as an administrator.
- Task 2 - Edit questions: change the text from questions and create labels for another language.
- Task 3 - Create a new question.
- Task 4 - Create a new question category at the lowest level (based on SustenAgro method hierarchy).
- Task 5 - Create a new question category at the highest level (based on SustenAgro method hierarchy).
- Task 6 - Create a new field on the farm.
- Task 7 - Edit one of the fields from the sustainability matrix widget.
- Task 8 - Change a formula on the SustenAgro script.

We asked the participant to think out loud as he completed the tasks. Afterward, we applied the SUS to assess his satisfaction with the system. Finally, we conducted a debriefing, including an interview to gather his feedback.

In this test, we analyzed whether the user could accomplish the tasks and noted his positive and negative feedback, extracted from the interview.

The domain expert was able to complete all tasks, although he required assistance from the interviewers for some of the more complex activities (3-6). During the tasks, the user reported no issues related to the generated UI. He was able to use the editors and tools to modify the ontology and DSL script, and subsequently, he was able to initialize the DSS with the new values.

This expert had used the system before and performed tasks 1 to 3, but that happened months ago, and he complained about not remembering some commands. During the test, he was able to complete tasks 1, 2, 7, and 8 with minimum or no help.

After completing all the tasks, the participant answered the SUS. The score reached the general mean of 52.5 points, indicating that the system needs improvements from the domain expert usability perspective. The user's answers can be seen in Table 3.

In the end, we interviewed the user. Among other things, we asked which tasks he felt he could do independently if given more time. He responded that he could accomplish all tasks if

System Usability Scale metrics	**Answers**				
Scale from 1 to 5	**1**	**2**	**3**	**4**	**5**
I think I'd like to use this system regularly	-	-	-	1	-
I think this system unnecessary complex	-	-	-	1	-
I think this system is easy to use	-	-	-	-	1
I think I'd need help from someone with technical knowledge to use this system	-	-	-	1	-
I think that the various functions of the system are well integrated	-	-	1	-	-
I think the system has inconsistency	-	-	-	1	-
I think people will learn how to use this system	-	-	-	-	1
I found the system goofy to use	-	-	-	-	1
I felt confident while using the system	-	-	-	-	1
I needed to learn lots of things while using the system	-	-	-	1	-

Table 3 – SUS results from domain expert evaluation.

he had some training. We also asked if he felt he could apply this same methodology to a new DSS using the same SustenAgro structure, and he answered positively.

The main issue is that the ontology and script editors are very basic. They provide error messages but do not treat the code at a high level as more professional editors do. Additionally, the ontology editor is just a text editor with no graphic interface to edit the ontology components. A graphic ontology editor would greatly simplify the process.

From a qualitative standpoint, this user was satisfied with the SustenAgro development experience, even with its shortcomings, compared to his past experiences with custom programs created by contractors.

5.3 Heuristic Evaluation

For the final system test, we applied Nielsen's heuristics evaluation following its methodology (NIELSEN; MOLICH, 1990). We are using three UI/UX specialists, to analyze the system's user interface generated for the DSS using the 10 Nielsen Heuristics:

- Visibility of system status.
- Match between the system and the real world.
- User control and freedom.
- Consistency and standards.
- Error prevention.
- Recognition rather than recall.

- Flexibility and efficiency of use.
- Aesthetic and minimalist design.
- Help users recognize, diagnose, and recover from errors.
- Help and documentation.

When a specialist judges that the UI does not follow some of these guidelines, he determined a severity rating, on a scale from 0 to 4, for each guideline with problems. There is also a suggestion for a solution in each problem found. Then, a report was created as a consolidation file with all problem evaluations, containing the problem's location, its description, its severity rating, and suggestions for improvements from the specialists.

In this case, we used three UI specialists from Intermidia Labs, affiliated to ICMC-USP, to analyze the generated interface of the Sustenagro DSS. At the appendix, the Table 4, Table 5, Table 6, and Table 7, contain the issues found by the specialists.

In total, 57 issues were identified. Nine problems were found on all pages, five in the sidebar, one in the top bar, two on the introduction page, one on the user page, 14 issues on the add farm and add mill pages, two issues spanning the add farm, mill and also add analysis pages, eight problems on the add analysis pages, three issues encountered while submitting the forms for the analysis, two issues on the report page, one spanning the report and the results page, three exclusively for the results pages, and four on the farm, mill, and analysis listing pages.

There are 10 issues with Severity 1, 23 with Severity 2, 11 with Severity 3, and 12 with Severity 4.

Regarding the frequency of each guideline violation:

- *Visibility of system status*- 6.
- *Match between the system and the real world* - 4.
- *User Control and freedom* - 1.
- *Consistency and standards* - 11 times.
- *Error prevention* - 19 times.
- *Recognition rather than recall* - 3.
- *Aesthetic and minimalist design* - 10.
- *Help users recognize, diagnose, and recover from errors* - 4.
- *Help and documentation* - 2.

The most problematic pages violated the following guidelines:

- Visibility of system status: the user does not know when the pagination ends, and they can finish filling out the form;
- Flexibility and efficiency of use: the user cannot scroll the page one single time to keep up with the activity of filling out the form; there is an internal scrollbar that is separate from the browser scrollbar, which annoyed the users during use.

These problems, which were also reported during the usability tests, have now been fixed, and the UI no longer exhibits these behaviors.

We can conclude from the evaluation that, as a prototype, the DSS Sustenagro generated by the EUL Framework satisfies its purposes of validation. But the evaluation shows that further improvements can be made before it can be delivered as a final product.

Some of the more critical problems found in both the usability tests and the Heuristic Evaluation have already been addressed, such as for instance, the scrollbars and pagination issues.

CHAPTER

6

CONCLUSION

The development and usability tests of the SustenAgro Decision Support System have demonstrated its efficacy, confirming that the EUL framework can successfully generate a DSS from its ontology and scripts, including the automatic generation of the interface.

The DSS, which was generated based on the EUL Framework, exhibited good usability. Users encountered no major difficulties in navigating the DSS and completing the proposed tasks. All necessary information was provided, and the system enabled the generation of final reports for decision-making processes.

User interviews provided valuable insights into potential system improvements, primarily related to the user interface. Despite users suggesting areas for enhancement, the overall structure of the system was deemed to be well-organized.

Domain experts were able to complete the proposed tasks, particularly those involving system modifications using ontologies and the Domain-Specific Language. Despite some limitations within the EUL framework, the domain expert interviewed expressed satisfaction with the framework's maintainability compared to software contractors. The Embrapa domain experts, who had not previously utilized an ontology to formalize their work, were pleased with the experience. The ontology has since been employed beyond the DSS as a tool for more precise modeling of the SustenAgro method.

Although evaluation tests of the EUL framework did not achieve the ideal System Usability Scale metrics for end-users, the results informed further research efforts. It is believed that refining the user interface generation will satisfy all usability heuristics that needs to be improved as it is reported on the heuristics evaluation.

In summary, the hypothesis that the EUL Framework can use data formats described by ontologies to create a user interface with good usability for a DSS is validated by the Sustenagro DSS prototype generated. The combination of ontologies and a DSL enables system generation, and they can be edited online by domain experts within the system. This simplifies

DSS maintainability and allows for the editing of basic features.

The goals outlined in this study were achieved: the Sustenagro DSS is a usable, automatically generated system based on the EUL Framework, and it has been tested and evaluated. Furthermore, it has been proven that domain experts can indeed modify the DSS themselves.

The EUL framework and the development of EUL-based DSS still require improvements and validation in other domains. One notable area for enhancement is the development of graphical tools that domain experts can use to define domain ontologies and create new versions of SustenAgro for other agricultural domains, such as the soy domain.

6.1 Future Work

Future work will involve the refinement of the UI generation, enhancing the online ontology and script editors to include a graphical ontology editor and an online Integrated Development Environment (IDE), providing domain experts with more comprehensive development support. Additionally, the creation of EUL-based DSS in other areas will demonstrate the tool's versatility and applicability across different domains.

BIBLIOGRAPHY

ALLEMANG, D.; HENDLER, J. **Semantic web for the working ontologist: effective modeling in RDFS and OWL**. [S.l.]: Elsevier, 2011. Citation on page 44.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. *et al.* The semantic web. **Scientific american**, New York, NY, USA:, v. 284, n. 5, p. 28–37, 2001. Citation on page 32.

BROOKE, J. Sus: a "quick and dirty'usability. **Usability evaluation in industry**, v. 189, n. 3, 1996. Citations on pages 24, 31, and 50.

CARDOSO, B. O. Avaliação da sustentabilidade de sistemas de produção da cana-de-açúcar no estado de são paulo: uma proposta metodológica e de modelo conceitual. Universidade Federal de São Carlos, 2013. Citation on page 44.

DEURSEN, A. V.; KLINT, P.; VISSER, J. Domain-specific languages: An annotated bibliography. **ACM Sigplan Notices**, ACM, v. 35, n. 6, p. 26–36, 2000. Available: <https://john.cs.olemiss.edu/~hcc/csci555/notes/localcopy/DSLAnnotatedBib.pdf>. Citation on page 33.

EVANS, E. **Domain-driven design: tackling complexity in the heart of software**. Addison-Wesley Professional, 2004. Available: <https://books.google.com.br/books?hl=en&lr=&id=xColAAPGubgC&oi=fnd&pg=PR9&ots=qbYDgiUM6p&sig=hrqyW6aokeekbsUJ8iYbw12m9bE&redir_esc=y#v=onepage&q&f=false>. Citation on page 28.

FENU, G.; MALLOCI, F. M. Dss lands: A decision support system for agriculture in sardinia. **HighTech and Innovation Journal**, v. 1, n. 3, p. 129–135, 2020. Citation on page 27.

FOWLER, M. Language workbenches: The killer-app for domain specific languages. 2005. Available: <http://www.cime.cl/archivos/ILI253/8870_cl2-MartinFowler-Language-Workbench-DSL.pdf>. Citation on page 33.

GHOSH, D. Dsl for the uninitiated. **Communications of the ACM**, ACM, v. 54, n. 7, p. 44–50, 2011. Available: <https://homepages.dcc.ufmg.br/~mtov/arqsw/p44-ghosh.pdf>. Citation on page 33.

HAILPERN, B.; TARR, P. Model-driven development: The good, the bad, and the ugly. **IBM systems journal**, IBM, v. 45, n. 3, p. 451–461, 2006. Citation on page 30.

HORROCKS, I. Ontologies and the semantic web. **Communications of the ACM**, ACM New York, NY, USA, v. 51, n. 12, p. 58–67, 2008. Citation on page 28.

JESUS, K. R. E. de; TORQUATO, S. A.; MACHADO, P. G.; Brumatti Zorzo, C. R.; CARDOSO, B. O.; LEAL, M. R. L. V.; PICOLI, M. C. A.; RAMOS, R. C.; DALMAGO, G. A.; CAPITANI, D. H. D.; DUFT, D. G.; SUáREZ, J. G.; Pierozzi Junior, I.; TREVELIN, L. C.; MOREIRA, D. A. Sustainability assessment of sugarcane production systems: Sustenagro decision support system. **Environmental Development**, v. 32, p. 100444, 2019. ISSN 2211-4645. Available: <https://www.sciencedirect.com/science/article/pii/S2211464517303408>. Citation on page 21.

KLEPPE, A. G.; WARMER, J.; WARMER, J. B.; BAST, W. **MDA explained: the model driven architecture: practice and promise**. Addison-Wesley Professional, 2003. Available: <https://lsr.omg.org/mda/mda_files/KleppeMDAFlyerMaster.pdf>. Citation on page 30.

KOLTHOFF, K. Automatic generation of graphical user interface prototypes from unrestricted natural language requirements. In: IEEE. **2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [S.l.], 2019. p. 1234–1237. Citation on page 36.

LAAZ, N.; MBARKI, S. Ontoifml: Automatic generation of annotated web pages from ifml and ontologies using the mda approach: A case study of an emr management application. In: **MODELSWARD**. [S.l.: s.n.], 2019. p. 353–361. Citation on page 36.

LUCRÉDIO, D. Uma abordagem orientada a modelos para reutilização de software. **INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO UNIVERSIDADE DE SÃO PAULO**, p. 37, 2009. Available: <http://www2.dc.ufscar.br/~daniel/files/teseDoutoradoDanielLucredio.pdf>. Citation on page 30.

MCGUINNESS, D. L. Owl web ontology language overview w3c recommendation 10 february 2004. **http://www. w3. org/TR/2004/REC-owl-features-20040210/**, 2007. Available: <https://www.w3.org/TR/2004/REC-owl-features-20040210/>. Citation on page 33.

MOLINA, P. J. Quid: A web-based dsl for defining user interfaces applied to web components. **Actas delas XXIII Jornadas de Ingeniería del Software y Bases de Datos**, 2018. Citation on page 35.

MOREIRA, A.; ALVARENGA, L.; OLIVEIRA, A. d. P. O nível do conhecimento e os instrumentos de representação: tesauros e ontologias. **DataGramaZero-Revista de Ciência da Informação**, v. 5, n. 6, p. 1–25, 2004. Available: <http://www.brapci.inf.br/_repositorio/2010/01/pdf_25cb9ed242_0007645.pdf>. Citation on page 32.

NICHOLS, J.; FAULRING, A. Automatic interface generation and future user interface tools. In: **ACM CHI 2005 Workshop on the future of user interface design tools**. [S.l.: s.n.], 2005. Citation on page 30.

NIELSEN, J. **Usability engineering**. [S.l.]: Morgan Kaufmann, 1994. Citations on pages 24, 31, 49, and 50.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: **Proceedings of the SIGCHI conference on Human factors in computing systems**. [S.l.: s.n.], 1990. p. 249–256. Citations on pages 31, 49, and 58.

PATEL-SCHNEIDER, P. F. Building the semantic web tower from rdf straw. In: **IJCAI**. [s.n.], 2005. p. 546–551. Available: <http://www.bell-labs.co/who/pfps/publications/fol.pdf>. Citation on page 32.

PIERONI, R. *et al.* Desenvolvimento de uma dsl para a gerência de configuração de um sistema de gerenciamento integrado de redes. Universidade Federal de São Carlos, 2014. Available: <https://repositorio.ufscar.br/bitstream/handle/ufscar/590/6469.pdf?sequence=1>. Citation on page 33.

POWER, D. J. **Decision support systems: concepts and resources for managers**. Greenwood Publishing Group, 2002. Available: <https://scholarworks.uni.edu/cgi/viewcontent.cgi?article=1066&context=facbook>. Citation on page 29.

ROUSSEY, C.; SOULIGNAC, V.; CHAMPOMIER, J.-C.; ABT, V.; CHANET, J.-P. Ontologies in agriculture. In: CEMAGREF. **AgEng 2010, International Conference on Agricultural Engineering**. [S.l.], 2010. p. p–p. Citation on page 22.

RUIZ, J.; SERRAL, E.; SNOECK, M. Evaluating user interface generation approaches: model-based versus model-driven development. **Software & Systems Modeling**, Springer, v. 18, n. 4, p. 2753–2776, 2019. Citation on page 28.

SAHAY, A.; INDAMUTSA, A.; RUSCIO, D. D.; PIERANTONIO, A. Supporting the understanding and comparison of low-code development platforms. In: **2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.: s.n.], 2020. p. 171–178. Citation on page 37.

SCHIUMA, G.; GAVRILOVA, T.; ANDREEVA, T. Knowledge elicitation techniques in a knowledge management context. **Journal of Knowledge Management**, Emerald Group Publishing Limited, 2012. Available: <https://www.emerald.com/insight/content/doi/10.1108/13673271211246112/full/html>. Citation on page 21.

SILVA, C. R. Q. Critérios para priorização de estudos primários identificados por snowballing com conjunto inicial gerado por string de busca. Universidade Federal de São Carlos, 2017. Citation on page 35.

SOTO, A.; MORA, H.; RIASCOS, J. A. Web generator: An open-source software for synthetic web-based user interface dataset generation. **SoftwareX**, Elsevier, v. 17, p. 100985, 2022. Citation on page 36.

SUAREZ, J. F. G. **Ontologias e DSLs na geração de sistemas de apoio à decisão, caso de estudo SustenAgro**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2017. Available: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-26072017-113829/en.php>. Citation on page 28.

SUAREZ, J. F. G. **Ontologias e DSLs na geração de sistemas de apoio à decisão, caso de estudo SustenAgro**. Phd Thesis (PhD Thesis) — Universidade de São Paulo, 2017. Citation on page 38.

TWEEDALE, J. W.; PHILLIPS-WREN, G.; JAIN, L. C. Advances in intelligent decision-making technology support. In: **Intelligent Decision Technology Support in Practice**. Springer, 2016. p. 1–15. Available: <https://link.springer.com/chapter/10.1007%2F978-3-319-21209-8_1>. Citations on pages 27 and 28.

W3C. **Web Ontology Language**. 2012. Available: <https://www.w3.org/OWL/>. Accessed: 27 jul. 2019. Citation on page 33.

APPENDIX

A

DSS SUSTENAGRO SCREENS

As mentioned in Chapter 4 this appendix is attached the screens automatically generated by the EUL Framework, the SustenAgro DSS.

Figure 16 – Introduction page for the SustenAgro DSS for End-Users.

Figure 17 – Introduction page for the SustenAgro DSS for Domain Experts.

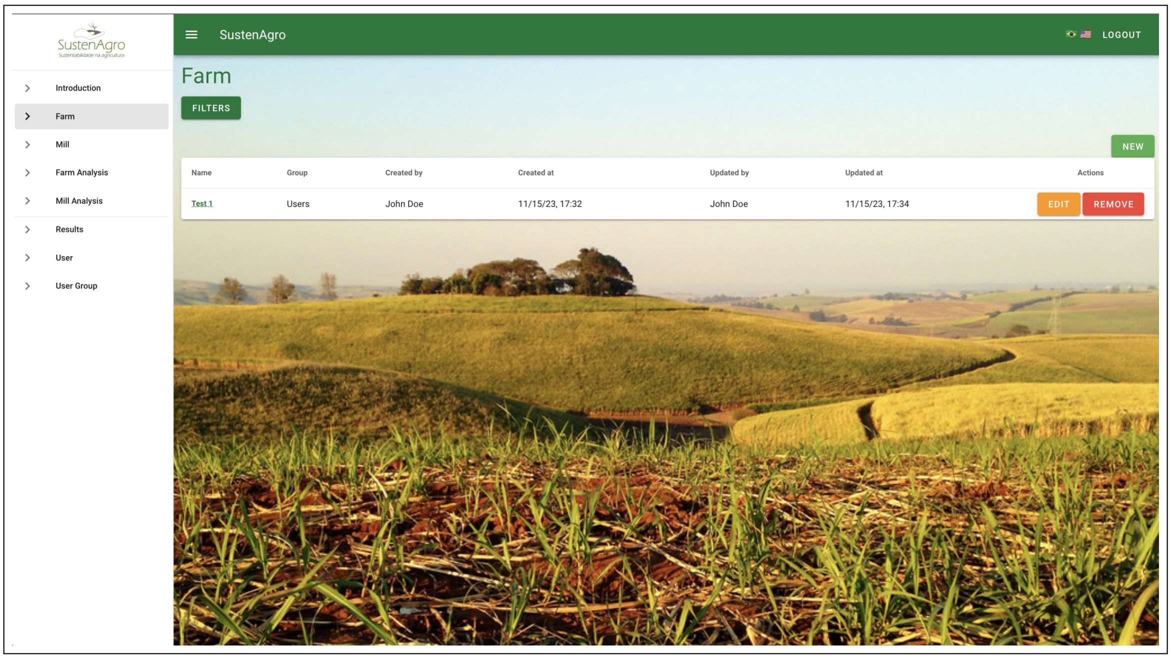


Figure 18 – Farm Page (to list and create a new one).

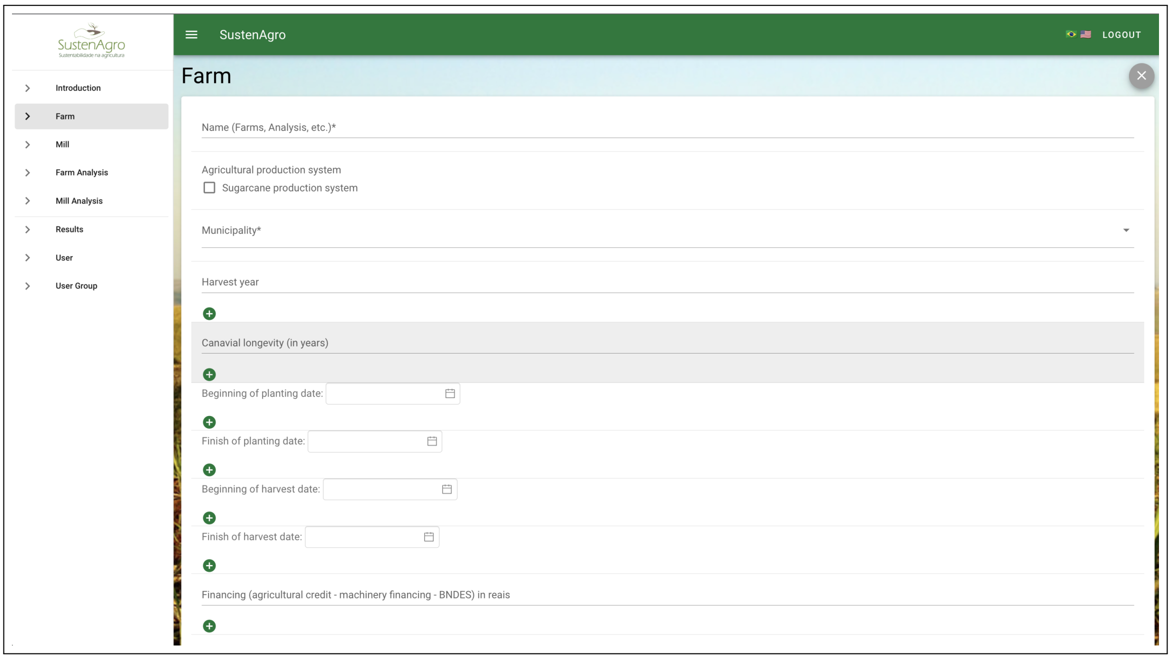


Figure 19 – Form to create a farm.

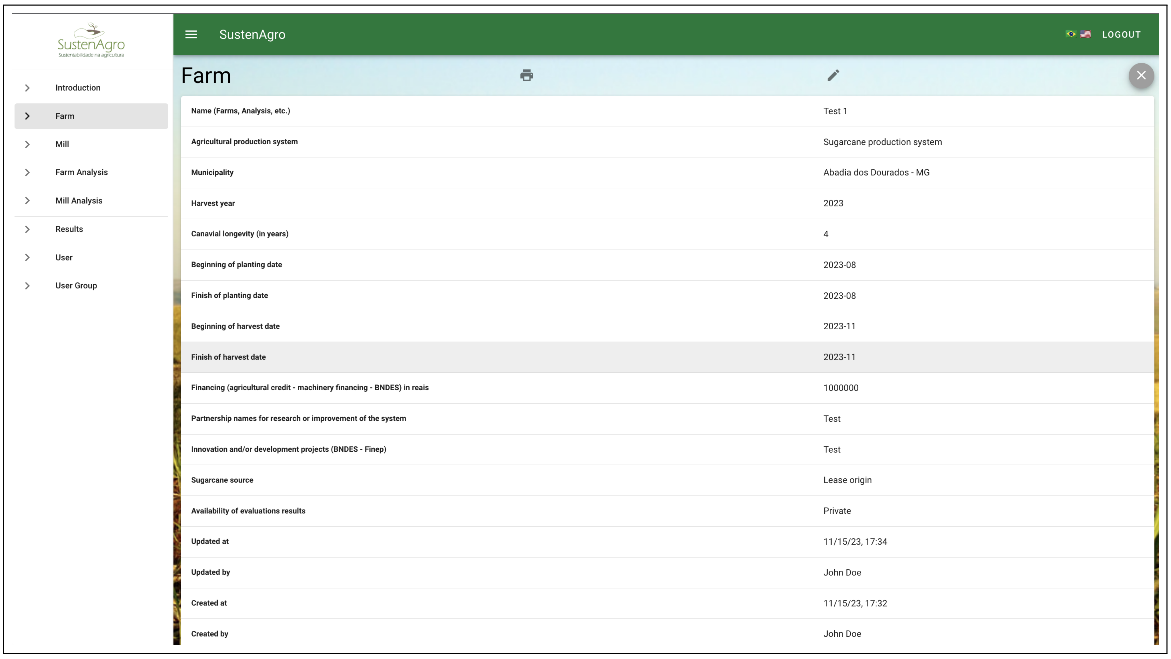


Figure 20 – Farm information.

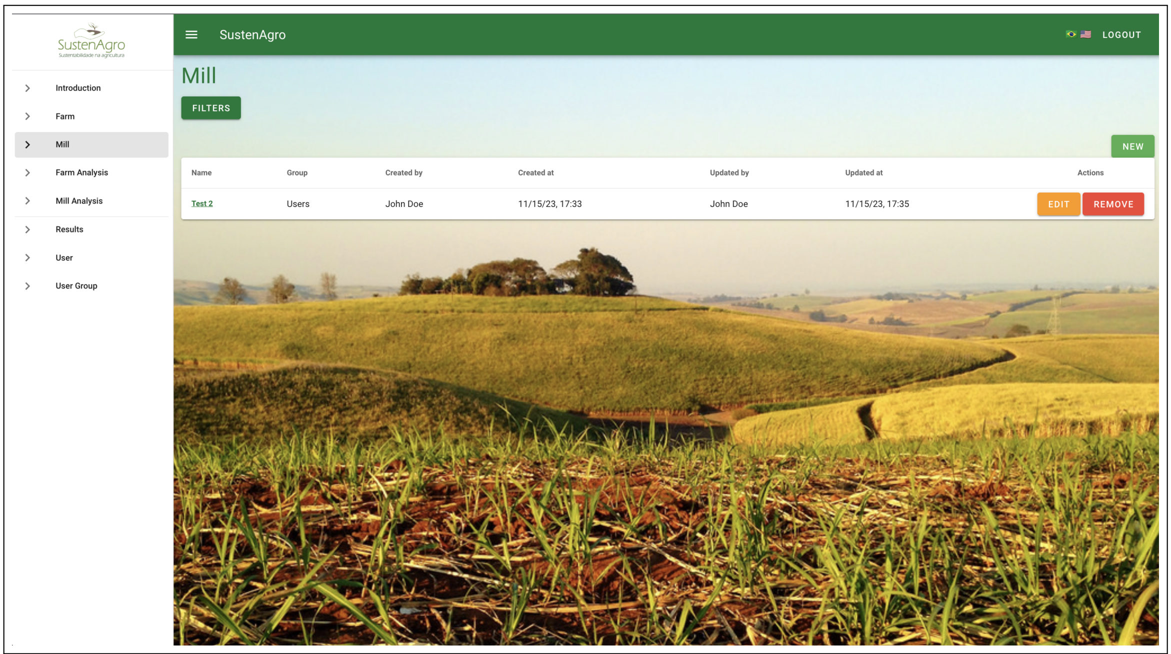


Figure 21 – Mill Page (to list and create a new one).

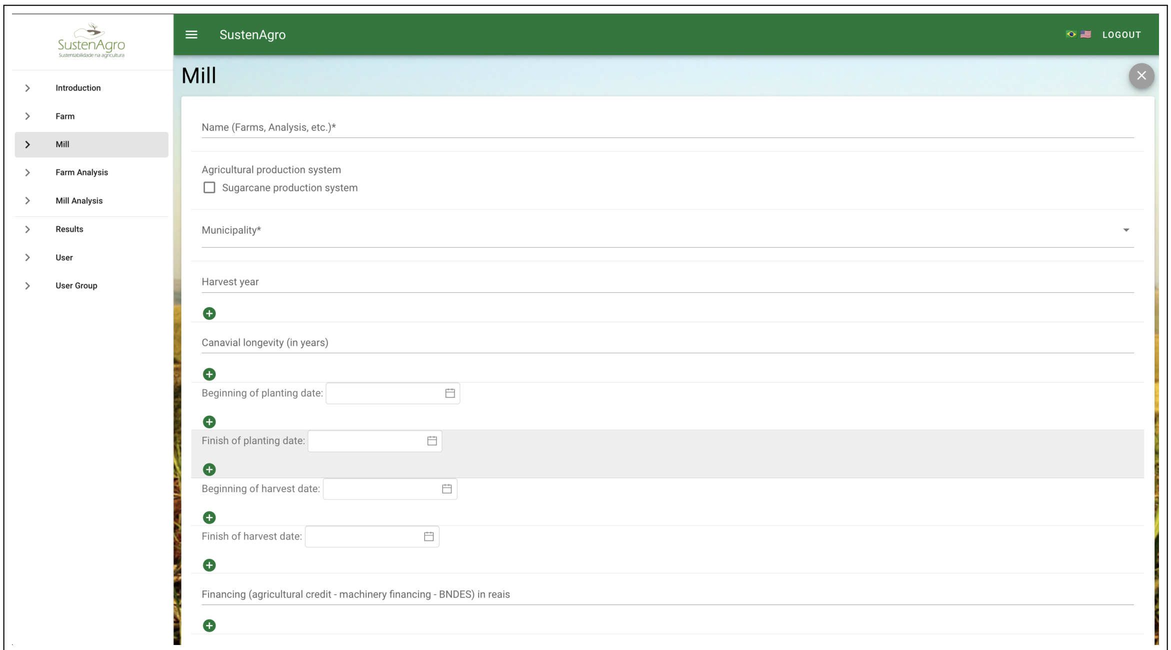


Figure 22 – Form to create a mill.

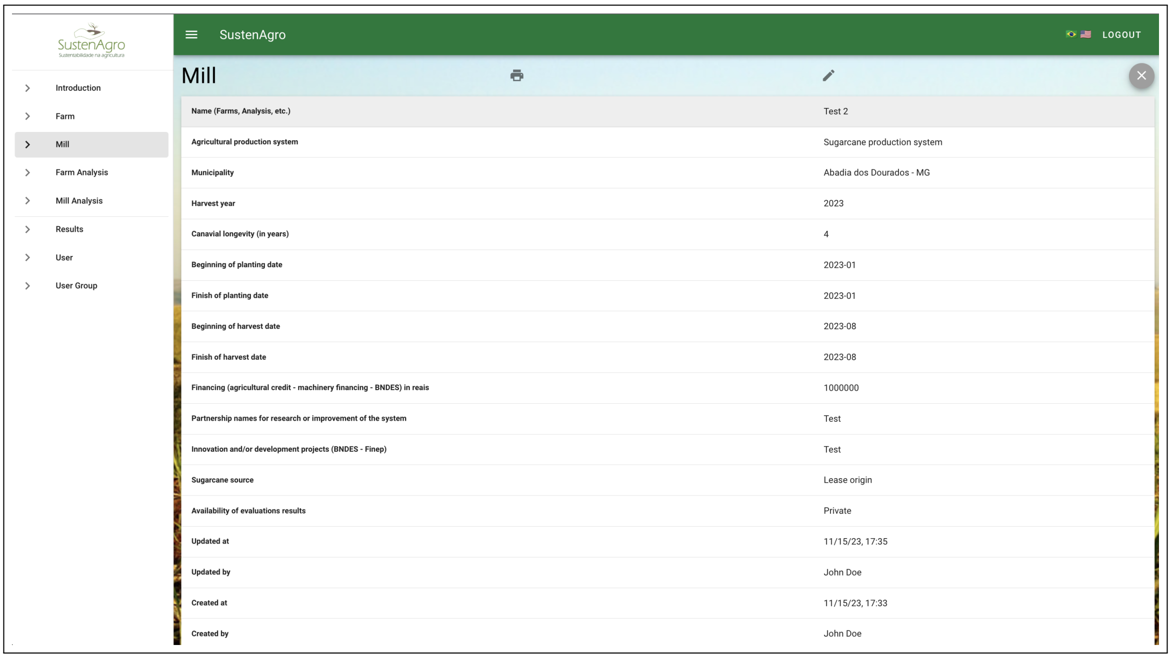


Figure 23 – Mill information.

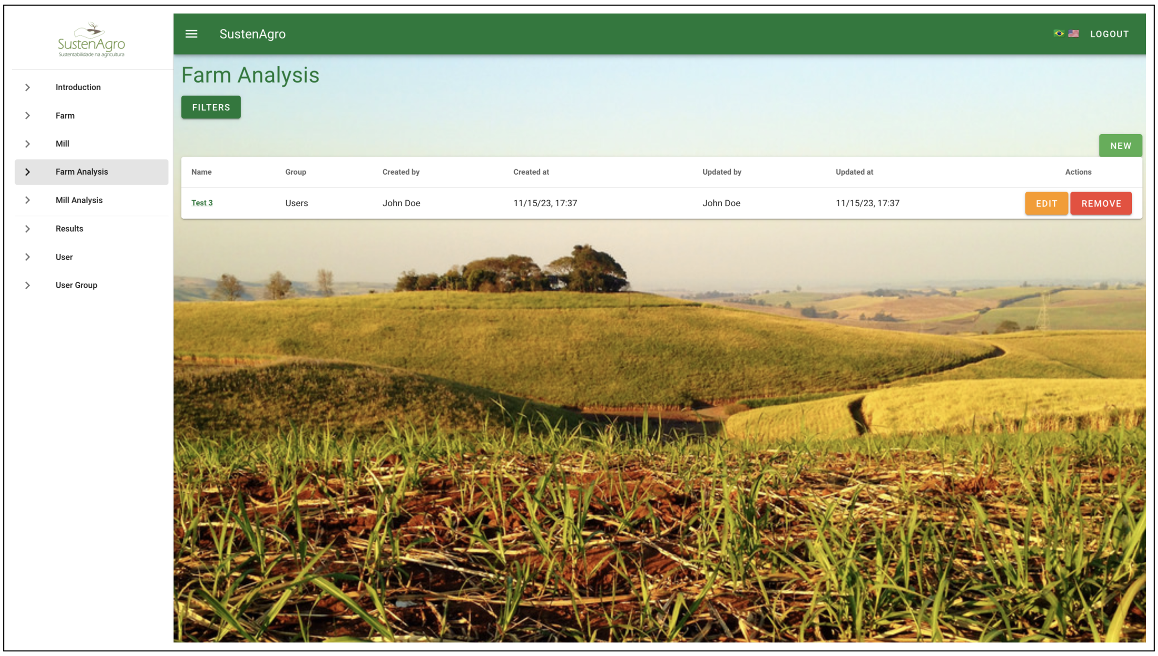


Figure 24 – Farm Analysis page (to list and create a new one).

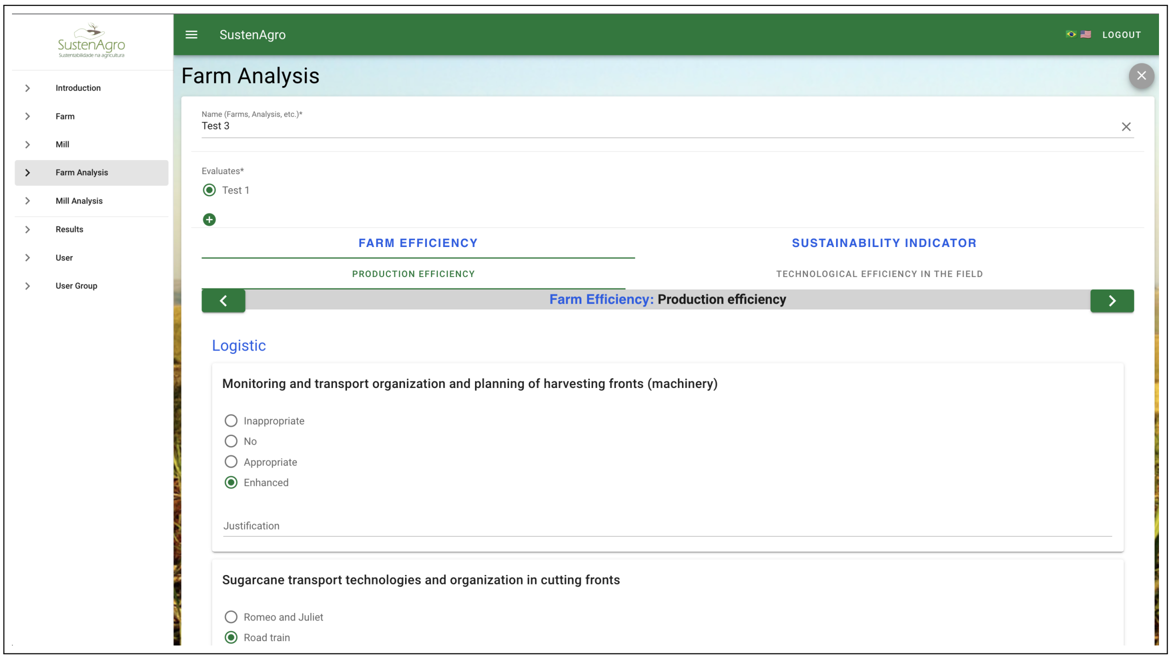


Figure 25 – Form to create a Farm Analysis.

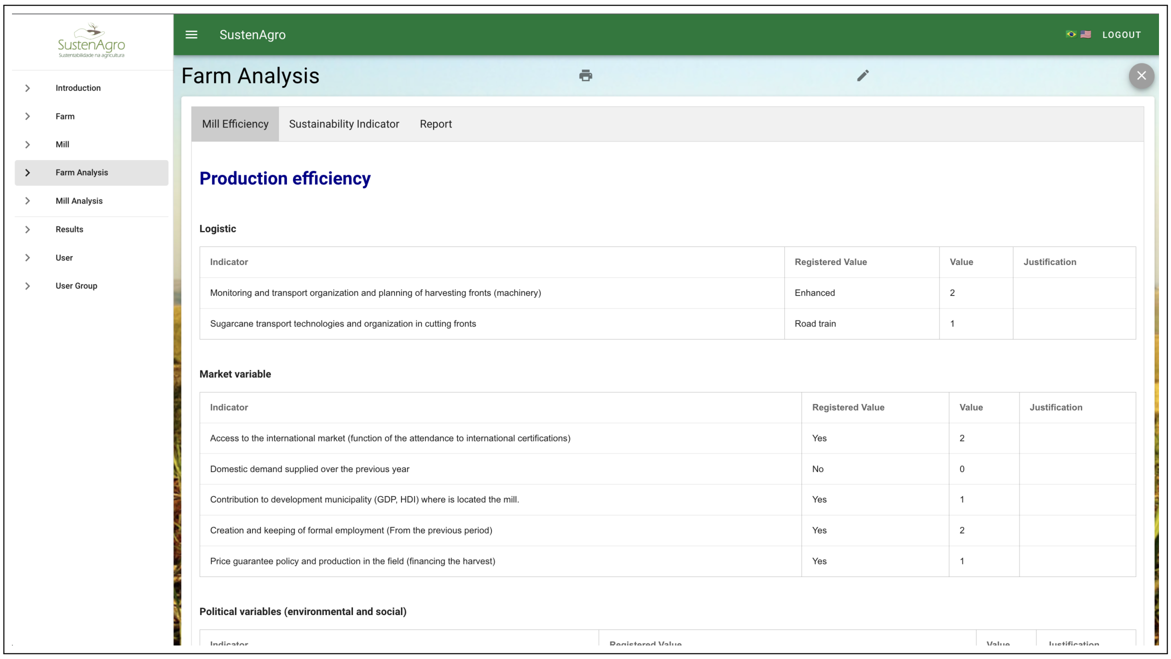


Figure 26 – Farm Analysis Report

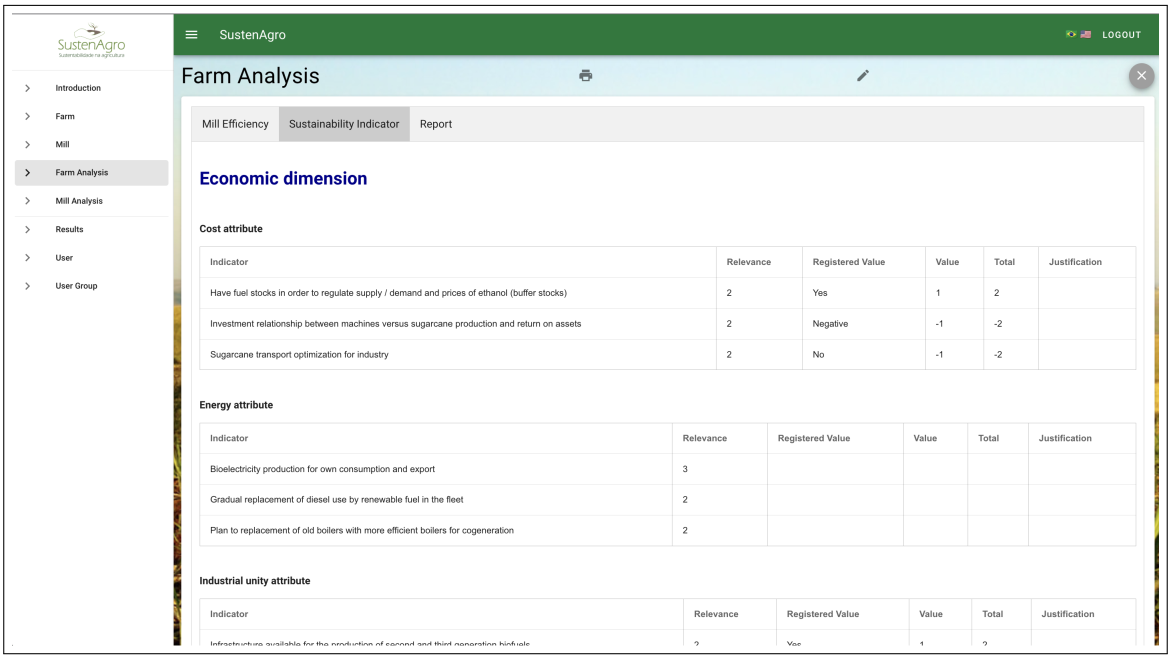


Figure 27 – Farm Analysis Report

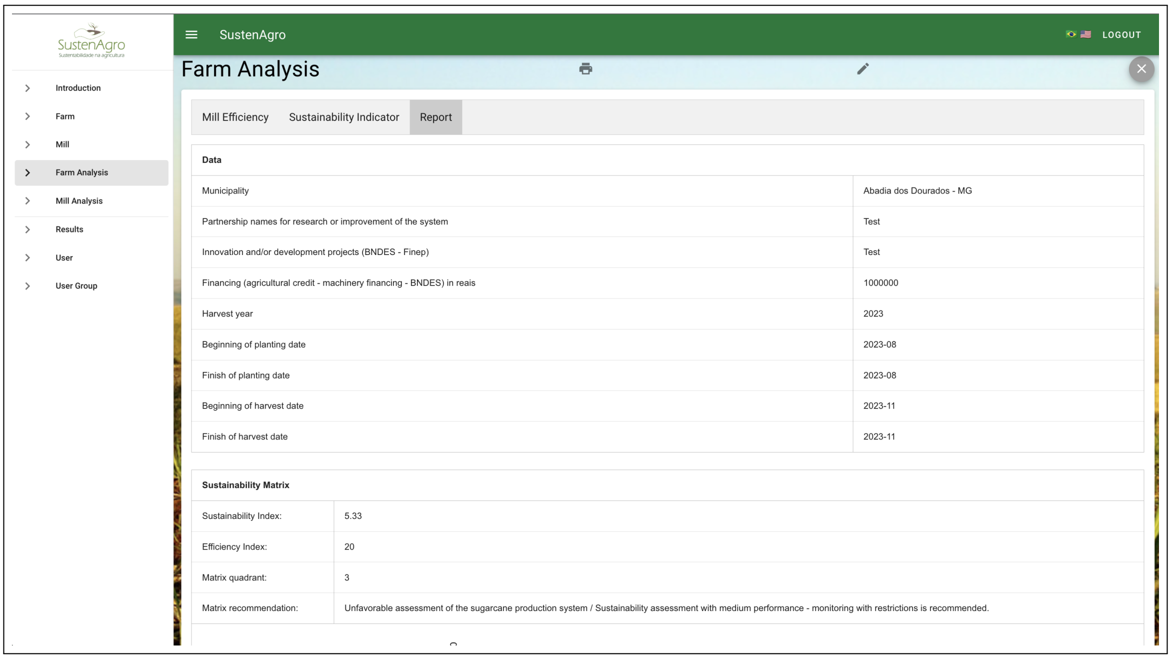


Figure 28 – Farm Analysis Report

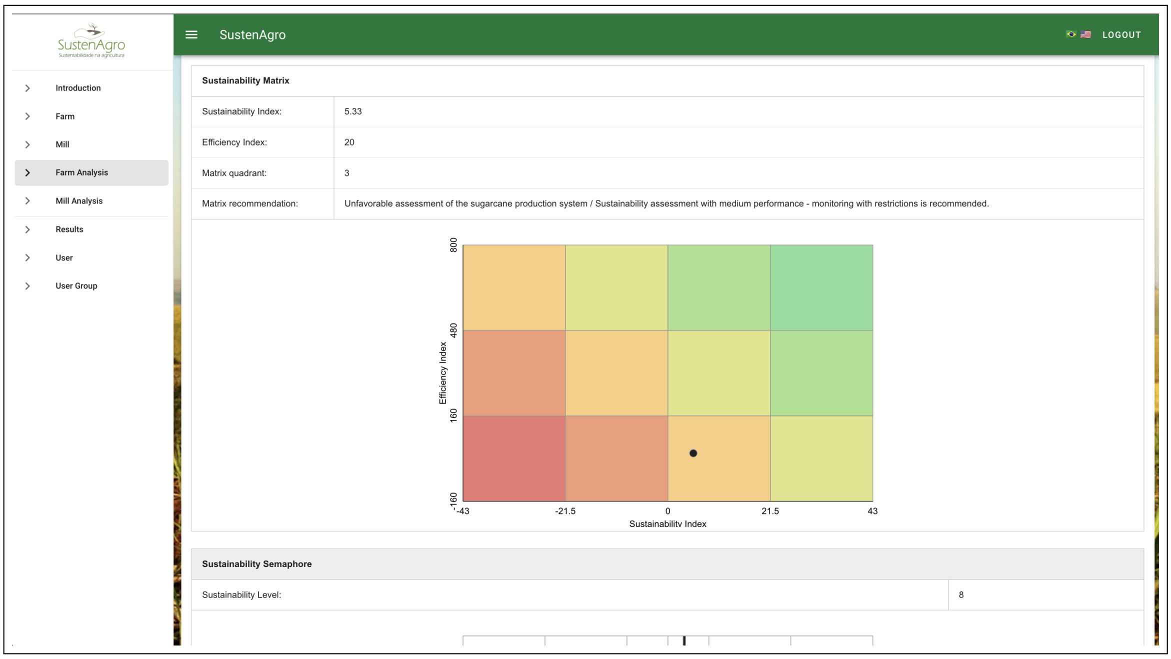


Figure 29 – Farm Analysis Report

Figure 30 – Farm Analysis Report

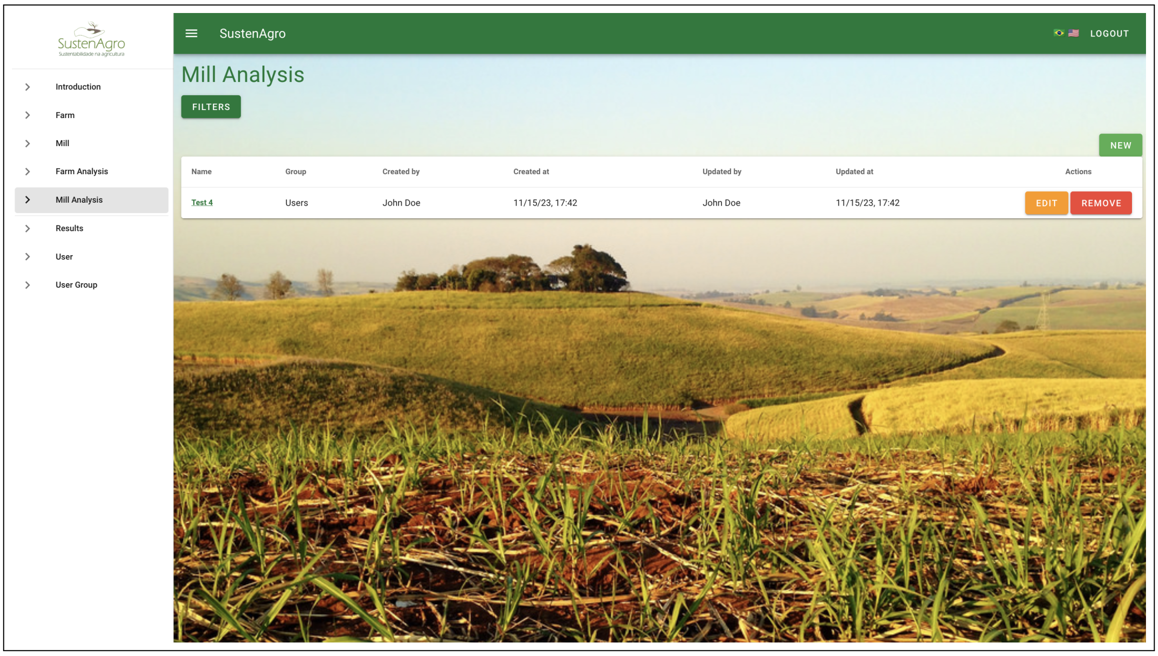


Figure 31 – Mill Analysis page (to list and create a new one).

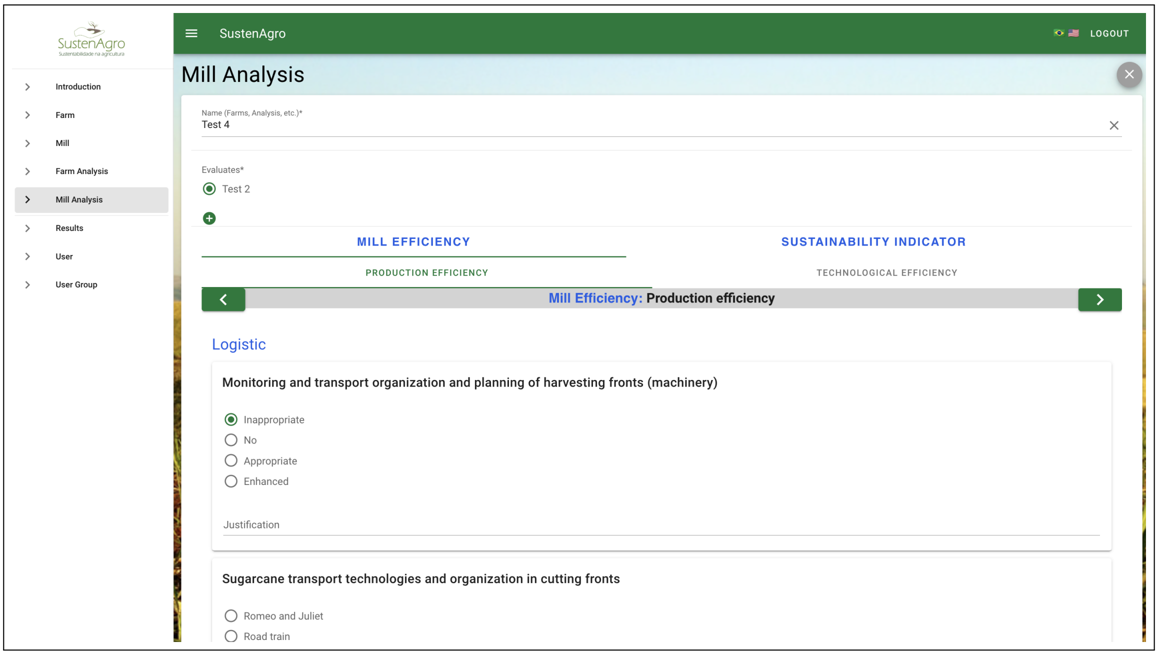


Figure 32 – Form to create a Mill Analysis.

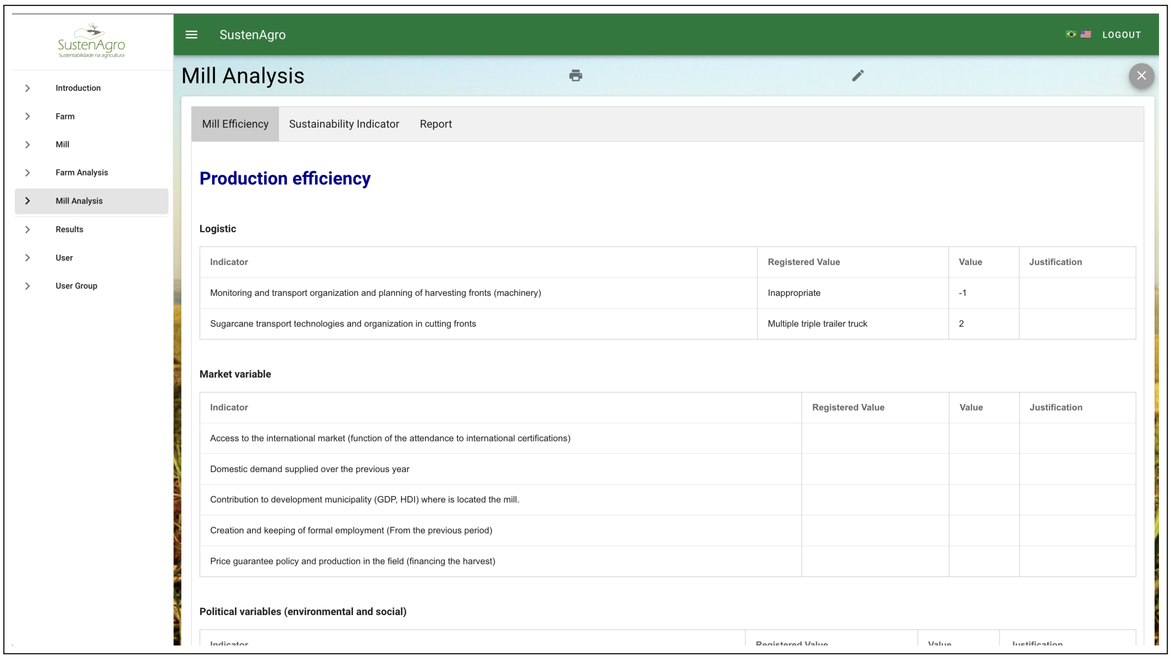


Figure 33 – Mill Analysis Report

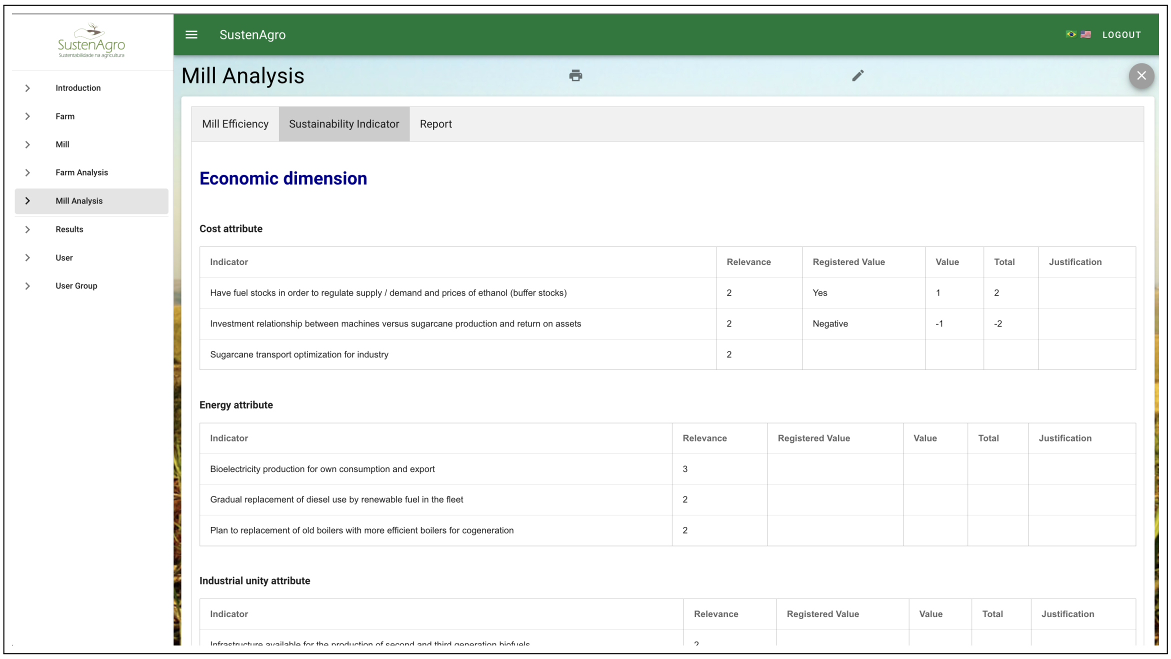


Figure 34 – Mill Analysis Report

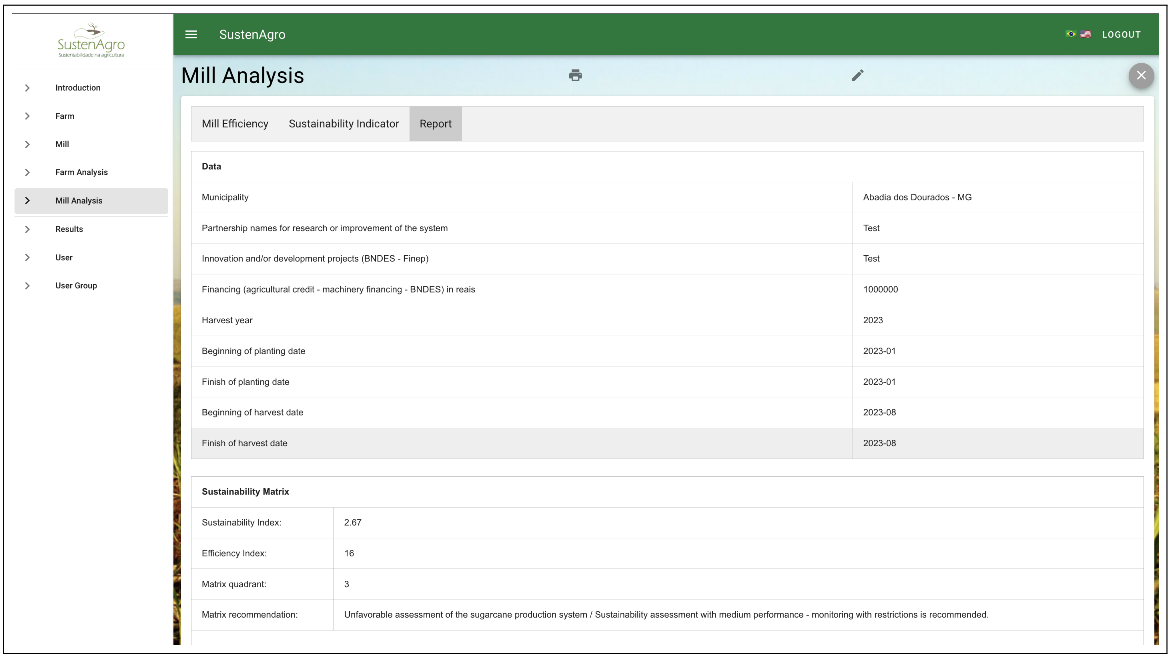


Figure 35 – Mill Analysis Report

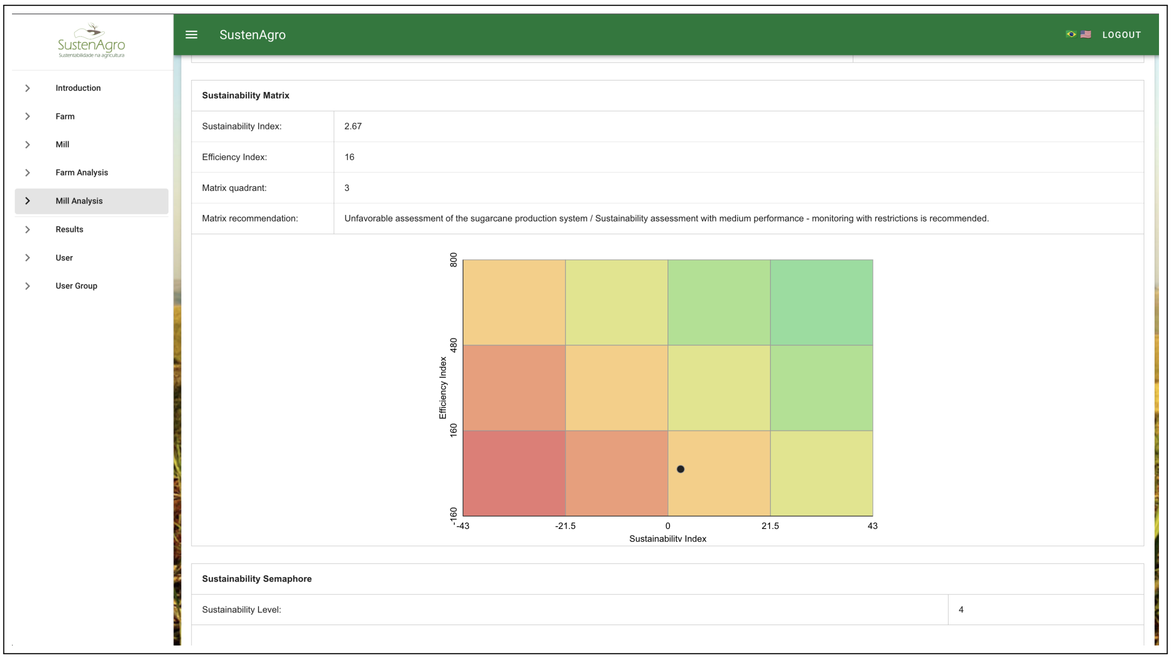


Figure 36 – Mill Analysis Report

Figure 37 – Mill Analysis Report

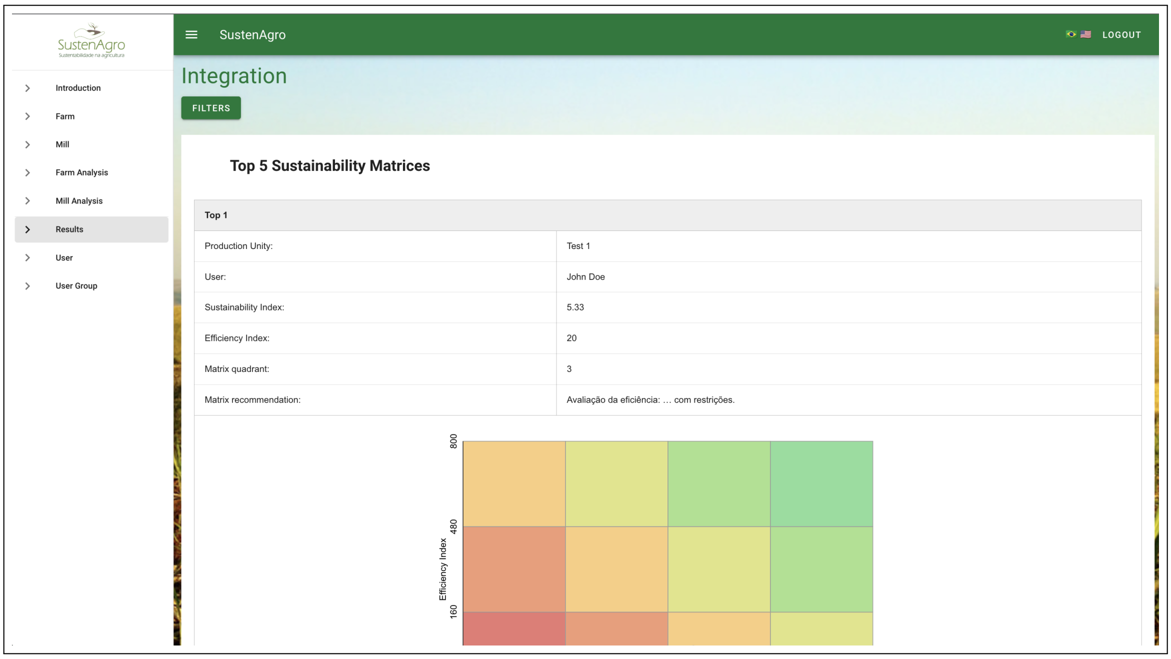


Figure 38 – Results Page that shows the top 5 sustainability matrices.

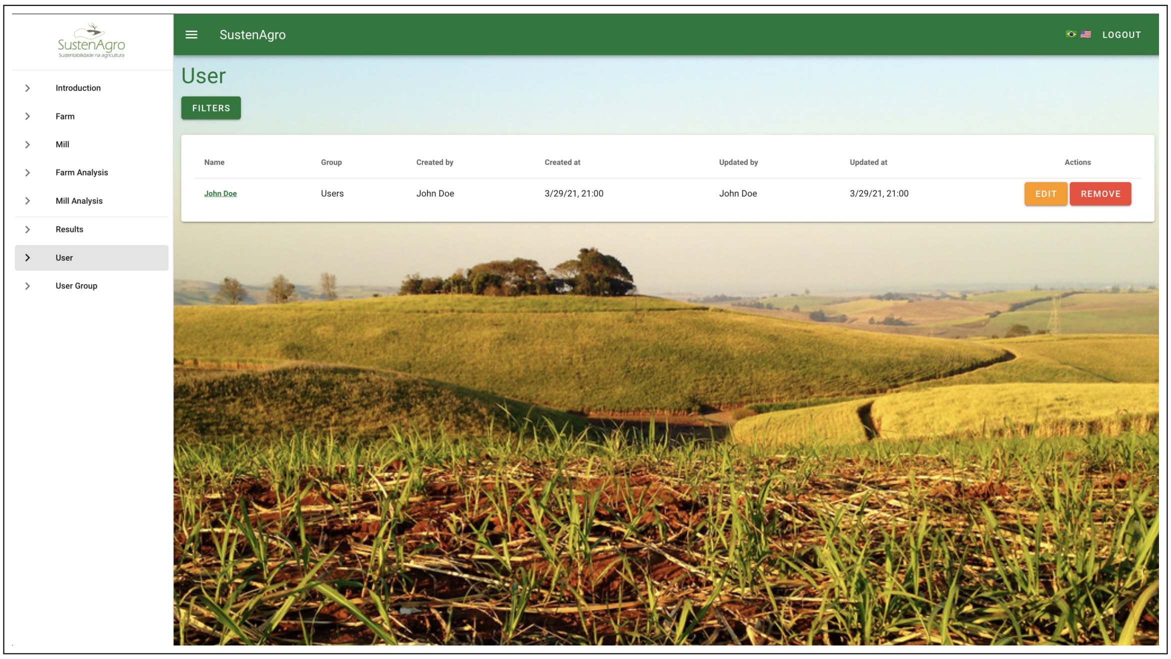


Figure 39 – Page to User information.

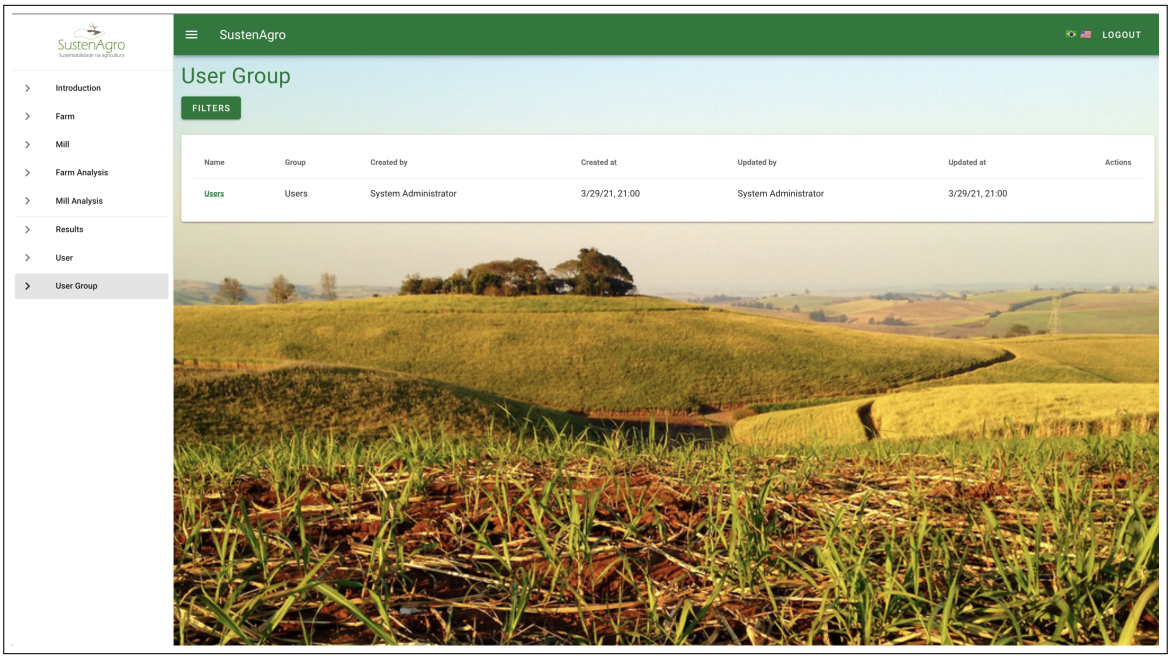


Figure 40 – Page to User Group Information.

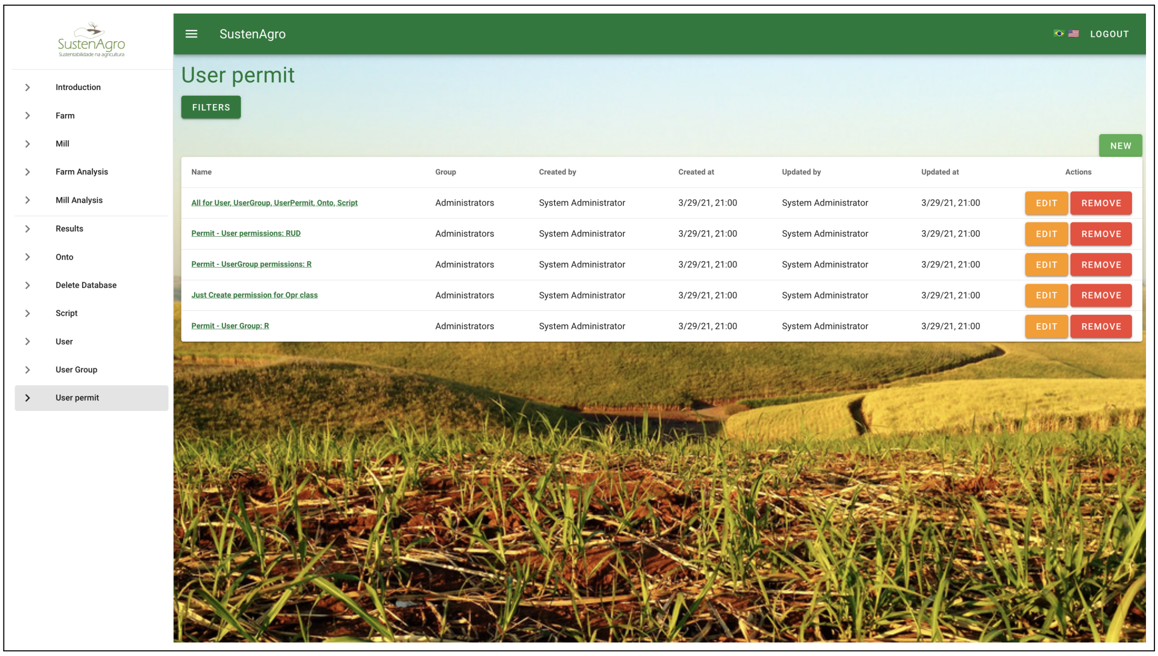


Figure 41 – Page to User Permit (manage user permissions (only available for Domain Experts UI.

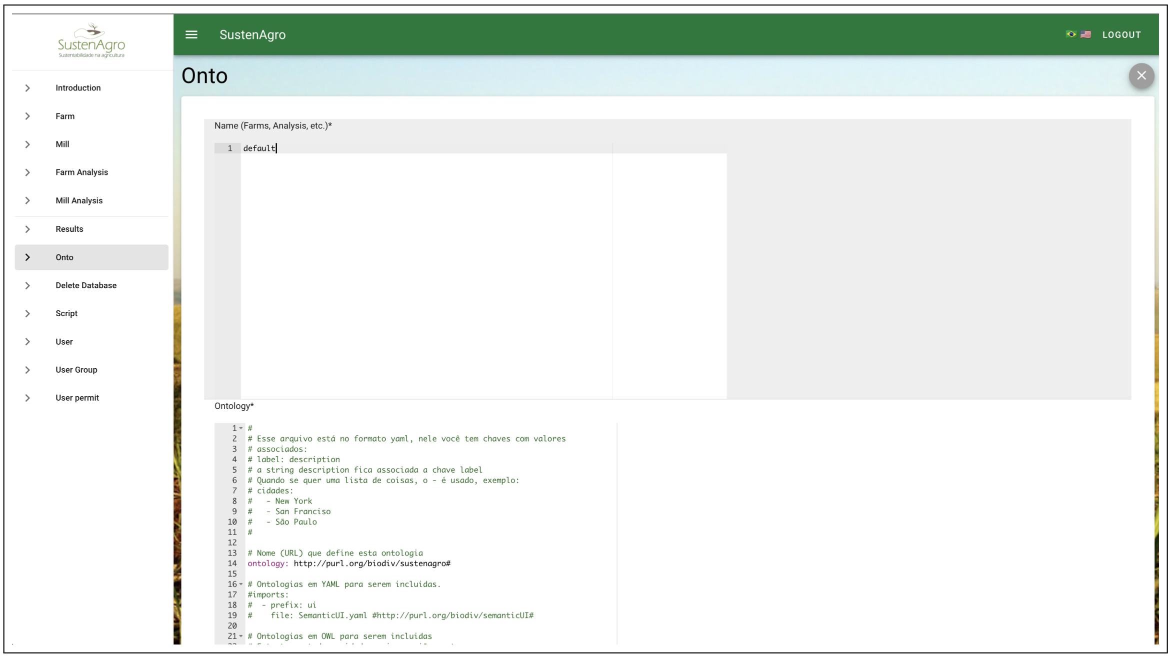


Figure 42 – Ontology Editor (only available for Domain Experts UI.

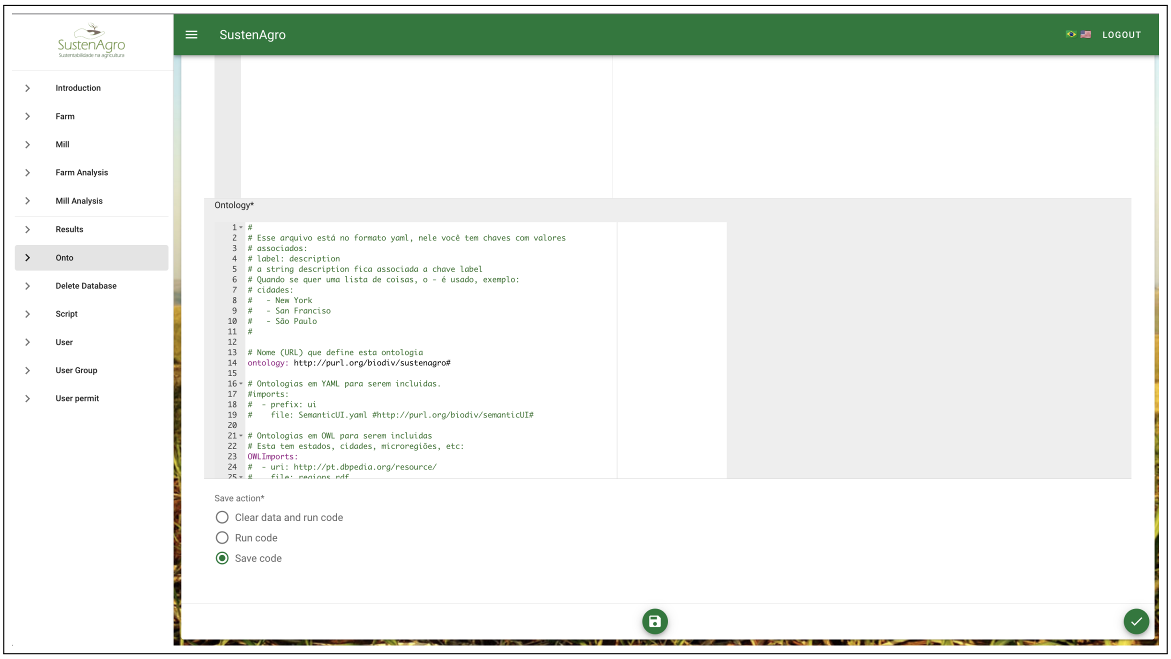


Figure 43 – Ontology Editor (only available for Domain Experts UI.

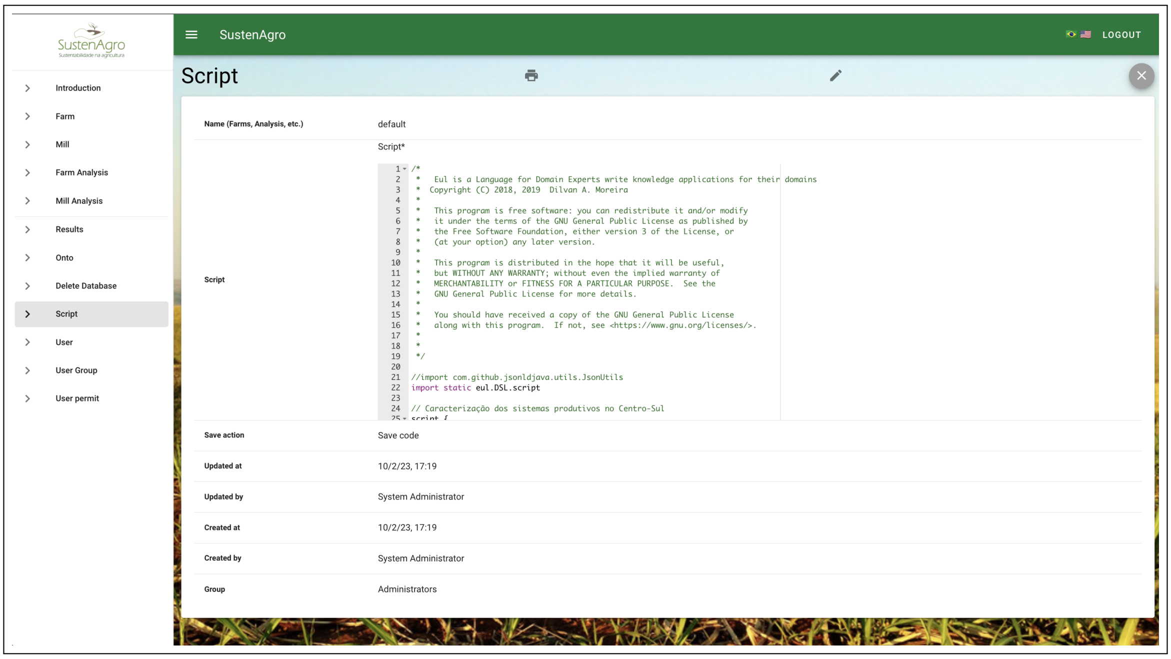


Figure 44 – DSL Editor (only available for Domain Experts UI..

APPENDIX

B

HEURISTICS EVALUATION CONSOLIDATION

As mentioned in Chapter 5, in the next pages you will be able to see the consolidation table for the Heuristics Evaluations done by the 3 specialists from UI/UX area.

Problem ID	Problem Identified	Description of the Problem	Guideline violated	Severity Degree (0 - 4)	Suggestion for problem solution
1	All pages	Misses a footer	Aesthetic and minimalist design	2	Add a footer so that the user does not need to go back to the introduction tab.
2	All pages	Breadcrumb missing to assist the user locate in the system	Visibility of system status	2	
3	All pages	After login the username is not shown anywhere, in a way that it does not make possible if the login is done successfully	Visibility of system status	1	
4	All pages	There are no accessibility resources available	Visibility of system status	3	
5	All pages	Severe degree of protanomaly, the colors are almost identical	Aesthetic and minimalist design	4	
6	All pages	Icons are small	Aesthetic and minimalist design	2	Increase icons and put them more distant
7	All pages	Printing is not working, the system freezes and closes without any warning	Error prevention	4	Fix this
8	All pages	Clicking in the logos is redirecting to other websites	Consistency and standards	2	The pattern is to go back to the main page
9	All pages	Printing and Editing icons are too distant	Aesthetic and minimalist design	2	One option is to decrease the distance in the report and put the icons in the right corner and close the form in the left corner
10	Aside bar	There is no navigation bar of easy access and visualization, all items are in the breadcrumb menu and when it is clicked, it hides	Consistency and standards	1	
11	Aside bar	Clicking on both different logos, the user is led to the partners websites, even as a partnership, it is important to keep the system identity	User control and freedom	2	A system identity is required so the user can return to the main page
12	Aside bar	The format of the Menu does not give the impression that they are going to be expanded because of the arrows	Aesthetic and minimalist design	2	Change the layout not making use of arrows
13	Aside bar	Missing information regarding the available options and functionalities	Help and documentation	2	
14	Aside bar	In the aside menu there are two logos that directs the user out of the system, the ideal location is on a footer	Visibility of system status	2	
15	Top bar	Either in English and Portuguese the "Logout" button remains written in English	Consistency and standards	2	Translate "Logout" to Portuguese
16	Introduction page	The introduction page could have less information	Aesthetic and minimalist design	2	Some contact, address, location information could be located in a footer, and regarding the research group, should be there a tab only for this

Table 4 – Heuristics Evaluation Consolidation. Part 1.

Problem ID	Problem Identified	Description of the Problem	Guideline violated	Severity Degree (0 - 4)	Suggestion for problem solution
17	Introduction page	Names and terms with the aside bar are not standardized, making difficult to be intuitive	Consistency and standards	2	Standardize the terms
18	User page	Box of confirming the user removal is a pop-up from browser	Error prevention	3	The system itself have a pop-up instead of using a browser one
19	Add Farm and Mill pages	The system neither validate the harvest nor planting beginning and finish dates	Error prevention	4	System adjustment to do not happen an error to plant in the future and harvest in the past
20	Add Farm and Mill pages	Items tabulation is tight	Aesthetic and minimalist design	2	Month/year of harvesting and planting can be put side to side
21	Add Farm and Mill pages	Options to add new are below between some items, which can cause some strangeness of its location	Error prevention	2	
22	Add Farm and Mill pages	The word "string" was used to define the place in which a must be inserted a sentence in the search	Match between system and the real world	2	Change the word used
23	Add Farm and Mill pages	Besides, it specifies in years the sugarcane longevity, if the mean is break the system won't accept	Error prevention	3	Specify the format
24	Add Farm and Mill pages	The format is not specified in Brazilian Real, in the financing field	Error prevention	3	Specify the format
25	Add Farm and Mill pages	It would be great if the harvest and planting dates are done in compatible dates	Error prevention	4	Warn if the dates are compatible
26	Add Farm and Mill pages	It only allows the exclusion after filling the calendar field	Error prevention	4	Adjust to be allowed to remove in any scenario
27	Add Farm and Mill pages	It is not asked if the item should be really removed	Help users recognize, diagnose, and recover from errors	4	Warn before removal
28	Add Farm and Mill pages	The content is saved but it is still on the editing screen	Visibility of system status	1	Go to the saved screen and then allow editing
29	Add Farm and Mill pages	The options to "Add New" is between the items to be filled, a strange place for it	Error prevention	2	An option is to put it on the right side
30	Add Farm and Mill pages	There is no information in the "Send" and "Cancel" in the top of the page.	Error prevention	3	It would be better to write "Send" and "Cancel" on buttons to prevent the error
31	Add Farm and Mill pages	It is allowed to save farms and mills with the same names and characteristics	Error prevention	3	Verify in the database which is the ID being used and bring it to the screen to differ the contents
32	Add Farm and Mill pages	Biased system, it shows first cities from Minas Gerais state, and it is not clear that it is possible to type other cities from other states	Recognition rather than recall/ Error prevention	2	Put an open field to type the city and put a separate field for the state

Table 5 – Heuristics Evaluation Consolidation. Part 2.

Problem ID	Problem Identified	Description of the Problem	Guideline violated	Severity Degree (0 - 4)	Suggestion for problem solution
33	Add Farm, Mill, and Analysis Pages	Title of registration is too close to the borders	Aesthetic and minimalist design	1	Adjust framing
34	Add Farm, Mill, and Analysis Pages	Title of registration is too close to the borders	Aesthetic and minimalist design	1	Adjust framing
35	Add Analysis pages	Various tabs inside other tabs confuse the user	Error prevention/ Help users recognize, diagnose, and recover from errors/ Recognition rather than recall	4	It would be better to create vertical tabs instead of horizontal in each item of the tabs
36	Add Analysis pages	Confuse tab buttons, it is not possible to understand in first sight that the top ones are different from the bottom ones	Consistency and standards	2	Think in a different way to show the relations of the forms
37	Add Analysis pages	A field with "Yes" and "No" answers inverted	Consistency and standards	2	Order in a unique way
38	Add Analysis pages	Not-known terminologies	Match between system and the real world	4	Search for the right terminologies or bring explanations
39	Add Analysis pages	Bad formatting in a question	Consistency and standards	1	Adjust formatting
40	Add Analysis pages	There is no warn that shows what has already been filled	Visibility of system status	3	Adjust the exhibition format
41	Add Analysis pages	The "Cancel" button is not shown every time	Consistency and standards	3	Put it in a fixed position
42	Add Analysis pages	Tabs written in blue break the consistency and aesthetics	Consistency and standards	1	
43	Submitting Analysis	Informs an error, but it does not show where they are	Error prevention/ Help users recognize, diagnose, and recover from errors	4	Mark with red and (*) where the errors that need to be corrected are
44	Submitting Analysis	It did not save the filling and showed an internal error	Error prevention	3	
45	Submitting Analysis	It is saving without the need for mandatory items	Error prevention	4	
46	Submitting form	It is allowed to save farms and analysis with symbols	Error prevention	4	Adjust formatting
47	Report page	Tabulation is too big, and there are a lot of screen rolling	Aesthetic and minimalist design	1	Tabulation can be decreased so the user does not need to do much screen rolling
48	Report page	Better explanation of the results	Match between system and the real world	3	Explain in a better way the results
49	Results and Report pages	There are few information to help reading the matrices and the other results	Help and documentation	1	

Table 6 – Heuristics Evaluation Consolidation. Part 3.

Problem ID	Problem Identified	Description of the Problem	Guideline violated	Severity Degree (0 - 4)	Suggestion for problem solution
50	Results page	In the results tab, the title of the page is in English, even selecting the Portuguese version on the system	Match between system and the real world	3	Modify the title so the user can comprehend better what they are seeing.
51	Results page	Filter button does not work in the results tab and there is no information that the user can recover from this error	Help users recognize, diagnose, and recover from errors	2	
52	Results page	The title is "Top 5" even if there is a lesser number of analysis	Consistency and standards	2	Adjust title to give compatibility to the number of analysis
53	Farm, Mill, and Analysis listing	The colors of the "edit" and "remove" buttons are close to each other with severe deutoronomaly	Recognition rather than recall	2	Place a pencil icon to the "edit" button and change its color and place a trash icon to the "remove" button
54	Farm, Mill, and Analysis listing	The previously analysis was not save, even if there was not anything filled, a saving feedback was shown	Error prevention	4	
55	Farm, Mill, and Analysis listing	The "Remove" button should be in a more flashy color	Consistency and standards	1	
56	Farm, Mill and Analysis listing	The items saved with the same name appears in the list without the ID to be identified	Error prevention	2	Verify in the database which is the ID being used and bring it to the screen to differ the contents

Table 7 – Heuristics Evaluation Consolidation. Part 4.