NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MARIA FILIPA DINIS NETO

BSc in Computer Science

USABILITY EVALUATION OF ENTITY RELATIONSHIP MODELLING TOOLS

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
September, 2025

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

USABILITY EVALUATION OF ENTITY RELATIONSHIP MODELLING TOOLS

MARIA FILIPA DINIS NETO

BSc in Computer Science

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, NOVA School of Science and Technology, NOVA University Lisbon

Co-adviser: Paulo Carreira
Associate Professor, Instituto Superior Técnico, University Lisbon

Examination Committee

Chair: Teresa Isabel Lopes Romão
Associate Professor, NOVA School of Science and Technology, NOVA University Lisbon

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, NOVA School of Science and Technology, NOVA University Lisbon

Member: Cristiano de Faveri
Assistant Professor, Escola Superior de Tecnologia de Setúbal, Instituto Politécnico de Setúbal

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
September, 2025

Usability Evaluation of Entity Relationship modelling tools

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.10) [32].

To my father, who gave me strength when I had none, and to my sister Leonor, whose guidance lit the way.

Acknowledgements

I would like to express my gratitude to my adviser, Vasco Amaral, for his guidance and support throughout this work.
A very special thank you goes to my father, who never gave up on me, even in the moments when I felt like giving up myself. His encouragement and persistence were essential to the completion of this dissertation.
I would also like to thank my mother for giving me life, staying by my side, and pushing me to see this project through.
I am also deeply grateful to my sisters Inês and Leonor for their constant support during this journey. In particular, I thank my sister Leonor, whose experience with experimental research was invaluable in helping me with the statistical analysis and interpretation of results.
A heartfelt thank you as well to Sara, my closest friend, who was developing her own thesis at the same time. Sharing the struggles and challenges of this process with her made the journey far less lonely.
I am deeply grateful to Xavier, my boyfriend, for never leaving my side throughout this journey, and encouraging me to be better.
Finally, I thank all those who, in different ways, encouraged me, believed in me, and accompanied me through this process.

Abstract

Entity-Relationship (ER) modelling is a cornerstone of database design, yet the tools available to support this practice vary widely in notation fidelity, usability, and functionality. This dissertation set out to evaluate the usability of existing ER modelling tools, while also exploring the feasibility of developing a more complete solution combining Chen's notation, diagram validation, SQL code generation, and an intuitive interface.

Through an experiment, we compared three existing ER modelling tools: ERDPlus, Draw.io, and BigER. The choice of these tools was made through a study of the current state-of-the-art ER modelling tools available, conducted in a Systematic Literature Review. Twenty participants, including students and software developers, modelled diagrams using each tool and completed usability surveys based on the System Usability Scale (SUS). Task completion times and qualitative feedback were also collected. The study followed the Goal-Question-Metric approach, addressing functional usability, user experience, and effort. The results revealed that none of the evaluated tools fully satisfied all key requirements.

This dissertation contributes empirical evidence and methodological insights to the evaluation of ER modelling tools and highlights the need for further development of an integrated solution that bridges conceptual accuracy, validation, and practical code generation.

Keywords: ER diagrams, Model-driven engineering, Modelling tools, Database design, Usability

Resumo

A modelação de diagramas Entidade-Relação (ER) é um pilar fundamental no desenho de bases de dados, mas as ferramentas disponíveis para apoiar esta prática variam bastante em termos de fidelidade à notação, usabilidade e funcionalidade. Esta dissertação teve como objetivo avaliar a usabilidade de ferramentas de modelação ER existentes, explorando igualmente a viabilidade de desenvolver uma solução mais completa que combinasse a notação de Chen, validação de diagramas, geração de código SQL e uma interface intuitiva.

Através de uma experiência, foram comparadas três ferramentas de modelação ER: ERDPlus, Draw.io e BigER. A escolha destas ferramentas foi feita a partir de um estudo do estado da arte de ferramentas de modelação ER, conduzido por uma Revisão Sistemática de Literatura. Vinte participantes, incluindo estudantes e software developers, construíram diagramas em cada ferramenta e completaram inquéritos de usabilidade com base na System Usability Scale (SUS). Foram também recolhidos tempos de execução das tarefas e feedback qualitativo. O estudo seguiu a abordagem Goal-Question-Metric, avaliando usabilidade funcional, experiência do utilizador e esforço. Os resultados mostraram que nenhuma das ferramentas avaliadas satisfaz plenamente todos os requisitos.

Esta dissertação contribui com evidência empírica e contributos metodológicos para a avaliação de ferramentas de modelação ER, e sublinha a necessidade de desenvolver uma solução integrada que una precisão conceptual, suporte à validação e geração prática de código.

Palavras-chave: Diagramas ER, Engenharia orientada a modelos, Ferramentas de modelação, Desenho de bases de dados, Usabilidade

Contents

List of Figures

List of Tables

Acronyms

CIM computation-independent model *(p. 11)*

DBMS Database Management System *(pp. 20, 21, 23, 24, 45)*
DSL Domain-Specific Language *(pp. 7, 8, 11)*
DSM Domain-Specific Modelling *(p. 11)*

EMF Eclipse Modelling Framework *(p. 56)*
ER Entity-Relationship *(pp. v, 1–3, 5, 14–19, 23–26, 28–31, 33, 45, 49, 52, 57–59)*
ERD Entity-Relationship Diagram *(pp. ix, 1–3, 5, 14, 15, 19, 23–25, 44, 45, 50, 51, 53, 55, 58)*

IDE Integrated Development Environment *(p. 7)*

MDA Model-Driven Architecture *(pp. 10, 11)*
MDD Model-Driven Development *(pp. 9–11, 14, 17)*
MDE Model-Driven Engineering *(pp. ix, 9–11)*

OCL Object Constraint Language *(pp. 12, 50)*
OMG Object Management Group *(pp. 10, 12)*

PICOC Population, Intervention, Comparison, Outcome, Context *(pp. 14, 15)*
PIM platform-independent model *(p. 11)*
PSM platform-specific model *(p. 11)*

SLR Systematic Literature Review *(pp. 2, 3, 14–17, 25, 57, 59)*
SQL Structured Query Language *(pp. 2, 19–26, 31, 35, 36, 44, 45, 50, 58, 59)*
SUS System Usability Scale *(pp. v, 29, 33–36, 43, 47, 57, 59)*

UML Unified Modelling Language *(pp. 1, 4, 5, 17, 19, 21–23, 25, 59)*

1 Introduction

1.1 Motivation

In the context of computer science engineering and education, encountering Entity-Relationship (ER) modelling is almost inevitable. Since its introduction by Chen in his seminal work "The Entity–Relationship Model - Toward a Unified View of Data" [8], ER modelling has been established as a central technique for representing the semantics of the real world in conceptual data models.
There has been further development within the scope of data modelling languages, and one example is the Unified Modelling Language (UML), which consists of a variety of diagram types, for different software design purposes. UML's Class Diagrams can be considered an alternative to the Entity-Relationship Diagram, since the terminology of the Class Diagram can be mapped to the terminology of the ERD [11]. However, the Class Diagram may be seen as more complex to design than the ER diagram, given that its language has more of an overview over the whole software, whereas ER modelling is more focused on database modelling [7]. In general, the ERD is simpler to describe a database, for software developers and especially in an educational context.
With that, ER modelling remains a cornerstone of both industry practice and academic curricula, serving as a foundation for database design, communication between stakeholders, and conceptual clarity [5][11].
ER modelling captures the relationships between the entities involved in a system, culminating in the creation of an Entity-Relationship Diagram (ERD). These diagrams hold significant importance, as they often serve as the blueprint guiding database design and implementation. Their relevance is reflected not only in academic instruction but also in professional standards. For example, the ACM/IEEE Computer Science Curricula (CC2020) explicitly include conceptual modelling and database design as fundamental competencies for computer science and software engineering graduates [16].
While it is possible to sketch ERDs by hand, this practice is increasingly impractical for real-world scenarios. Instead, a wide range of ER modelling tools has emerged, offering

graphical interfaces that simplify diagram creation and maintenance. Beyond visualization, more advanced needs arise in the context of database development - notably validation of diagrams and the automatic generation of SQL scripts. Tools that integrate these features not only support correct database implementation but also reduce ambiguity, time, and errors in the development process. [40]
This dissertation aims to investigate the current state-of-the-art in ER modelling tools and to conduct a thorough empirical experiment with human participants. The study focuses on the usability and effectiveness of selected tools, paying particular attention to functionalities such as diagram validation and SQL code generation. By combining both quantitative and qualitative analyses, this work seeks to provide insights into how these tools support users in bridging the gap between conceptual modelling and practical database development.

1.2 Problem Statement

The modelling of the Entity-Relationship Diagram (ERD) is an essential step and should be executed in the cleanest way possible. In the academic context, as well as in the context of building a new application or database, a strong base for data modelling is needed, and so, a strong tool for ER modelling is very likely to contribute positively, focusing on the entities and relationships that may exist within the software.
A strong and complete tool to model Entity-Relationship Diagrams would be a tool that is pleasant to use and intuitive, while offering functionalities useful for consequent database development, such as diagram validation and SQL code generation. With that in consideration, the main research question of this dissertation is *"What are the current offers on Entity-Relationship modelling tools and are they complete?"*.
In order to answer our research question, first it was acknowledged what defines ER modelling, the advantages of adopting a model-driven approach in the development stage, and, through a Systematic Literature Review (SLR), what is currently available regarding this area of interest was discovered and studied.
The problem is that currently, as it will be seen further on, the state of the art in this area is considered poor, in terms of offers for ER modelling tools.
The goal of this dissertation is to study what currently exists in this area, and create awareness to what could be necessary to further implement and enhance it.

1.3 Methodology and Contributions

The adequate methodology that this dissertation will follow will be an experiment (with human participants), following the guidelines in [24].
While traditionally, a scientific research would be conducted only by observing and analysing, without any modification, nowadays the idea behind researching is to invoke some kind of change and measure the impact of it. Basing engineering on scientific

knowledge - tested through experiments - helps ensure the reliability of what is built. The main benefit of scientific knowledge is its predictability.[26]
With this methodology, the objective is to evaluate in depth what can currently be used to model Entity-Relationship Diagrams in terms of tools and what can be missing within those tools, that could be helpful in order to facilitate the user's process while building the diagrams and for possible further developments, such as database construction.
As for contributions, this dissertation involves a Systematic Literature Review around the state-of-the-art of ER modelling tools, as well as a feasibility study around the development of an ER modelling tool.

1.4 Document Structure

The document is structured overall according to the guidelines in [24], with some minor adaptations to adjust to the content of this dissertation.
Firstly, in the *Background* chapter (Chapter 2), some crucial concepts will be explained in detail in order to give some context of what are the topics of this dissertation. Followed by a *Related Work* chapter (Chapter 3), where a Systematic Literature Review is conducted in order to thoroughly identify the state of the art for this dissertation, and the tools to perform the Experiment are chosen. After these introductory chapters, the *Experiment* chapter (Chapter 4) introduces the actual experiment, with its planning, execution, result analysis and interpretation, followed by a *Feasibility Study using a Cloud-based MDE Modelling Workbench* (Chapter 5) conducted within the scope of the development of an Entity-Relationship modelling tool. The *Conclusion* chapter (Chapter 6), synthesises and brings this dissertation to a close.

2

Background

2.1 Entity-Relationship Modelling

Entity-Relationship (ER) Modelling is a modelling language originally invented by Peter P.S. Chen. The idea for this ER Model arose from the need to model data more naturally and intuitively while combining the existing data models' advantages. This model combines the natural way of separating entities and relationships of the Network Model while also taking advantage of the Relational Model, which is known for achieving high data independence. Finally, the Entity Set Model provided the set theory, which would become very applicable in this modelling approach since it can again offer data independence. With that, Chen's proposal became a strong suit for database design [8].

ER Modelling can also ease the enterprise view of data [9]. Since the output for the database design process - the user schema - is not a pure representation of the real world and can become quite complex and not so simple to interpret, it is well thought to model the enterprise schema - which consists in defining the general build of the project - with the ER model. This approach can simplify communication between the related parties while also easing the interpretation for the ones who didn't acquire the specific knowledge on the matter.

ER modelling has been utilised for over 40 years, which shows its relevance and also means there is more empirical data for analysis. This modelling language has undergone many changes over the years derived from data modelling needs and has evolved to better adapt to them [38].

The entity-relationship diagram (ERD) is the technique to follow when there is the desire to exhibit the entities and relationships involved in the ER model. It helps to visualise how things are connected and understand what dependencies may emerge during the data modelling and database design processes.

ER modelling has had its appeal for some time now. However, according to Cabot [7], ER models became less relevant in recent years partly because of the Unified Modelling Language (UML). With the arrival of UML, which models a wide variety of diagrams defining several software development processes, the ER model became almost replaceable.

However, as mentioned in Section 1.1, UML could not match the overall simplicity of the Entity-Relationship Diagram and its construction.

Concept	**Description**	**Concrete Syntax**
Entity	Represents a real-world object or concept	Entity
Relationship	Connects two entities	Rel
Attribute	Describes a property of an entity or relationship	Attr
Primary Key	Attribute that uniquely identifies an entity	Key
Weak Entity	Entity that depends on another entity	Weak
Weak/Identifying Relationship	Links a weak entity to its owner	Rel

Table 2.1: Concrete syntax of original Chen's ER notation

It is important to note that the ER model has suffered some adaptations and has been extended, where new features were developed. The original ER model from Chen dealt with more structural representations such as entities, relationships and attributes.
Entities are used to represent instances of information, and each entity has characteristics, which are called attributes. Entities may or not have an identifiable attribute, called a key attribute, and if they don't have it, they are known as weak entities, as opposed to strong entities which contain a key attribute. Entities are represented by boxes/rectangles with their name inside it, as can be seen in table 2.1. Weak entities are represented with a double line. Weak entities are mostly used when the strong entities or relationships become overloaded with attributes. [44]
To associate and connect entities with one another, relationships are used, which are represented by a diamond shape, with the name of the relationship inside it, as we can see in Figure 2.1 and in table 2.1.
As stated previously, Chen's ER model suffered some adaptations throughout the years, and there are now more elements to it. For instance, *n-ary* relationships were an addition, since from the original Chen's model, only binary relationships were permitted.
Aggregations (grouping) were also an extension from the original ER model, used to enable the association between relationships. In this way, the group is created and can be abstracted from its components, to then be associated with an outside relationship. An example can be seen in Figure 2.2.
Generalisations/Specialisations are another extension for Chen's original model, and

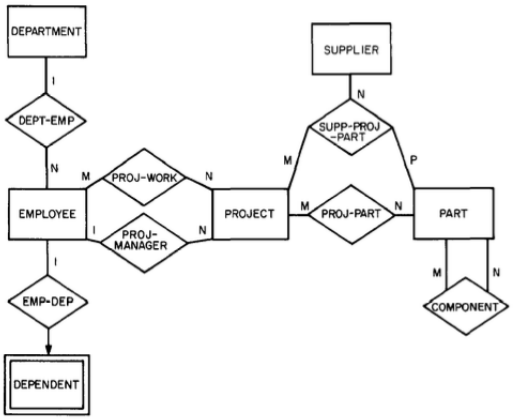


Figure 2.1: Example ER Diagram with the original Chen's notation [8]

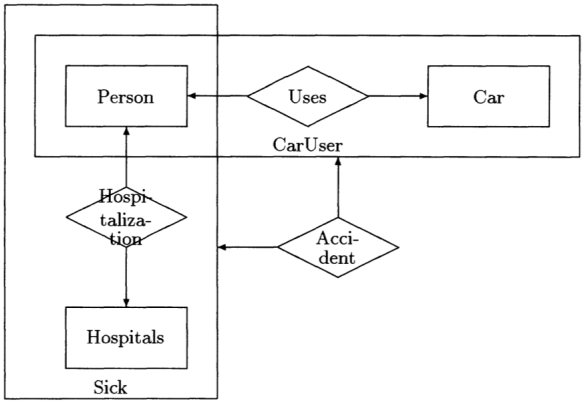


Figure 2.2: Example ER Diagram with Aggregation (grouping) [44]

Figure 2.3: Example Participation and Multiplicity in a relationship between two entities [44]

they are a type of abstraction where one entity type is a more specific or general version of another. These specialisations can be optional, meaning the parent entity may exist without being specified to a child entity; mandatory, meaning the parent entity must be at least one of the children entities; or disjoint, meaning it is only one of the children entities. There are also some constraints regarding the multiplicity and participation of the entities in its relationships/associations. Depending on many factors, one may choose to model a relationship between two entities which is mandatory on one side and optional on the other, while one of the participation types is for multiple instances and another is for at most one, for example, as we can see in Figure 2.3.
In summary, the ER Model can serve as a gateway to software development by modelling the conceptual projections of the project. When built and evaluated correctly, the ER Model is complete, concise, and understandable by all parties involved [27].

2.2 Domain-Specific Languages

A Domain-Specific Language (DSL) can be defined as a type of programming language designed to address a narrow, well-defined problem area, offering only the features necessary for that domain. According to Martin Fowler [20], the definition of a DSL can be understood through four main aspects:

- **Computer programming language**: A DSL is a language that people use to give instructions to a computer. Like other modern programming languages, it is structured to be readable and understandable by humans while remaining executable by machines.
- **Language nature**: As a programming language, a DSL should feel coherent and fluent. Its power comes not only from individual statements but also from how those statements can be combined to form meaningful programs.
- **Limited expressiveness**: Unlike general-purpose languages, which support many kinds of data, control flow, and abstraction mechanisms, a DSL deliberately limits its capabilities. This restriction makes it easier to learn and apply. A DSL is not meant to implement an entire application; instead, it is used to handle a specific concern within a larger system.
- **Domain focus**: The usefulness of a DSL depends on its clear emphasis on a particular domain. This focused design is what gives a limited language its value and effectiveness.

Domain-Specific Languages can be divided into three main categories: external DSLs, internal DSLs and language workbenches. [20]
External DSLs are languages which are defined independently from the main programming language of the application it is supporting. Such languages typically employ a dedicated syntax, although it is also common for them to adopt the syntax of an existing language. Programs written in an external DSL are generally processed by the host application through text-parsing mechanisms, which interpret the DSL specifications at runtime or during a preprocessing stage.
Internal DSLs use general-purpose languages in a specialized manner to handle one small aspect of the overall system. Code written in an internal DSL is valid in the host language, but it relies on only a limited set of language features and follows a particular style to address a specific part of the overall system. The goal is for the code to resemble a custom language rather than typical code written in the host language.
Language workbenches are dedicated IDEs used to create DSLs. They are not only used to define the structure of a DSL but also to provide a dedicated editor in which users can write DSL programs. As a result, the language and its editing environment are closely integrated.

Domain-Specific Languages are meant to be used in very particular conditions, since they are tools with limited focus. They are not meant to cover a whole project, instead a project may have many DSLs, each designed for a different area of the project. There are many advantages of using Domain-Specific Languages. [20]
One advantage of using a DSL is that it provides a means to more clearly communicate the intent of a part of a system. This improves development productivity, by facilitating both the identification of mistakes or defects in the code as well as the correction of those mistakes.
Another advantage of DSLs is that they improve communication between software engineers and customers and users of that software. DSLs provide a clear yet concise language to deal with domains, hence becoming understandable by the domain experts. By being able to understand the DSL code, domain experts can detect defects, and communicate them to the programmers in a simpler way.
Lastly, another key motivation for using Domain-Specific Languages is the ability to shift behaviour from compile time to runtime or to a more suitable execution environment. By expressing rules in a DSL, users can describe logic in a readable and domain-oriented way while enabling that logic to be executed efficiently elsewhere, such as within a database. This approach allows complex behaviour to be specified by non-technical stakeholders and translated into executable code for the appropriate platform.
Along with these advantages, according to Fowler [20], there are also some problems that some consider when using DSLs.
Firstly, it may not be justifiable to apply a DSL to a project if it will increase the complexity and the clarity of the project. Some may argue that bringing too many DSLs to a project may make it harder to understand what's going on. Fowler [20] somewhat refutes this by affirming that projects and software will always have more complex areas to learn, and that the underlying model behind the DSL may be considered even more costly to learn than the DSL itself.
One can also consider the cost of building and maintaining a DSL to be a problem or disadvantage, as well as the costs of learning how to build it. Sometimes there is no need to add a DSL. And even when there is, if there is not enough knowledge around building a correct DSL, one might end up building something costly and that will become a waste. The costs of building a DSL should be weighed in contrast with the costs of managing the underlying model, as well as the costs of learning how to build a correct DSL.
Finally, the last concern to be mentioned around DSLs is about the abstraction created when building and using a DSL. DSLs help express complex domain behaviour clearly, but they can encourage rigidity in thinking. The abstraction should always be considered evolving, and changes should be welcomed rather than resisted.

2.3 Model-Driven Engineering

In the computer science context, a model is a powerful tool whose primary goal is to ease communication between the several stakeholders involved in system development, and to facilitate the interpretation of the system visually, since visual notation is more comfortably assimilated than pure code [41]. The idea behind a model is to represent some feature of the system to be developed - with that representation being an abstraction of a specific aspect -, without having to worry about the representation of the rest of the system. In its essence, models allow us to represent parts of the system and manipulate them independently from other parts. Many advantages come from the use of models to structure the system, such as improving communication between stakeholders, being able to visualise the finished product without having a full development, and allowing to specify in detail many features of the construction phase [30].
Regarding software construction, be it by explicitly creating artefacts or by implicitly mapping the models mentally, the programming in any language will go through a modelling process.
Models facilitate effective communication among various stakeholders involved in software development. Using graphical representations, such as diagrams, helps clients grasp concepts more quickly than reading code directly. Improved communication leads to better understanding, realistic expectations, and superior outcomes. Models allow us to visualise the final product without requiring complete construction. By examining the model, design flaws can be identified early on, which is more cost-effective to address in advance rather than during or after construction. Models also serve as precise specifications of the work to be done. They provide an accurate road map for project managers to estimate, schedule, and plan the construction phase, enabling effective project management [30].
Model-Driven Engineering (MDE) and Model-Driven Development (MDD) are generic terms which define the approach to software construction where the main artefacts are models and where the systems are described as models which conform to meta-models [30].
Modelling emerging thoughts and ideas for a project is a good way to facilitate software development by contributing to software re-usability and maintainability, decreasing the cost of construction, and increasing developer productivity. It aids in the early detection of design flaws or misalignments between the stakeholders and developers and can facilitate the integration of a large project involving many different parties [30]. These are the main advantages of using MDD.
The models involved and included in a model-driven approach must also be defined, and that is where meta-models come in. Another benefit of using MDD is that it takes advantage of the meta-model definition to apply as many structural restrictions as possible, which ensures maximum potential for the model and minimal necessities for manual adjustments. Nevertheless, Model-Driven Development must be applied keenly. MDD can

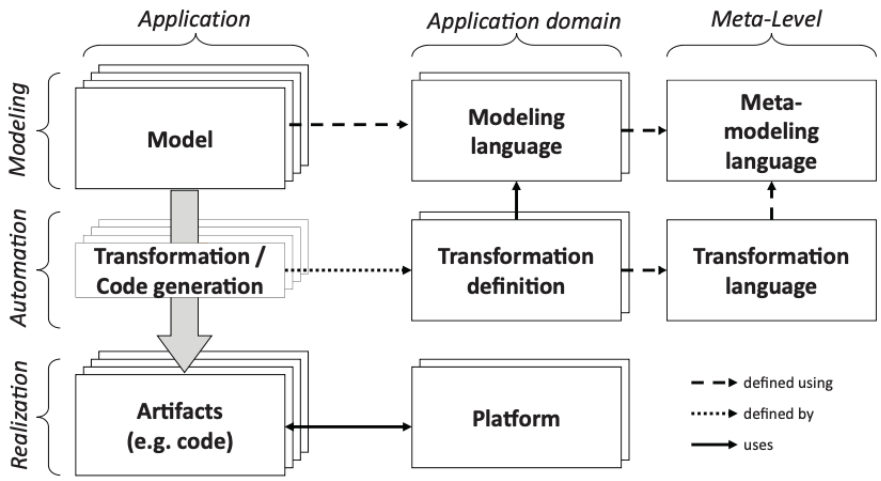


Figure 2.4: Representation of MDE methodology [6]

be considered very complex and should be dealt with wisely. With this in mind, usually, it does not pay off to apply MDD in a small project [36]. When applied to larger systems, MDD has all the advantages mentioned above and can ease many software development processes, such as needed modifications arising during or after construction.

Figure 2.4 gives an overview of the main considerations in MDE and how different aspects are addressed. Model-Driven Engineering tackles issues from two dimensions: conceptualisation (columns) and implementation (rows).

The implementation aspect focuses on mapping models to existing or future running systems. It involves three core aspects [6]:

- **The modelling level**: defining the models themselves.
- **The realisation level**: implementing solutions through artefacts, such as code in software systems.
- **The automation level**: establishing mappings between the modelling and realisation levels.

The conceptualisation aspect revolves around defining conceptual models that describe the real world. This can be applied at three main levels [6]:

- **The application level**: defining application models, performing transformation rules, and generating actual running components.
- **The application domain level**: defining modelling languages, transformations, and implementation platforms specific to a particular domain.
- **The meta-level**: defining models and transformations for conceptualisation itself.

Model-Driven Architecture (MDA) is an approach to MDE proposed by Object Management Group (OMG), which provides a framework for software development and uses

models to describe the system to be built [34]. MDA's idea is to separate a system's crucial models and offer a consistent structure to these models.
At a high-level of abstraction, the computation-independent model (CIM) defines the system's environment and requirements without specifying any details regarding its computation, using language that is easily understood by professionals within the system's domain.
Next comes the platform-independent model (PIM), which is meant to provide precise descriptions of the system's structure and functionality while keeping technical details hidden or abstracted. It is accompanied by the platform-independent component view, which is in charge of describing computational components and how they interact with each other, without being tied to any specific platform or technology. [21]
At the lowest layer of this architecture is present the platform-specific model (PSM), which is derived from the PIM and is in charge of defining important implementation details of a system in a given platform [30].
Using Model-Driven Development as the preferred development approach comes with benefits, one of them being the ease to deal with the implementation step for multiple platforms, the PIM-to-PSM transformation.
Besides MDA, there are other approaches, such as Domain-Specific Modelling (DSM), that can be considered model-driven.
Domain-Specific Modelling (DSM) involves using a specialised language tailored to a specific domain to represent systems. DSM often includes the ability to generate code for corresponding software systems automatically. A key concept in MDE is the use of Domain-Specific Languages (DSLs), which allow models to be expressed in terms familiar to a particular domain, improving clarity, reducing errors, and enabling automatic transformations into executable code [6]. To create and use DSLs effectively, Model-Driven Engineering relies on modelling workbenches. A modelling workbench is an integrated environment that allows users to define the syntax and semantics of a DSL, edit models, validate them, and generate code or other artifacts. It provides the infrastructure to ensure that models are both precise and executable, bridging the gap between domain concepts and software implementation [6].
The creation of a Domain-Specific Language (DSL) can be described as a Software Language Engineering cycle, which according to [6], can be dissected into seven steps:

1. **Domain Analysis**: With the objective of understanding the problem to be modelled, this first step goes through identifying the domain and extracting its entities, relationships and rules, while analysing the concerns and possible use cases of the target end-users of the Domain-Specific Language. The outcome of this step should be to have a domain model defined which formally describes the concepts of the domain.

2. **Abstract Syntax Definition**: After defining the domain model, the goal of this next step is to develop the modelling language's formal structure, by using a metamodelling framework and defining meta-classes and their relationships which will

represent the elements in the language. What results of this is a meta-model which specifies the abstract syntax of the language.

3. **Concrete Syntax Specification**: With the objective of making the language usable, this step seeks to specify textual or visual syntax and to map out the elements of the meta-model with their concrete representation, resulting in a textual/visual notation which will allow the end-user to create models.

4. **Semantic Definition**: During this phase the goal is to give a formal meaning to the models, by specifying how the model executes, and defining denotational or even translational semantics, with the outcome of having semantic rules which allow for interpretation, validation and simulation of the models.

5. **Tool Support**: This step involves the implementation of graphical/textual editors, support for automatic validation of restrictions (using Object Constraint Language (OCL) for example), and integration with model transformation engines, in order to have an environment to support the creation and analysis of models. The results of this step would be the practical modelling tools, such as modelling IDEs or web editors.

6. **Model Transformations and Code Generation**: To connect the language to other software artifacts, this step involves the definitions of Model-to-Model (M2M) and Model-to-Text (M2T) transformations, as well as the implementation of model-driven engineering pipelines. This results in code generation from the models, transformations, and automatic validations of the models.

7. **Validation and Evolution**: To guarantee the quality and evolution of the language, one needs to apply it to real-world scenarios, compare it with existing languages, and retrieve feedback from end-users and adapt if necessary. The outcome of this last step revolves around evolutionary versions of the language, each time more adequate and mature.

This process is generally aligned with frameworks like the four-layer meta-modelling architecture (M0-M3) of OMG. The four-layer meta-modelling architecture consists in having real-world examples at level M0; their modelled representation at level M1; at level M2, the model of the model in M1; and finally M3, features the meta-meta-model that defines the concepts used at M2.

2.4 Sirius

Sirius is an Eclipse-based open-source technology which allows the developer to create a graphical modelling workbench by leveraging and encapsulating Eclipse's modelling technologies, such as EMF and GMF while using little to no code. Sirius is a platform developed by Obeo, a French independent software provider, and it is meant to provide

a generic workbench for model-based architecture which can be tailored to the user's requirements. [42]
EMF's tree-based structure could make it inefficient for large models, and GMF's high level of complexity requires considerable knowledge in object-oriented programming. To address these issues, Sirius was developed. Sirius is built on top of GMF and allows you to quickly build a graphical editor without understanding the underlying processes. [39]
Sirius is made out of two separate parts:

- Sirius tooling refers to the side of Sirius meant for developers to build their graphical editors and modelling tools.
- Sirius runtime provides the end-users with the modeller definitions produced by the developers.

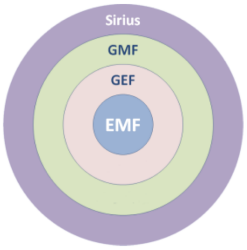


Figure 2.5: Sirius [47]

The result of Sirius tooling is a set of description models which can be deployed as an Eclipse plugin, and end-users do not need the Sirius tooling to use Sirius-based modellers. [3]
Sirius also has an advantage since it allows developers to integrate the graphical editor in their own applications. [19]

3

Related Work

3.1 Research Objectives

Within this chapter, the work related to this dissertation is presented.
The idea behind this dissertation is to perform an experiment with human participants between some of the state-of-the-art Entity-Relationship diagrams' modelling tools, in terms of functionality and usability. For that, a Systematic Literature Review (SLR) was conducted in order to understand what tools exist in the market, and what is available to use with minimal complications, and with interesting features and functionalities.

3.2 Systematic Literature Review

A Systematic Literature Review, commonly known as a Systematic Review, is a structured approach to identifying, analysing, and interpreting all available secondary research related to a specific research question, topic, or area of interest.
The following Systematic Review was elaborated according to Kitchenham's guidelines in [28].
This Systematic Review arose from the need to cluster and summarise all available and sparse information regarding what there is currently in the market in terms of tools to develop and draw Entity-Relationship diagrams. A factor that was considered in this research is if the tools follow a Model-Driven Development, since, as stated previously, this approach could be very beneficial in terms of confirming there is a language defined for the development of Entity-Relationship Diagrams, and that the tool is not just a drawing tool for any kind of diagram.
In the first stage of this review, all criteria were established. Objectives were structured, PICOC was defined - which refers to the Population, Intervention, Comparison, Outcome, Context -, the keywords and research questions were designated, and the selection criteria were specified. Since this literature review aims to understand what model-driven ER modelling tools exist, the planning process revolved around correctly defining the research questions and stating what to consider when assessing the quality of the findings.

3.2.1 Review Questions

The main question in this dissertation context is *"What are the current offers on Entity-Relationship modelling tools and are they complete?"*, which led to the execution of the Systematic Literature Review. This research question can be sub-sectioned into two smaller questions, *"What are the offers on ER Modelling tools?"* and *"What features are most valued in an ER modelling tool?"*.
Since the idea of this dissertation is to perform an experiment with human participants regarding ERD modelling tools, thoroughly understanding what currently exists in educational and professional environments is crucial. The following PICOC criteria structured the questions:

- **Population**: software engineers, computer science students
- **Intervention**: ER modelling tools with valuable features
- **Comparison**: different ER modelling tools
- **Outcome**: improving and easing the process of modelling ER models and building a database
- **Context**: educational, software engineering

To understand if the studies selected had information which could be considered relevant for the review, six questions were specified:

1. Does it mention ER modelling tools?
2. Does it mention features from ER modelling tools?
3. Does it mention usability on ER modelling tools?
4. Does it include information on model-driven approaches?
5. Does it include information on ER modelling?
6. Can the findings contribute to the knowledge of this review?

With these questions in mind, it was possible to filter the studies selected, obtaining the necessary information to elaborate the review.

3.2.2 Review Methods

Data Sources and Search Strategy

The SLR included a research over several digital libraries, such as IEEEXplore, ACM Digital Library, Google Scholar, ResearchGate and ScienceDirect. After retrieving the studies, the quality assessment checklist was applied to filter the studies and exclude the least relevant ones.
During the search strategy phase, a search string is defined to obtain the best results when searching for relevant documentation.
To find articles which mentioned ER modelling and modelling tools, the search string based on the research question and sub-questions consisted of combining the strings "*Entity-Relationship*", "*ER*", or "*ERD*" with the strings "*model*", "*modelling tool*", "*model editor*", or "*meta model*". With this in mind, the resulting search string was the following: *("ER"* ***OR*** *"ERD"* ***OR*** *"Entity-Relationship")* ***AND*** *("model"* ***OR*** *"meta model"* ***OR*** *"model editor"* ***OR*** *"modelling tool"* ***OR*** *"tool")*.

Study Selection

The study selection step has the objective to identify and filter the studies found, based on pre-defined criteria. These criteria are called the inclusion and exclusion criteria. In the case of this Systematic Literature Review, the following inclusion and exclusion criteria were defined:

- **Inclusion Criteria**: studies including information on ER modelling tools, as well as studies comparing different tools from different functionality and usability perspectives; studies in English or translated to English; more recent studies, between 2020 and 2024;

- **Exclusion Criteria**: studies which do not have any mention of ER modelling tools, or when mentioning ER modelling tools, not developing from the point of view of usability or functionality; duplicate studies, as well as papers that were not in English nor were translated.

Quality Assessment

The quality assessment step in a Systematic Review acts as an additional validation to the studies which are being chosen. As the name indicates, it is meant to assess the quality of the studies, after the inclusion and exclusion criteria are applied.
For this review in particular, the goal of the quality assessment is to verify that any studies on comparison of tools don't contain a biased opinion, meaning that the studies are concrete and thoroughly validated. Studies without threats to validity transmit less confidence than studies or papers where the threats to validity were studied and evaluated.

Data Extraction

The data extraction strategy was based on a few studies, which included ER modelling tools, some with Model-Driven Development (MDD).
Even though there are tens if not hundreds of ER modelling tools, the documentation stating whether they are model-driven or not is lacking, so there can be no conclusion for some of the ER modelling tools' development.

Data Synthesis

This review went through a qualitative data synthesis, where there was a need to translate the different studies found to be able to analyse them correctly.
At first, the goal of this Systematic Literature Review was to find and evaluate only Model-Driven Development (MDD) approaches since, as mentioned previously, the MDD brings many advantages to this type of development. However, the model-driven tools found were too few, and so the scope was broadened to also include non MDD tools.
The research has in consideration Chen's notation for the evaluation of the Entity-Relationship diagram modellers, since various different tools claim to have ER diagram modelling when in reality what they offer is a modelling of UML's Class diagrams. These two concepts tend to be mistaken with one another since they are both used to represent and define database design. This confusion is most impactful to undergraduates of database courses, since these concepts may overlap and become confusing.
This review also highlights tools which are free of charge since the main objective is to find tools that are easy to use for students and undergraduates of database related courses.
There were around 10 studies included in this SLR for the review of the different tools considered. Studies were excluded by reading the abstract and concluding they would have no value for the SLR.

3.2.3 Results

This section presents the results of the research conducted on Entity-Relationship (ER) Diagram Editors. The findings are categorised into two groups: editors known to have been developed using Model-Driven Development (MDD) and those for which MDD involvement is not evident or known.
The primary objective of this research, as outlined earlier, is to identify and analyse tools that are particularly beneficial for students and undergraduates enrolled in database courses, as well as for software engineers and developers. These tools are evaluated based on their educational value and overall effectiveness in simplifying the learning process. Additionally, the research seeks to identify tools that can streamline the teaching process for professors, offering features that enhance efficiency and improve the overall teaching experience.
Given this context, the review also placed significant emphasis on tools that respect the

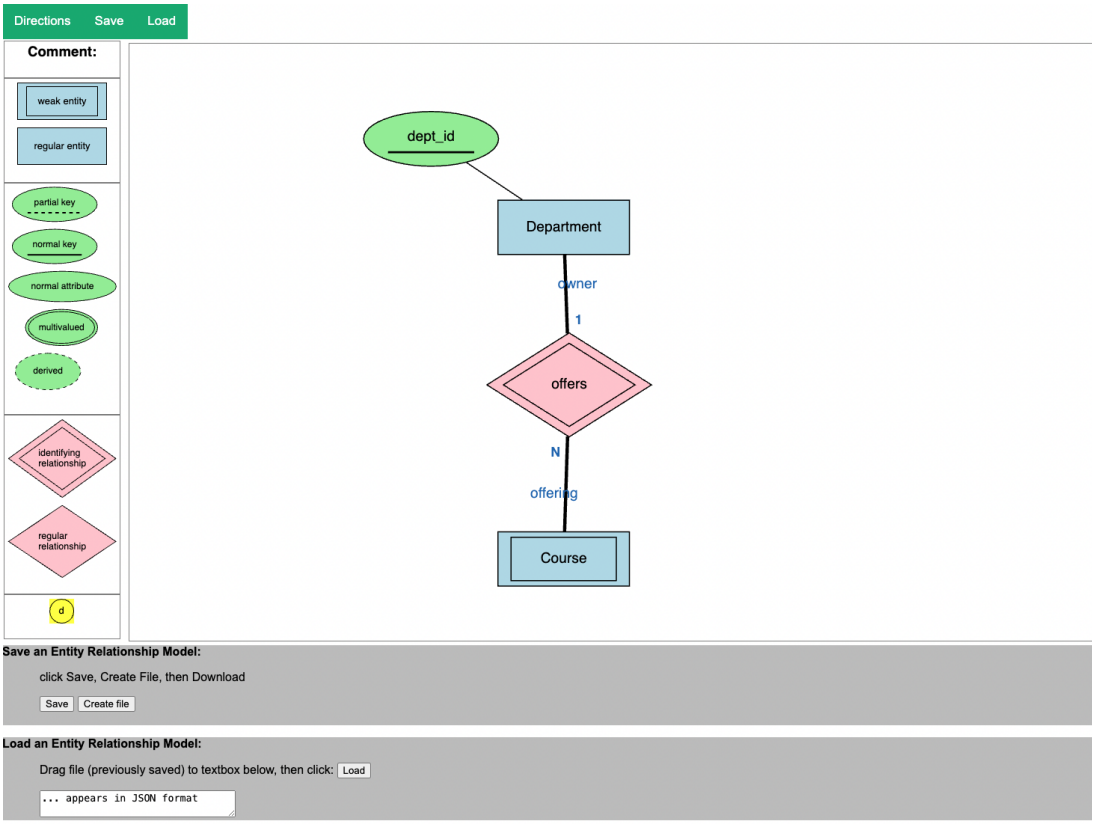


Figure 3.1: Layout of ERD Tool

original Chen's notation for Entity-Relationship (ER) diagrams. As Chen's notation is widely regarded as a foundational approach in database modelling, tools adhering to this standard were given special consideration. Their adherence was contrasted with tools that diverge from Chen's notation, as the latter may introduce inconsistencies or confusion, particularly for students new to database concepts. Highlighting this distinction ensures that the findings are aligned with the pedagogical goals of providing clarity and consistency in database education.
Regarding the usability of the tools identified and analysed, the studies evaluating various ER modelling tools provided little to no discussion on user-friendliness. This is an important aspect to consider, as a tool may offer numerous features, but if it lacks intuitiveness or a good user experience, those features can become significantly less valuable.

3.2.3.1 Model-driven Development Tools

As a first approach to this Systematic Review, only ER modelling editors with a model-driven development were considered. After extensive research, only two tools were found that covered the requisites and deserve to be mentioned. These are the ERD Tool and the BigER Tool.

ERD Tool The first tool under discussion is the ERD Tool, developed by Ron McFadyen in 2015 [33]. Its interface layout is depicted in Figure 3.1. The ERD Tool was created to address the need for an intuitive and straightforward ER modelling tool tailored specifically for educational purposes. According to McFadyen, many existing ER modelling tools are not well suited for academic or educational use, due to their complex and non-intuitive interfaces. This tool was designed to overcome these limitations while maintaining robust support for all main elements and features of ER modelling. These include entities (both regular and weak), relationships (and their variants), as well as attributes categorised as key, derived, multi-valued or partial key. However, this ERD Tool does not include more complex components of the Entity-Relationship diagram, such as generalisations (ISAs) or aggregations.
In terms of functionality, the ERD Tool adopts Chen's Notation for Entity-Relationship Diagrams, which is not common to find since, as mentioned before, many of these tools prefer to adopt UML's class diagrams notation.
It also provides basic validation for the models created and includes functionality to transform modelled ERDs into JSON format, as well as to convert JSON text back into diagrams.
The tool's development leverages the Eclipse Modelling Framework (EMF) and the Graphical Modelling Framework (GMF), underscoring its basis in model-driven development. Originally, the ERD Tool was implemented as a stand-alone application for Windows. However, in recent years, it has been redeveloped and is now available as a web application, with its latest update in 2023, further enhancing its accessibility and usability.

BigER Tool The BigER Tool, developed in 2021 by Philipp-Lorenz Glaser and Dominik Bork [22], is a plug-in for Visual Studio Code[1], and uses a textual representation as its leading model for Entity-Relationship diagram modelling. This means that it consists of a textual editor for the specification of ER models, which is then interpreted and transformed into a Diagram View, with the desired Entity-Relationship diagram. The resulting ER diagram, from the textual representation, follows Chen's notation in part, by having entities as rectangles and relationships as diamonds. However, the entities' attributes are contained within the entities, represented as a list, and not outside the entities, connected to them, and represented as ellipses, as Chen's notation states.
Even though its main developing method for diagram modelling is the textual representation, the BigER tool has some diagram manipulation features directly in the diagram view, such as addition of components (entities, relationships and attributes), and deleting or renaming entities and relationships. It does not, however, allow to delete or rename attributes directly in the Diagram View. For that, one would have to use the textual representation.
Another very valuable feature of this tool is model validation and consequent SQL code

[1] https://code.visualstudio.com/

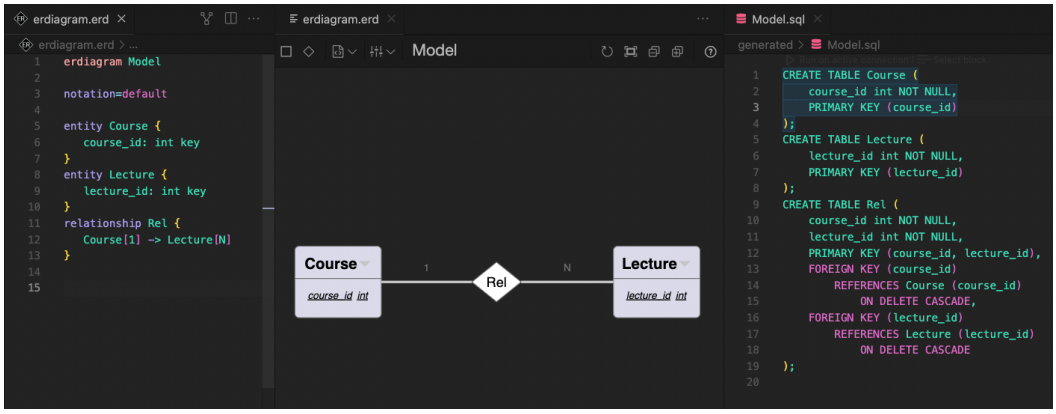


Figure 3.2: Layout of BigER Tool

generation.

In terms of model validation, since the creation of the diagram is textual, there are already many restrictions defined, such as only allowing connections between entities through the relationships for example. Another example is that, if one does not give a primary key to an entity, the system will warn the user that a primary key is missing. It also warns the user if the cardinality is not defined. These are just some examples of the type of validation that BigER offers.

This tool also has a variety of options to choose from when the user wishes to translate their database model (ER diagram) into functional SQL code, depending on the desired Database Management System (DBMS). It includes, for example, generic SQL, PostgreSQL, Oracle SQL, MySQL, and even Neo4j, which is not a relational database system but rather a graph database system.

As can be seen in Figure 3.2, the layout of this tool has on the left side the textual representation of the ER diagram, at the centre it offers the rendering of the ER diagram according to its textual representation, and on the right is the SQL code generation, for which the type of database system can be defined.

This tool follows a client-server architecture, with the server side providing the validation and code generation of the model, while the client side is responsible for the rendering of the generated diagram elements next to the textual editor.

As for the technologies utilised, this tool takes advantage of Eclipse XText and Sprotty framework for the syntax specification of the ER language and for the diagram generation through the textual model. For the code generation, Eclipse XTend was used, since it integrates well with XText. This type of development can be considered a model-driven development since a meta-model is defined in order to map the concepts to the semantic model. One needs to have both Visual Studio Code and at least Java JDK version 11 to be able to use this tool.

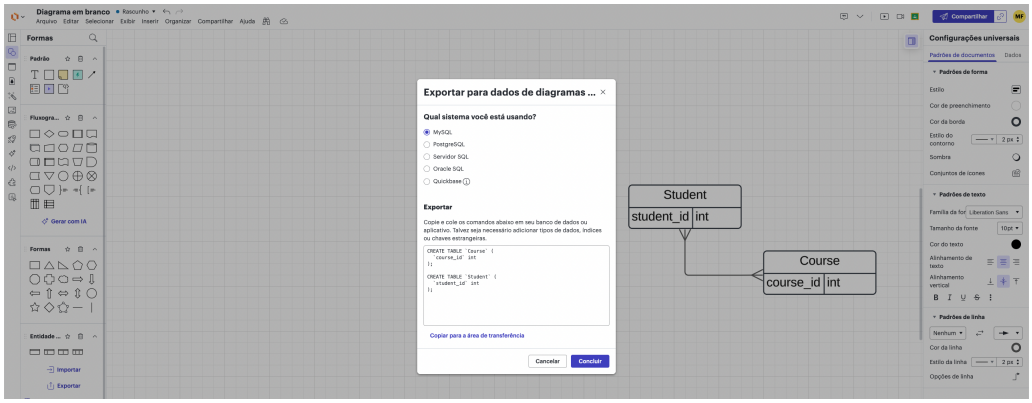


Figure 3.3: Layout of Lucidchart

3.2.3.2 Non Model-Driven Development Tools

Since the research on model-driven ER modelling tools resulted in poor quantity of results, and since it is still important to accurately point out the existing and usable ER modelling tools, the scope of the research must be broadened to consider the most used ER modelling tools, regardless of their development approach.
The tools considered for this section are mentioned in [45] and [31]. These two sources offer some concrete feedback on the state-of-the-art tools for ER modelling.

Lucidchart Lucidchart [13] is a common modelling tool people opt to use, since it allows modelling of several types of diagrams, including ERDs, and offers some interesting functionalities in its free-of-charge plan.
This tool, even though it has some presets for specific types of diagrams, allows to build any kind of diagram, since it gives some liberty to the user to drag and drop any kind of shape to the diagram, including diamonds and rectangles, which would then indicate it is possible to manually build an ER diagram. However, in order to take advantage of the feature of SQL code generation, the user must develop a UML Class diagram, and not an ER diagram.
As mentioned previously, Lucidchart allows the user to generate SQL code according to a Database Management System, from a formerly developed Class diagram, and also allows live collaboration. However, its SQL generation has some flaws, since, as can be seen in Figure 3.3, the diagram has a many-to-many relationship between *Student* and *Course*, which would mean an extra table is needed for the relationship, and none has been generated.
It has a free-of-charge plan that includes three private documents (diagrams) with a limit of 60 elements per document. Despite that, there is no documentation to confirm that it supports diagram validation.

ERDPlus The ERDPlus, which was developed in 2013 and has been maintained with their last update in 2023, created by Nenad Jukic et al. [25], is a tool mainly developed

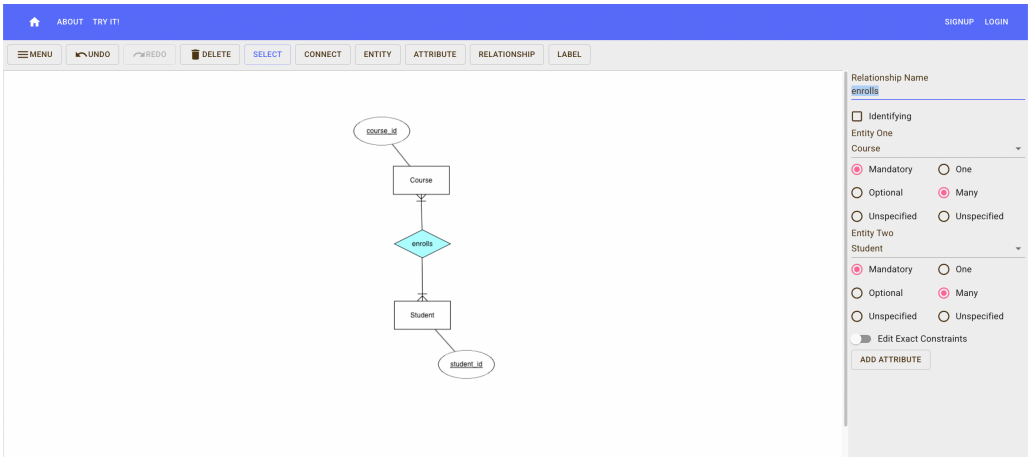


Figure 3.4: Layout of ERDPlus Tool

for educational purposes. Its purpose was to deliver a free of charge tool for students and undergraduates of computer science and other data modelling related courses to use, with minimal complication and direct access.
Its development started from two prior downloadable tools for Windows, which allowed to build ER diagrams and star schemas, back in 2005. The authors then decided to create something more accessible and flexible, so a cloud-based data modelling tool was developed, combining the functionalities from the two previous tools while also adding new features. ERDPlus was officially launched in the beginning of 2013.
In terms of graphical design, ERDPlus uses a mixture of Chen's notation with Crow's Foot notation, where the components themselves respect Chen's notation and the connections between the entities and relationships use Crow's Foot notation.
In terms of usability, this tool allows to create entities, relationships and attributes, and connect them with each other. However it does not support generalisations (ISAs) or aggregations, which are more complex features of the ER model. Also, the definition of the unique attribute of each entity is not entirely correct, since it underlines the attribute when defining it as unique, which according to Chen's definition, means the attribute is a primary key [8]. However there may be attributes defined in a model which are unique but are not meant to be primary keys.
This tool does not support the generation of SQL code from the ER diagram itself.
ERDPlus' layout can be seen in Figure 3.4.

DBDiagram.io DBDiagram.io [23] is a web modelling tool to help build database related diagrams which is code-based, since the diagram is generated according to the user's code. This diagram that is mentioned is not an ER diagram since it does not Chen's notation, but rather a UML Class diagram. This type of diagram creation and construction is not by diagramming but rather by developing a textual representation of the idealised structure of the diagram, and seeing the diagram developing alone from that same textual representation.

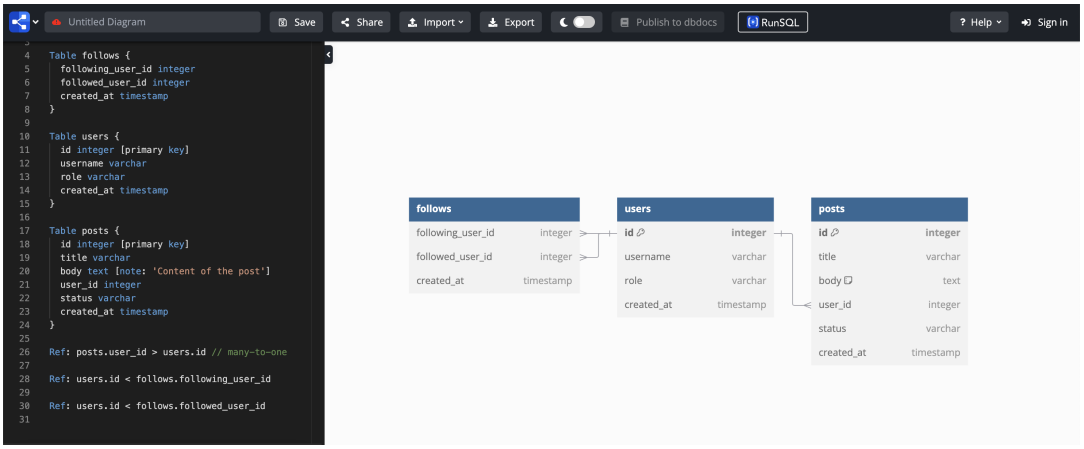


Figure 3.5: Layout of DBDiagram.io

DBDiagram.io allows the user to export a diagram into DBMS-specific SQL code (MySQL, SQL Server, PostgreSQL) and has a free of charge plan which includes this SQL code export and also import, lets the user build up to 10 diagrams, and also lets the user export the diagram as a PDF/PNG/SVG file. A login is needed to be able to benefit from these features.
Although this tool appears to have some functional requirements, the fact that the development of the ERD is code-based and not by diagramming may be considered a disadvantage since the user has to programmatically develop the diagram which can be seen as a redundancy if the goal is to obtain SQL code for example. However, it does guarantee a validation of the diagram since the diagram will not render if something in the textual representation is not correct. A layout of the tool can be seen in Figure 3.5.

Draw.io Draw.io [1] is a free-of-charge browser-based diagramming tool which allows the user to model ERDs, among many other types of diagrams. Its offer of modelling ER diagrams respecting Chen's Notation, is similar to the case of Lucidchart, where it has the shapes available for one to drag and drop.
It supports live collaboration [35] and has a stand-alone offline application.[14] However, no documentation confirms that it supports model validation nor allows exporting SQL code from the modelled diagram.
Rather than enforcing modelling rules, the tool enables users to freely draw ER diagrams with its standard shapes, which can make it appear more like a drawing tool than a dedicated ER modelling environment.

Visual Paradigm Visual Paradigm [2] is a business and IT transformation software solutions provider, which allows to model Entity-Relationship Diagrams with Chen's notation as well as relational models and UML Class diagrams, in the browser. It has a free-of-charge Community Edition desktop application, which also allows to build these models. It supports generation of scripts to build the database defined in the diagram,

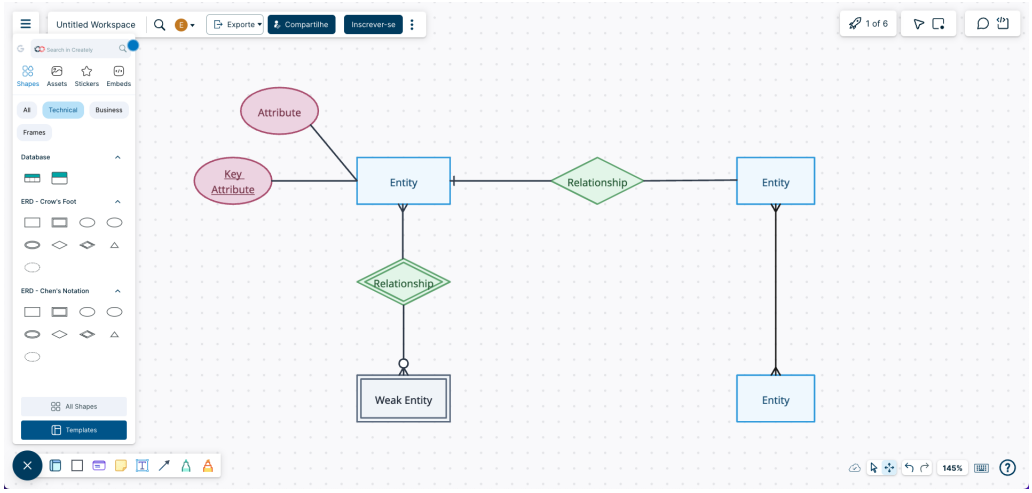


Figure 3.6: Layout of Creately

with the option to choose from different DBMSs. To generate the scripts without errors, Visual Paradigm validates the modelled relational model and checks for correctness, so in some way, it supports model validation. However, these features are not available in the Community Edition of Visual Paradigm.

Creately Creately [48] is a visual collaboration platform which allows modelling of ERDs, which can follow Chen's Notation since, just like other tools that have been analysed previously, it has a broad group of shapes to choose from. It supports live collaboration among several parties in its free-of-charge plan which also includes three private diagrams with a limit of 60 elements in total. However, it does not support the generation of SQL code from the diagram, and it also does not validate the models. It is simply a platform to design your diagrams while collaborating. Creately's layout can be seen in Figure 3.6.

3.2.4 Discussion

To analyse the results of this review, as to clarify what currently exists in terms of ER Modelling features and offers, a comparative table (Table 3.1) was structured.
Detailing the requisites defined previously, which are:

- Supporting Chen's Notation which is the one of the first notations taught in database courses,
- Allowing validation of the developed models in order to thoroughly understand the ER modelling language,
- Offering SQL code generation from the diagrams modelled to ease the process of database development,

- Giving the option of live collaboration for students to share projects and work together.

- Being free of charge or having a usable free of charge plan so that it can be used in the academic/educational context.

In general, and without much research, the user may have the impression that there are many options to choose from which offer most of these points. However, with a more thorough analysis, we can see that even though some of these Editors and Modellers offer some of the functionalities considered, there is no option to choose from there tools which includes all of the features mentioned above.
With the information retrieved from this SLR, various points can be concluded.
It can be observed that both the ERD Tool and ERDPlus, even though they support Chen's Notation and are free of charge, do not offer many features besides the modelling of the diagrams. The ERD Tool leaves much to be desired, since it being a model-driven tool, could become very powerful, in terms of model validation and SQL code generation, which are features the tool currently does not offer.
Besides that, BigER Tool and DBDiagram.io, which are very similar tools since both involve a textual development of the diagrams - in contrast with the other option which is modelling them visually - both have this disadvantage since the user ends up having to develop the text to represent the diagram they want to build. It may be considered a redundancy going through the development of the textual representation of the diagram, if the goal is to obtain SQL code. However, between these two tools, BigER Tool respects more of the Chen's Notation then the DBDiagram.io tool, and DBDiagram.io supports live collaboration if the user logs in with an account, while BigER Tool does not.
Lucidchart, Creately and Draw.io are very similar tools, in the sense that all three are very broad in terms of offers for different types of diagrams. In terms of ER diagrams, the three tools offer Chen's Notation if the user builds the diagram with the basic shapes and with no further logic or structure behind it. None of these three tools offer model validation, and in terms of SQL code generation, neither Draw.io nor Creately offer that feature, while Lucidchart does offer a very basic and flawed feature of SQL code generation only from the modelling of UML Class diagrams, as stated previously.
Visual Paradigm is a promising tool. However, it is not a concrete ER modelling tool, since the model validation and SQL code generation from the models are only possible when modelling a UML Class diagram, and not an Entity-Relationship Diagram. Not having live collaboration also delimits the tool.
In summary, none of the tools discussed fully cover all the functionalities that might be expected from a comprehensive Entity-Relationship modelling solution. Although a wide range of tools is available in the market, each tends to emphasize certain strengths while lacking in others. As such, the eight tools analysed can be seen as offering partial support, but not a complete set of desirable features.

	Chen's Notation	Model validation	SQL code generation	Live collaboration	Free of charge
ERD Tool	X	-	-	-	X
BigER Tool	X	X	X	-	X
Lucidchart	-	-	X	X	X
ERDPlus	X	-	-	-	X
DBDiagram.io	-	X	X	X	X
Draw.io	X	-	-	X	X
Visual Paradigm	X	X	X	-	-
Creately	X	-	-	X	X

Table 3.1: Comparative table of the discussed ER Modelling tools

3.3 Choice of Tools for the Experiment

The choice of tools to perform the experiment was based on the results from Section 3.2 (listed in Table 3.1) where a Systematic Review was conducted to understand what is the state-of-the-art in terms of ER modelling tools.
The first constraint applied to decide on the three tools to be evaluated was the pricing of the tool, and with that Visual Paradigm stopped being considered.
The second constraint to aid in this choosing, was that the tools must respect Chen's Notation. With that, DBDiagram.io and Lucidchart were excluded.
The ERD Tool and ERDPlus are very similar tools, as mentioned previously, but from the viewpoint of accessibility, the ERD Tool is not as friendly as ERDPlus, and because of that it was discarded for this experiment.
BigER, it being a model-driven tool, respecting Chen's Notation and being free of charge with SQL generation, is an interesting choice to bring to the Experiment.
Regarding Creately and Draw.io, which, as mentioned previously, are very similar tools. As can be seen from Figure 3.7, the trends from Google[2] show that Draw.io (also known as *diagrams.net*) has been generating a higher search interest than Creately, showing that it is more familiar and well-know. And so, for that reason, for the experiment the participants would be more familiarised with Draw.io.
With those exclusions, the three tools were chosen: BigER Tool, ERDPlus, and Draw.io.

[2] https://trends.google.com/

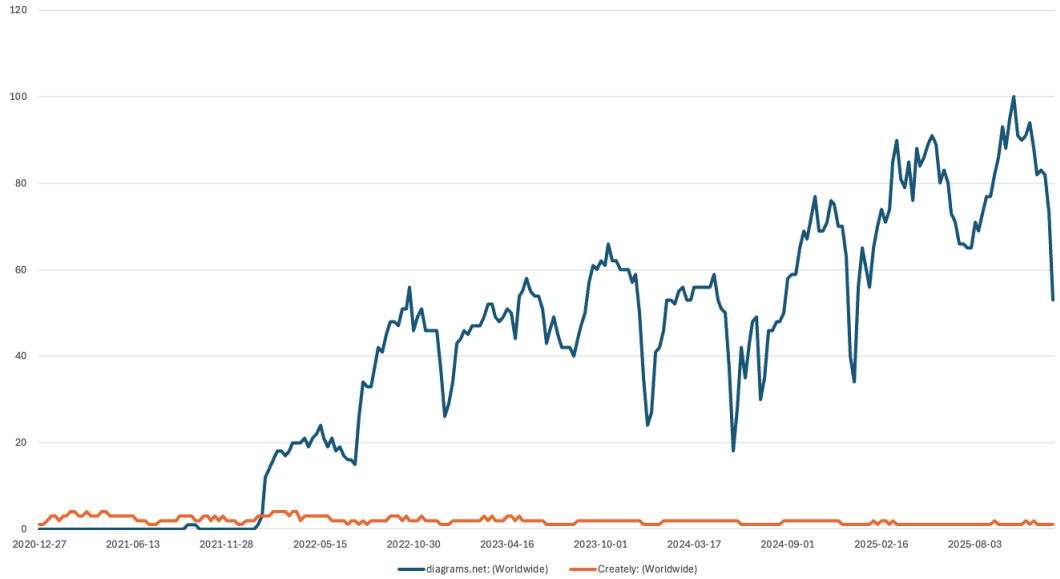


Figure 3.7: Search interest from Creately vs Draw.io

4

Experiment

As mentioned in Section 1.3, this dissertation will be conducting an Experiment. An experiment in scientific research involves manipulating variables and observing their effects on other variables. It requires the researcher to control the process, scheduling treatments and measurements for optimal statistical efficiency. Experimental research typically involves randomly assigning participants to different treatments, followed by observations to assess the effects. [29]

4.1 Planning

4.1.1 Goals

The goals of this experiment will be described according to the Goal-Question-Metric approach [4]. The primary objective of this experiment is to evaluate the effectiveness, user experience, and time-related effort associated with three ER modelling tools, in order to assess their suitability for academic and professional use. The evaluation considers the potential impact on students and software developers, from the perspective of learning and development, within the context of a controlled experiment involving participants from both groups.
To make this main goal more measurable, it is divided into three specific sub-goals, each of which is addressed in the course of this experiment:

- Goal 1: Analyse the three tools considered for this experiment, for the purpose of evaluation, with respect to their expressiveness and effectiveness in supporting Entity-Relationship modelling, from the viewpoint of learning and developing, in the context of an experiment conducted between students and software developers.
- Goal 2: Analyse the three tools considered for this experiment, for the purpose of evaluation, with respect to their usability in terms of user experience, ease of navigation, and satisfaction, from the viewpoint of learning and developing, in the context of an experiment conducted between students and software developers.

- Goal 3: Analyse the three tools considered for this experiment, for the purpose of evaluation, with respect to their efficiency, specifically time and effort required for Entity-Relationship modelling tasks, from the viewpoint of learning and developing, in the context of an experiment conducted between students and software developers.

For each of these goals, according to the Goal Question Metric approach, questions were defined.

- Question 1: How do the tools differ in terms of users' perception of their ability to effectively perform ER modelling tasks?
- Question 2: How do users rate each tool in terms of user experience, ease of use, and overall satisfaction?
- Question 3: Do the tools differ in the time and effort users spend to complete ER modelling tasks?

In order to answer these questions, three metrics were defined:

- Metric 1: Users' effectiveness score using the System Usability Scale (SUS)
- Metric 2: Users' user experience usability score using the System Usability Scale (SUS)
- Metric 3: Tasks' completion time

4.1.2 Participants

The experimental units in this experiment are the participants. These varied between Computer Science students who have had at least one database course and software developers, since the goal is to analyse the impact of different ER modelling tools in modelling databases, as described in Section 4.1.1. Some participants had Portuguese as their native language, and some did not, hence the written parts of experiment were done in English, a familiar language for all participants.
One of the tools chosen for this experiment, the BigER Tool, mandates that the participant has Visual Studio Code in their machine, in order to be able to install the tool (as an extension). With this in mind, the non-probability sampling methods applied are convenience sampling and snowballing. The participants were invited to join the experiment through the researcher's network, and encouraged to share the invitations to their connections.
Ten Computer Science students and ten software developers completed the experiment. From this group of participants, nineteen were male and one was female, and their ages ranged from 20 years old to 42 years old.

4.1.2.1 Grouping

The participants were separated into four different groups, which differ on the order in which the three tools are used, in order to minimise any possible biased results from using one tool before another. The four different groups are:

- Group A, using firstly Draw.io, then ERDPlus, and lastly BigER Tool,
- Group B, using firstly ERDPlus, then BigER Tool, and lastly Draw.io,
- Group C, using firstly BigER Tool, then Draw.io, and lastly ERDPlus,
- Group D, using firstly ERDPlus, then Draw.io, and lastly BigER Tool.

4.1.3 Experimental Material

This experiment was conducted online, through a video conference, using Google Meet. Participants were asked to join the meeting and share their screen while doing the several tasks for this experiment. To gather the feedback from each tool, the participants were asked to answer three surveys developed in Google Forms, one after using each tool. The surveys were in English, since some participants did not speak Portuguese.
The tasks, together with brief explanatory videos of each tool and some ER model definitions were delivered to the participants through a Google Drive folder that was shared with them. The task and the three videos were also in English.
Since one of the tools (BigER Tool) is a Visual Studio Code extension and requires a previous installation of Java JDK, participants were given the option to use Pop[1] - a collaboration tool in which two or more users can share one screen and work collaboratively - to use the researcher's screen and not have to install Visual Studio Code and/or JDK.

4.1.4 Tasks

The tasks for this experiment are the same for each tool. From what is in the scope of Entity-Relationship modelling, and according to what is meant to be studied through this experiment, four different tasks were created:

Task 1: Basic ER Model Creation: The participants are asked to create an ER model for a Library Management System, with the following entities and relationships:

- Entities: Book, Author, Library, Member
- Relationships:
  - Books are written by one or more Authors
  - Members can borrow Books

[1] https://pop.com/

  - A Library contains multiple Books

- Attributes:

  - For entity Book: ISBN
  - For entity Author: Full name
  - For entity Library: Library name
  - For entity Member: Identification number

Task 2: Adding Multiplicity and Participation Constraints: The participants are asked to modify the previous ER model to apply the following constraints:

- Each Book **must belong** to exactly one Library
- A Member can borrow **at most one** book at a time

Task 3: Advanced ER Features: The participants are asked to modify the previous ER model to have in consideration some more advanced features of the ER model such as weak entities and generalisation:

- A **Membership** is a weak entity that depends on a **Member**, and has a partial key attribute named **Membership type**
- A Book can be either a **Physical Book** or an **E-Book**

Task 4: Generation and Export Features: The participants are asked to try and export the resulting Entity-Relationship model into SQL code, if the tool supports the feature.

4.1.5 Hypotheses and Variables

The independent variables for this experiment are the three tools chosen to be analysed and evaluated. The choice for these tools was based on the results from Section 3.2 (listed in Table 3.1).
As explained in Section 3.3, the three tools were chosen: **BigER Tool**, **ERDPlus**, and **Draw.io** are the independent variables of this experiment.

The dependent variables are:

- user experience usability of the tools;
- effectiveness/expressiveness of the tools;
- time taken to complete the tasks, to measure the effort.

With the variables defined, null (H_0) and alternative hypotheses (H_1) are constructed:
$H_{0\,\text{User Experience}}$: The use of different tools does not influence the usability and user experience of modelling ER models.
$H_{1\,\text{User Experience}}$: The use of different tools influences the usability and user experience of modelling ER models.
$H_{0\,\text{Effectiveness}}$: The use of different tools does not influence the modelling of ER models in a correct and complete way.
$H_{1\,\text{Effectiveness}}$: The use of different tools influences the modelling of ER models in a correct and complete way.
$H_{0\,\text{Time}}$: The use of different tools does not influence the time spent modelling ER models.
$H_{1\,\text{Time}}$: The use of different tools influences the time spent modelling ER models.

4.1.6 Design

According to Y. Levi and T. Ellis in [29], there are three key categories of experimental design, with the first two - the true-experiment and the quasi-experiment - being the most well known.
The Lab experiment, or true-experiment, is an experimental design where the researcher exercises a high degree of control over the study conditions, particularly through the selection of participants and the random assignment of participants and/or events to two or more experimental groups.
The quasi-experiment, is an experimental design in which the researcher has limited control over the selection of participants. In such studies, participants cannot be randomly assigned to conditions, and the resulting sample may therefore lack the level of homogeneity typically desired in true-experiments.
Finally, there is the ex post facto design with a control group, which is particularly susceptible to threats to validity. Even so, this design is sometimes necessary in research contexts where obtaining pre-event or pre-experience measurements is not feasible, or where such measurements were not collected because the need for them could not be anticipated in advance.
For this dissertation, the design used is the Quasi-experiment, in which its key difference from a true experiment is that the researcher has limited leverage over the selection of the participants, which means not having the ability to truly randomly assign the participants and make sure that the sample is homogeneous. Since the attribution of the groups mentioned above was not purely random, this can not be a true experiment.
The participants were organised following a Within-Subjects design, meaning that each of the 20 participants tested the three tools. This design, in contrast with the Between-Subjects design, requires less participants since each participant can contribute with information for each of the independent variables.
In this case specifically, since it is a Within-Subjects design, the researcher could randomly assign participants to the order of the independent variables.

4.2 Execution

4.2.1 Preparation

The preparation of this experiment consisted on firstly introducing the participants to the tools they were going to be using in the interviews, with the purpose of reducing the bias due to lack of knowledge of each tool. The researcher recorded three videos, one for each of the tools, briefly explaining how each tool was used and what were its features.
The participants were also presented with a small summary of the Entity-Relationship modelling definitions, so that in case they hadn't worked with ER modelling in a while, they would have their memories refreshed.

4.2.2 Procedure

Each subject participated in the experiment through a private video conference (online) with the researcher, hence, twenty video conferences were conducted. Each participant was firstly asked to prepare for the experiment beforehand, by watching three brief explanatory videos of each tool, as well as read a small summary of the ER model definitions to refresh their memory, if needed.
Then, during each interview, each participant was firstly asked to share their screen with the researcher for the researcher to see each step taken by the participants, and to be able to time each task.
The participants were then asked to open each tool, with the order of the tools dependent of which group they were assigned to (section 4.1.2.1 Grouping). They had both the Tasks file (pdf) as well as the tool open, and performed the four tasks, while the researcher timed them. At the end of the four tasks for each tool, the participants were asked to fill out the survey correspondent to the tool just used. This procedure was repeated three times, one for each of the tools. Description of the procedure is in Figure 4.1.

4.3 Analysis

4.3.1 Dataset Preparation

As mentioned previously, all the participants filled out three surveys, one corresponding to each tool they tested. These surveys had the same format for each tool, and included the already mentioned System Usability Scale sections, containing ten questions for the effectiveness and ten questions for the user experience usability - each of the ten questions being answered with a score from 1 (strongly disagree) to 5 (strongly agree), as the System Usability Scale demands. Since the idea was to analyse the effectiveness/expressiveness and the user experience usability, the most common SUS was slightly adapted to fit into the desired evaluations. Both SUS questionnaires are presented in Tables 4.1 and 4.2. The raw results of these questionnaires are presented in Annex I and Annex II.
The time of each participant for each tool per task was timed during the videoconference,

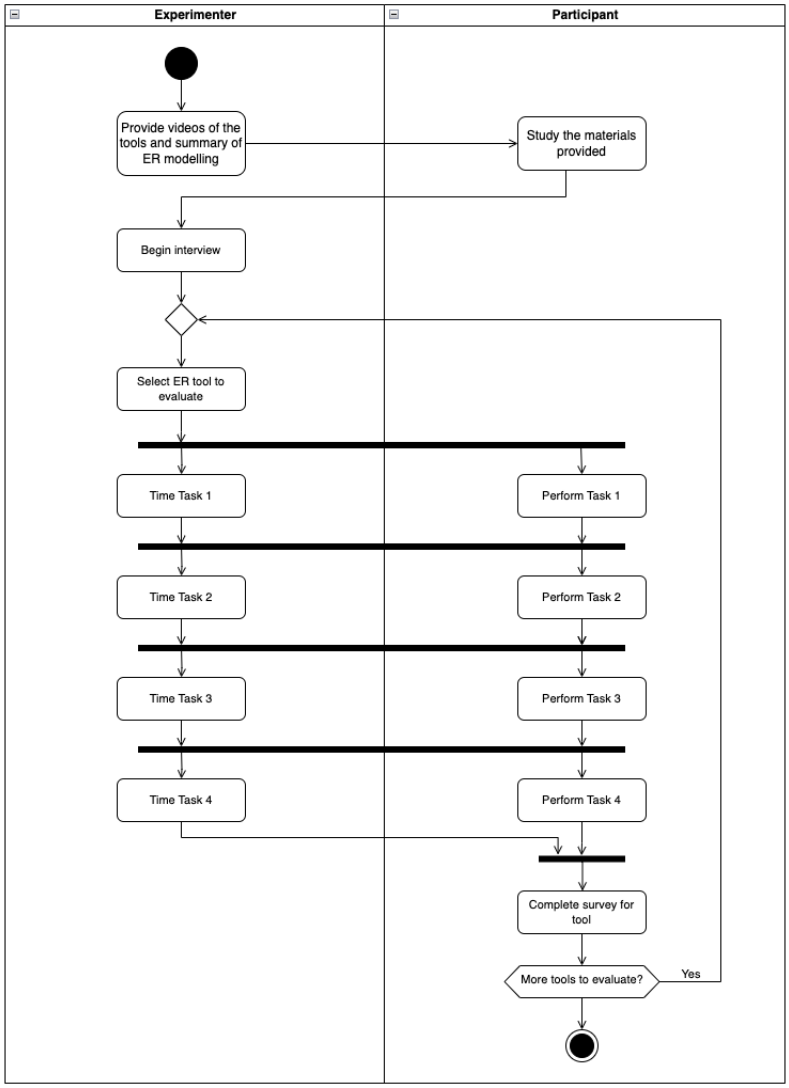


Figure 4.1: Activity Diagram describing interview process for each participant

while the participants solved the tasks. The times were then registered in an Excel file. The raw times for each task and tool are presented in Annex III.
All the responses from the surveys were downloaded as *.csv* files, and interpreted using Excel and the Real Statistics add-in, to obtain different kinds of results and plots.

4.3.2 Descriptive Statistics

Before conducting inferential statistical analyses, data visualizations were used to explore the distribution and variation of participant responses across the three tools.
As mentioned earlier, ten SUS statements were defined for the user experience and expressiveness variables. The System Usability Scale provides a score from 0 to 100, and as indicated, statements must alternate between positive and negative - odd statements are positive and even statements are negative -, and they are evaluated by the participants

Table 4.1: Effectiveness SUS Questions

Question
1. The tool allowed me to create all essential ER components (entities, relationships, attributes).
2. I found the tool lacking necessary functionality for ER modelling.
3. The tool made it easy to define cardinality and constraints.
4. I struggled to implement more advanced ER features (e.g., generalisation, weak entities).
5. The tool provided adequate options for exporting or generating SQL code.
6. The tool was inconsistent in how different functionalities were accessed.
7. I could easily modify my ER model after making changes.
8. The tool made performing simple modelling tasks unnecessarily complicated.
9. I felt confident that the tool supported all required ER modelling functionalities.
10. I had to spend too much time figuring out how to perform basic ER modelling tasks.

Table 4.2: User Experience SUS Questions

Question
1. I think that I would like to use this tool frequently.
2. I found the tool unnecessarily complex.
3. I thought the tool was easy to use.
4. I think I would need technical support to use this tool.
5. The tool's interface was visually clear and well-organised.
6. I found navigating between different features of the tool confusing.
7. I would imagine that most people would learn to use this tool very quickly.
8. I found the tool very complicated to use.
9. I felt very confident using the tool.
10. I needed to learn many things before I could get going with this tool.

on a scale from 1 (strongly disagree) to 5 (strongly agree). For the positive statements, the score is calculated by subtracting one from the participant's response. For the negative statements, the score is calculated by subtracting the participant's answer from five. These two calculations give scores for each statement ranging from 0 to 4. Then, the ten scores from each participant are summed up, ranging from 0 to 40, before being multiplied by 2.5 which gives a range of scores from 0 to 100. Finally, the means, standard deviations and interquartile ranges for each variable's SUS scores are calculated.

The data related to the time variable was firstly split into the three main tasks of the experiment (time per tool and per task), so that they could be analysed separately, but a sum of the times by task (time per tool) was also considered, to facilitate the analysis of the hypotheses for the time variable. This sum does not include Task 4, since this task was only solvable by one of the tools, the BigER tool.

The following tables and figures summarise the data collected and ease the interpretation:

- Table 4.3: A table with the means, standard deviations (SD), medians and interquartile ranges (IQR) of the Effectiveness SUS scores, for each tool;
- Table 4.4: A table with the means, standard deviations (SD), medians and interquartile ranges (IQR) of the User Experience SUS scores, for each tool;
- Figure 4.2: A box plot of the effectiveness measured, using the SUS, per tool;
- Figure 4.3: A box plot of the user experience usability measured, using the SUS, per tool;
- Figure 4.4: A box plot of the total time measured per tool;
- Table 4.5: A table with the means, standard deviations (SD), medians and interquartile ranges (IQR) for the time taken by each task, for each tool, as well as for the total time (sum of the three tasks);
- Figure 4.5: A box plot of the time (in seconds) of Task 1, for each tool;
- Figure 4.6: A box plot of the time (in seconds) of Task 2, for each tool;
- Figure 4.7: A box plot of the time (in seconds) of Task 3, for each tool;

Task 4 is not included as a box plot nor summed in the total times since, as mentioned previously, it could only be solved by one of the tools, the BigER tool, as it was the only tool supplying the required functionality (the generation of SQL code). Therefore, no comparison in completion time was possible across tools for this task.

Table 4.3: Mean, standard deviation (SD), median and interquartile range (IQR) of Effectiveness SUS Scores for each tool

	Effectiveness			
Tool	Mean	SD	Median	IQR
ERDPlus	71.50	15.03	76.25	17.5
Draw.io	65.50	15.66	66.25	16.25
BigER	88.50	9.61	90.00	15.63

Table 4.4: Mean, standard deviation (SD), median and interquartile range (IQR) of User Experience SUS Scores for each tool

	User Experience			
Tool	Mean	SD	Median	IQR
ERDPlus	80.50	19.66	87.50	18.13
Draw.io	73.88	16.27	76.25	18.13
BigER	73.50	16.75	75.00	28.13

Table 4.5: Mean, standard deviation, median and interquartile range for time variable for each task, as well as in total

Task	Tool	Mean	SD	Median	IQR
Total	ERDPlus	370.4	98.11	353	155.25
	Draw.io	413.9	107.68	377	92.5
	BigER	348	85.52	314.5	126.75
Task 1	ERDPlus	189.05	50.73	202.5	79
	Draw.io	225.3	59.42	205.5	34.75
	BigER	189.05	48.62	180.5	69.5
Task 2	ERDPlus	55	24.40	47.5	25.5
	Draw.io	43.9	21.13	40	21.25
	BigER	38.65	14.89	34	20
Task 3	ERDPlus	126,35	35.75	122	54
	Draw.io	144.7	34.76	135	32.5
	BigER	120.3	30.89	116.5	45

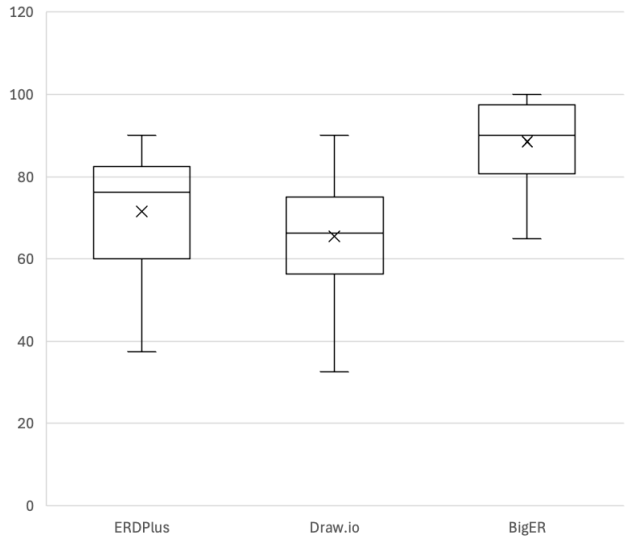


Figure 4.2: Effectiveness SUS scores box plot. Xs represent the mean

4.3.3 Hypotheses Testing

To properly interpret the data collected in this experiment, statistical testing was necessary, and the book [15] was used as guide for the steps that follow.
The right statistical tests to apply are:

- One-way Repeated Measures ANOVA test: This test is suitable for data that meets the assumptions of normality and sphericity. It is appropriate in this experiment because all participants were exposed to all conditions (i.e., each participant used all three tools), making it a repeated measures design. Additionally, it is considered "one-way" since there is only one within-subjects factor: the tool used.

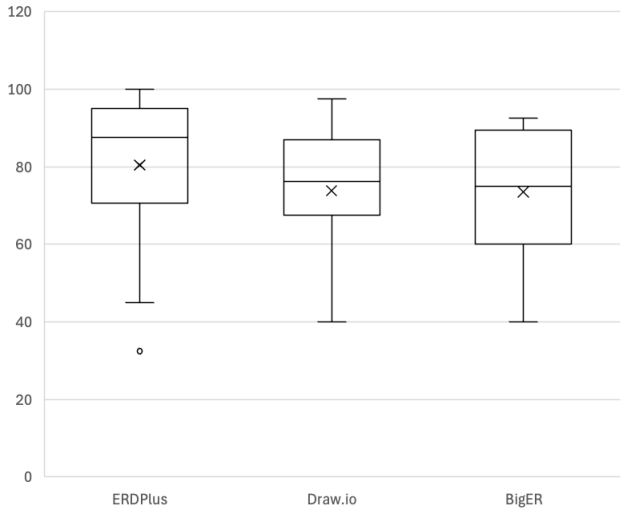


Figure 4.3: User Experience usability SUS scores box plot. Xs represent the mean

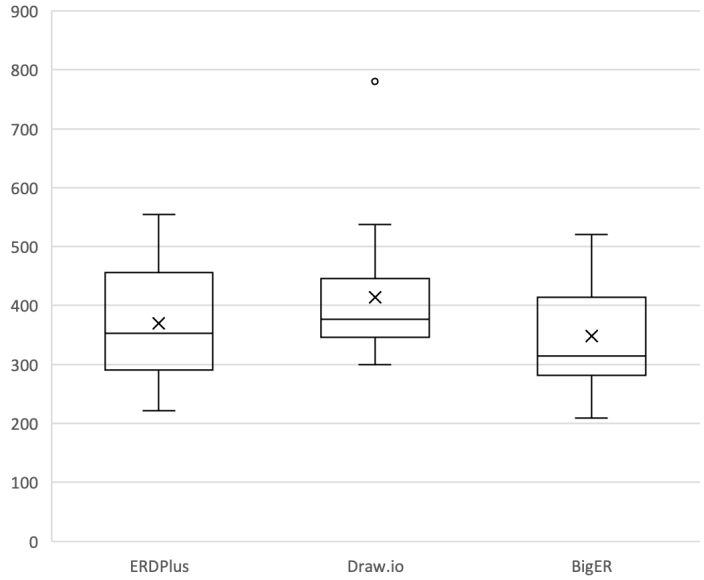


Figure 4.4: Time box plot. Xs represent the mean

- Friedman's test: As noted by Field in [15], this is the non-parametric alternative to the one-way repeated measures ANOVA. It should be used when the assumptions of normality or sphericity are violated.

The first step was to analyse the distribution of the data for the three dependent variables: effectiveness/expressiveness, user experience, and time. This was done to determine whether parametric tests (such as ANOVA) could possibly be conducted or if non-parametric tests (such as Friedman's test) would be more appropriate. To this end, the Shapiro-Wilk test was firstly conducted to evaluate the normality of each dataset.
As shown in Table 4.6, the distributions for both effectiveness and user experience usability scores are not normal for the ERDPlus tool, since both p-values are lower than 0.05. This violation of the normality assumption rules out the use of statistical test ANOVA for these

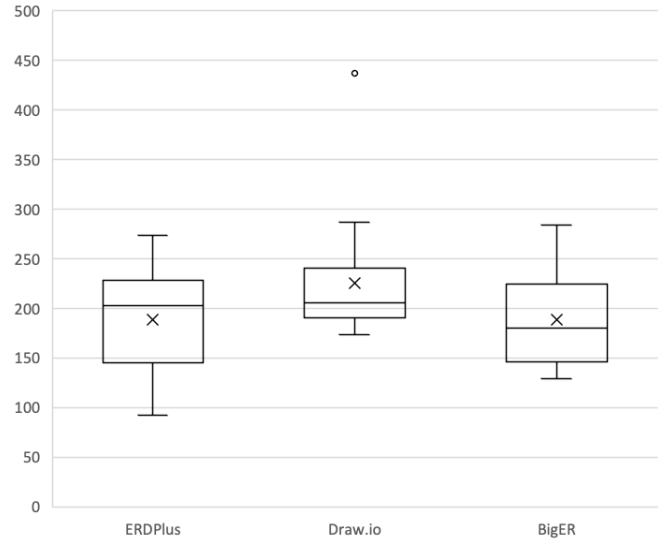


Figure 4.5: Task 1 time box plot. Xs represent the mean

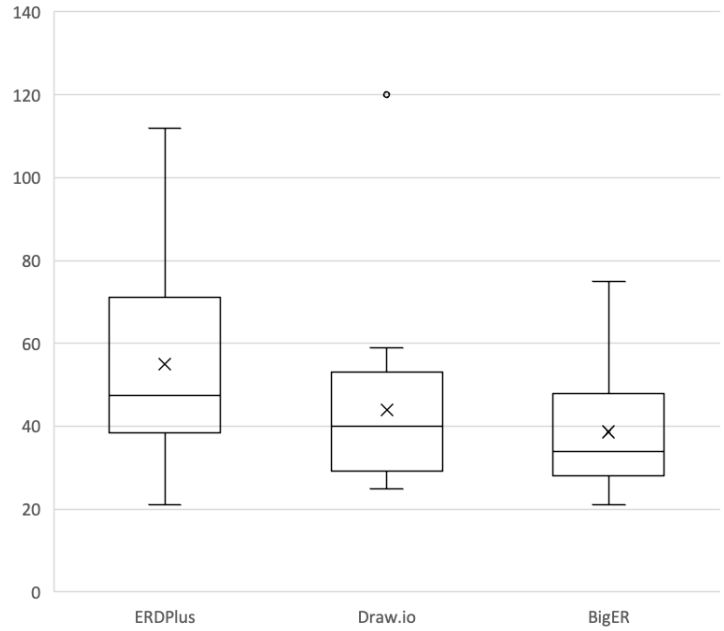


Figure 4.6: Task 2 time box plot. Xs represent the mean

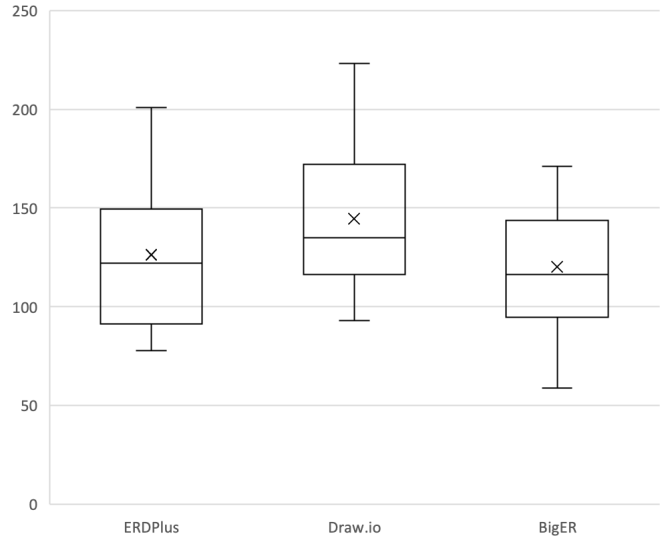


Figure 4.7: Task 3 time box plot. Xs represent the mean

variables. Instead, Friedman's test was selected to evaluate whether there are statistically significant differences across tools regarding these two variables.
Regarding the time variable, Table 4.6 shows that the **total** time does not meet the assumption for normality since Draw.io does not have a normal distribution of data. Task 3 meets the assumption of normality across all tools, while Task 1 and Task 2 do not, due to a non-normal distribution for the Draw.io tool for Task 1, and due to a non-normal distribution for the Draw.io and BigER tools for Task 2. With that, Friedman's test was selected to evaluate total time, as well as Tasks 1 and 2, for statistically significant difference.
Regarding Task 3, to run the ANOVA statistical test confidently, another assumption should be met, the sphericity, which refers to the equality of variances of the differences between all pairs of within-subjects conditions. In order to verify this assumption, Mauchly's test for Sphericity was conducted, using R and the *ez* library, and this test indicated that the assumption of sphericity had not been violated, with W(2) = 0.93 and p = 0.532 (> 0.05).

Table 4.6: Shapiro-Wilk test p-values for normality by variable, tool, and task (p = 0.05)

Variable	**Task**	**ERDPlus (p)**	**Draw.io (p)**	**BigER (p)**
time	Total	0,3061	**0,0003**	0,3546
	Task 1	0,5381	**3,27E-05**	0,1236
	Task 2	0,1002	**7,43E-05**	**0,0458**
	Task 3	0,3273	0,1044	0,7584
Effectiveness	-	**0.0133**	0.761	0.0837
User Experience	-	**0.0011**	0.255	0.0517

Following the assumptions' testing done, Friedman's and ANOVA tests were conducted. Friedman's test was conducted both for the effectiveness and user experience variables,

as well as for Task 1 and Task 2 time. One way repeated measures ANOVA test was conducted for Task 3 time.

Effectiveness

For the effectiveness variable, the results of Friedman's test gave a p-value of 1.41e−5, which being lower than 0.05, means that there is a statistically significant difference between the three tools tested, in terms of effectiveness.
Consequently, *post-hoc* tests were conducted, to identify this difference. The test used to compare the conditions in pairs was Wilcoxon's Signed-Rank test for Paired Samples, since the same participants were subjects to the different conditions. Wilcoxon's Signed-Rank test was chosen since it is the appropriate one to use when there are two sets of scores to be compared, and these scores come from the same participants, which is the case for this experiment.
The results of Wilcoxon's Signed-Rank test for Paired Samples are in table 4.7.
From these results it can be deduced that both the comparison between ERDPlus and

Table 4.7: Wilcoxon's Signed-Rank test results for Effectiveness

Comparison	Raw p-value	Bonferroni Correction	Holm's Correction
Draw.io vs BigER	3.82e−6	1.14e−5	1.14e−5
ERDPlus vs BigER	1.53e−5	4.58e−5	3.05e−5
ERDPlus vs Draw.io	0.13	0.39	0.13

BigER and the comparison between Draw.io vs BigER yielded a statistically significant difference, even after Holm's correction and Bonferroni correction, since the p-values are very low, meaning that BigER outperformed the other two tools in terms of effectiveness/expressiveness, with an average score of 88.50, while ERDPlus had an average score of 71.50 and Draw.io had an average score of 65.50.
This means that from the hypotheses stated in section 4.1.5, the non-null hypothesis regarding effectiveness under these conditions ($H_{1\,\text{Effectiveness}}$) cannot be rejected, and the use of different tools influences the modelling of ER models in a correct and complete way.

User Experience

For the user experience variable, the results of Friedman's test gave a p-value of 0.135, which being higher than 0.05, means that we failed to detect any statistically significant differences in user experience scores between the three tools. This means that from the hypotheses stated in section 4.1.5, the null hypothesis regarding user experience under these conditions ($H_{0\,\text{User Experience}}$) cannot be rejected, and the use of different tools does not influence the usability and user experience of modelling ER models.

Time

For the time variable, as mentioned previously, both individual (by task) and total time analyses were conducted.
In individual (by task) terms, for Task 1 and Task 2, Friedman's test was run.

Regarding Task 1, the resulting p-value from Friedman's test was at 0.06, which being higher than 0.05, means that there is no statistically significant difference between tools in terms of time for this task. Hence, no *post-hoc* tests were conducted.

For Task 2, the p-value was 0.01 which, being lower than 0.05, means that there is in fact a statistically significant difference between tools regarding this variable, and *post-hoc* comparisons should be conducted.
As justified before, for the post-hoc tests, Wilcoxon's Signed-Rank test was used to compare each pair of tools. The p-values resulting of these tests plus their corrections (Holm and Bonferroni) are stated in table 4.8. As it can be seen from the table, after the corrections, only the comparison between ERDPlus and BigER yielded a statistically significant difference.

Table 4.8: Wilcoxon's Signed-Rank test results for pairwise comparison for Task 2 time

Comparison	Raw p-value	Bonferroni Correction	Holm's Correction
ERDPlus vs BigER	0.001	0.003	0.003
ERDPlus vs Draw.io	0.04	0.13	0.088
Draw.io vs BigER	0.346	1	0.346

For Task 3, as the normality and sphericity assumptions were verified, the One-way repeated measures ANOVA test could be confidently conducted, and the resulting p-value was 0.033. Since it was lower than 0.05, this means that there is a statistically significant difference across tools regarding time for Task 3, so *post-hoc* pairwise comparisons should be conducted, to find this difference.
As the data was normally distributed and this is a repeated measures design, the *post-hoc* adequate test to compare between tools is the Paired Samples T-Test. The results of the test are in table 4.9. By analysing the results after Holm's correction, and even though ANOVA's results indicated that there was a statistically significant difference between tools, what can be concluded by these *post-hoc* tests is that there is no evidence of a difference across tools regarding time for Task 3.

As for the total time spent by each participant in the experiment, as mentioned previously, Friedman's test was applied. The resulting p-value of this test was not significant, with a value of 0.086, meaning there is no statistically significant difference in terms of time spent by participant on the experiment.

Table 4.9: Paired Samples T-Test results for pairwise comparison for Task 3 time

Comparison	Raw p-value	Holm's Correction
Draw.io vs BigER	0.02	0.06
ERDPlus vs Draw.io	0.08	0.17
ERDPlus vs BigER	0.45	0.45

This means that from the hypotheses stated in section 4.1.5, the null hypothesis regarding time ($H_{0\,\text{Time}}$) cannot be rejected, and the use of different tools does not influence the time spent modelling ER models.

4.4 Discussion

4.4.1 Evaluation of Results and Implications

Having the results from the experiment, it was then possible to answer the questions previously defined in Section 4.1.1.

- **Question 1.1: How do the tools differ in terms of users' perception of their ability to effectively perform ER modelling tasks?**
  By accepting the hypothesis that the use of different tools influences the modelling of ER models in a correct and complete way, and by analysing the statistics on the SUS scores for effectiveness (in Table 4.3), it can be concluded that in terms of effectiveness, the tools ERDPlus and Draw.io do not differ much between themselves, with mean scores of 71.50 and 65.50, respectively. However, BigER stands out positively and has the most significant difference, with a mean of 88.50 on its Effectiveness SUS scores.

- **Question 1.2: How do users rate each tool in terms of user experience, ease of use, and overall satisfaction?**
  By not rejecting the null hypothesis, which states that the use of different tools does not influence the usability and user experience of modelling ER models, this suggests that all three tools are rated approximately the same in terms of user experience, ease of use and overall satisfaction, within this experiment.

- **Question 1.3: Do the tools differ in the time and effort users spend to complete ER modelling tasks?**
  By not rejecting the null hypothesis, which states that the use of different tools does not influence the time spent modelling ER models, it can be inferred that the tools do not significantly differ in the time and effort users spend completing ER modelling tasks.

4.4.2 Other Results

In addition to the results already presented, open-ended questions were performed to the participants in the survey, where they had the opportunity to freely lay out some feedback on their experience with each tool.
The three open-ended questions given to the participants were:

- What was the best feature of this tool?
- What limitations or missing features did you notice?
- How would you improve this tool for academic use?

There was a lot of, and very diverse feedback. It was possible to find some consistencies between comments, and be able to assess some pros and cons within each tool.

ERDPlus

Starting with negative points on the ERDPlus tool:

- 11 out of the 20 participants (65%) stated that the creation of the generalisation element (ISA) was inconsistent with the creation of the rest of the elements, since most elements were presented in the top bar and the ISA could only be created by defining the Parent entity as a super type.
- 9 out of the 20 participants (45%) stated that ERDPlus' creation of diagrams involved too many clicks on the screen/canvas.
- 10 out of the 20 participants (50%) stated that one improvement to this tool could be to have SQL code generation.
- 7 out of 10 participants (35%) stated that the tool could offer keyboard shortcuts to facilitate usage - such as *Ctrl+C/V/Z* and *Delete*.

Moving on to the positive feedback on the ERDPlus tool:

- 10 out of the 20 participants (50%) stated that the tool had a clean and straightforward UI, with simplicity as its best feature.
- 6 out of the 20 participants (30%) found the tool very ERD focused.

Draw.io

Regarding Draw.io, and its negative feedback:

- 10 out of 20 participants (50%) stated that they had to manually edit and organise the diagram, which was hard.

- 17 out of 20 participants (85%) stated that the Draw.io tool does not have the generalisation element (ISA) in the dedicated section for ER modelling. The participants had to find any type of triangle in other sections to represent the ISA.
- 10 out of the 20 participants (50%) stated that one improvement to this tool could be to have SQL code generation.
- 6 out of the 20 participants (30%) stated that the tool was not ER modelling dedicated and because of it, it had too much information/noise.

Moving to the positive comments on Draw.io:

- 9 out of the 20 participants (45%) stated that the tool is very flexible, meaning it allows the user to build any kind of diagram.
- 5 out of the 20 participants (25%) enjoyed the Draw.io canvas.

BigER

Starting with BigER tool's cons:

- 7 out of the 20 participants (35%) stated the tool has a learning curve, by having to learn the syntax of the textual representation of the diagrams.
- 4 out of 20 participants (20%) stated that the tool had a weak graphical interface (diagram representation).
- 2 out of the 20 participants (10%) noticed the SQL script generated by the tool was not 100% correct.

In terms of BigER's pros:

- 12 out of the 20 participants (60%) were pleased that the diagram was organised automatically, as they were writing the textual representation.
- 8 out of the 20 participants (40%) pointed out that they enjoyed creating the diagram programmatically, since it is familiar to software developers and computer science students.
- 6 out of the 20 participants (30%) noticed the tool provided many DBMS SQL export options.

BigER was the only tool from the three which offered SQL exportation options. By being more than just a diagram editor, with model-driven development sustaining it, it has a meta-model to define the Entity-Relationship Diagrams and brings validation to the picture, hence opening the doors to generation of code from the definition of diagrams.

4.4.3 Threats to Validity

According to Wohlin et al.[49], an assessment of threats to validity was conducted in order to critically evaluate the reliability and generalisability of the experimental results, as well as to provide transparency and strengthen the credibility of the conclusions drawn from the study.

4.4.3.1 Internal Validity

Beginning with internal validity, which directly relates to the control over experimental conditions and their influence on observed effects.
In this experiment, one of the threats to internal validity was the potential learning or fatigue effect from participants completing multiple tasks with different tools. To mitigate this, participants were divided into groups with different tool usage orders, ensuring that the sequence in which each tool was used varied across participants. This counterbalancing aimed to reduce any systematic bias caused by the order of tool exposure.
Another threat to internal validity was the instrumentation. Since time was measured manually, it is possible that some inconsistencies may have risen from that.
The last threat to internal validity identified is regarding testing effect, where participants might've performed better in later tasks simply because they had already completed similar ones before.

4.4.3.2 External Validity

External validity refers to the extent to which the results of this experiment can be generalised beyond the specific context in which it was conducted. In this study, some factors may limit such generalisation, and these should be considered when interpreting the findings.
One potential threat to external validity is task realism. While the tasks used in the experiment were designed to resemble real-world ER modelling scenarios, they were necessarily simplified to ensure consistency across participants and time constraints. As such, the complexity and variability present in real software development or academic modelling tasks may not be fully captured, which may limit the generalisability of the results to more complex or varied modelling environments.
Another threat to external validity is the fact that the participants' sample does not fully represent the population. Having nineteen male participants and only one female participant does not reflect on the real impacted audience.
Another possible threat is the sampling method for the selection of participants. Since the choice of participants was through convenience sampling, the set of people collected for this experiment could have had specific characteristics or traits in common, which could bring some bias and affect the validity of the experiment.
Another threat concerns tool familiarity. Some participants reported prior experience with

one of the tools - particularly Draw.io - which may have influenced their performance and perceptions of usability. Familiarity with a tool could result in faster task completion times and more favourable usability ratings, independent of the tool's actual capabilities. This prior exposure introduces a potential bias that could affect the generalisation of the results to users encountering the tools for the first time.
Finally, the fact that the interviews were conducted remotely may bring some inconsistencies and biases related to possible latencies or interruptions due to internet issues.

4.4.3.3 Conclusion Validity

Conclusion validity refers to the extent to which the conclusions drawn from the results of an experiment are reasonable, credible, and statistically sound. It concerns the degree to which a relationship between variables is correctly identified - or rejected - based on the data and the methods used. In this experiment, several factors may pose threats to conclusion validity.
One potential threat is the limited sample size. Although the number of participants was sufficient to perform statistical analysis, a relatively small sample can reduce statistical power, increasing the risk of Type II errors - failing to detect real effects. Moreover, small samples can also make the results more sensitive to outliers or individual variation.
Another threat involves the choice of statistical tests and assumptions. While appropriate tests were selected based on data distribution and design (e.g., one-way repeated measures ANOVA, Friedman's test, and paired t-tests), violations of assumptions like normality or sphericity may affect the robustness of the results. To control for Type I errors in post hoc pairwise comparisons, both Holm's and Bonferroni corrections were applied. While effective in minimizing false positives, these corrections - especially Bonferroni - are conservative and may increase the likelihood of Type II errors, potentially masking real differences between tools. Holm's correction offers a more balanced approach, reducing this risk compared to Bonferroni, but the possibility of overlooking true effects remains a threat to conclusion validity.
Lastly, measurement reliability could also influence conclusion validity. Although validated instruments like the System Usability Scale (SUS) were used to collect subjective usability data, there is always a possibility that participants misunderstood questions or responded inconsistently, which could add noise to the data and obscure actual differences.

4.4.3.4 Construct Validity

Construct validity concerns whether the experiment's measurements accurately capture the intended concepts [49]. In this study, the constructs were expressiveness, user experience, and effort in ER modelling tools.
The System Usability Scale (SUS) was adapted for both expressiveness and user experience evaluation. While SUS is a widely used and validated usability instrument, splitting it into two separate constructs may affect its original psychometric properties. This could

influence how accurately it reflects participants' perceived usability in each category. Task completion time was used as the main quantitative measure of effort. However, time alone may not fully capture all aspects of effort, such as mental workload, frustration, or perceived complexity, which could lead to a partial representation of the construct.

5

Feasibility Study using a Cloud-based MDE Modelling Workbench

This chapter presents the results of a Technological Feasibility Study, conducted within the scope of a model-driven development of an ER modelling tool.

5.1 Introduction

5.1.1 Sirius Web

The framework Sirius Web was the model-driven option chosen for the development of this ER modelling tool.
Sirius Web is an open-source framework, developed under the Eclipse Foundation, for creating domain-specific modelling tools that run directly in a web browser. It extends the philosophy of Eclipse Sirius - which allowed the definition of graphical modelling editors in the desktop Eclipse platform - into a modern, web-based environment. Sirius Web provides a low-code approach where developers configure how domain models (defined in EMF/Ecore) are represented through diagrams, forms, trees, or tables, rather than building graphical editors from scratch.[43]
Technically, Sirius Web is structured as a Spring Boot backend that manages models and exposes a GraphQL API, coupled with a React frontend that renders interactive editors. This architecture enables model creation, editing, and persistence in collaborative and cloud-based scenarios, eliminating the installation overhead of desktop tools.[10]
Sirius Web has an advantage since it allows developers to integrate the graphical editor in their own applications. [19]

5.1.2 Epsilon

Epsilon [12] is a group of scripting languages and tools for automating common model-based software engineering tasks, such as model-to-model transformations, code generation, and model validation.
There are some advantages of Epsilon compared to other technologies which are worth mentioning. Epsilon's languages are all built on top of this common expression language, the Epsilon Object Language (EOL). EOL serves as the primary expression language in the Epsilon framework, as it forms the basis for creating specialised languages tailored for specific tasks such as model validation, transforming models into other models or text, and model migration. EOL has an imperative style similar to Java and the functional model capabilities of Object Constraint Language (OCL). [37]
Epsilon also includes languages for code generation (Epsilon Generation Language (EGL)), model validation (Epsilon Validation Language (EVL)), model transformation (Epsilon Transformation Language (ETL)) and much more.

5.1.3 Initial Plan

The initial plan for this development was to rely on Sirius Web as the model-driven solution for the modelling layer.
Within Sirius Web's backend, the meta-model for Entity-Relationship Diagrams would be defined, and then it would be stored on a database server. Sirius Web provides a GraphQL API which acts as the communication layer between the frontend and backend, allowing for efficient and flexible data retrieval and manipulation. User interactions in the frontend trigger GraphQL queries and mutations to retrieve or modify data. In Sirius Web's frontend, components would be adapted to present to the user only what they need to build an ERD.
Additional validation rules were to be specified using Epsilon's Validation Language, while Epsilon's Transformation Language and Generation Language would transform the validated models into SQL code.
As shown in Figure 5.1, the frontend was intended to provide the user-facing diagram editor. The backend would coordinate model management and interact with the Epsilon framework for validation and transformations. The resulting SQL scripts would be delivered to the frontend for download.

5.2 Development Process

The process of this model-driven solution's development roughly follows the Software Language Engineering process mentioned in 2.3.

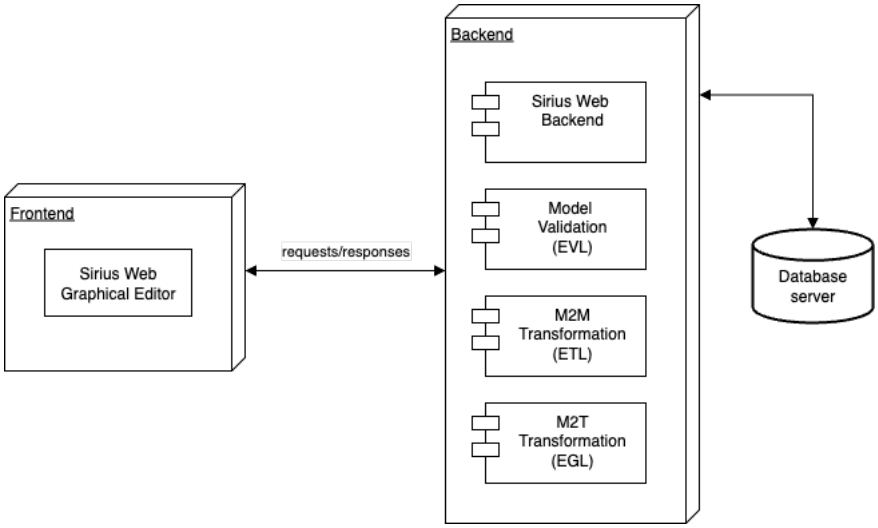


Figure 5.1: Planned architecture for the integration of Sirius Web and Epsilon

5.2.1 Domain Analysis

The use case at hand was thoroughly analysed and the domain was understood. From the juice of what was analysed, the outcome was a deep understanding of the domain to be worked on, as well as a initial sketch of what the domain model would become.

5.2.2 Abstract and Concrete Syntax Definition

To develop this modelling language's formal structure, the idea was to use Sirius Web, as mentioned previously.
The first step was to understand how to integrate Sirius Web into an application. The main reason for this integration was to enable the future use of other technologies (like Epsilon) to perform further enhancements in functionality, other than simply diagram creation.
Through Sirius Web's documentation, which did not offer much information about this integration at the time, this first step was not reached successfully through the main route. There was a point of contact with the Obeo team, where the question of where the documentation for a possible integration was located was put, and the answer was that there was no official documentation at the time, and that the question should be posted on their forum for someone from the Obeo team to answer. As advised, the question was put and there was no timely response.
An alternative solution was then necessary, and from Eclipse Sirius' GitHub repository [10], the code of a sample application which leverages the Sirius Components was available and could be forked and altered according to this use case's needs.
From the alternative solution, the code was adapted to have a meta-model of an Entity-Relationship Diagram, with all of its elements. This was built through the sample application, as already mentioned, by mimicking what was already done and adapting it to the structure of an Entity-Relationship Diagram.
The meta-model in Sirius Web is composed by:

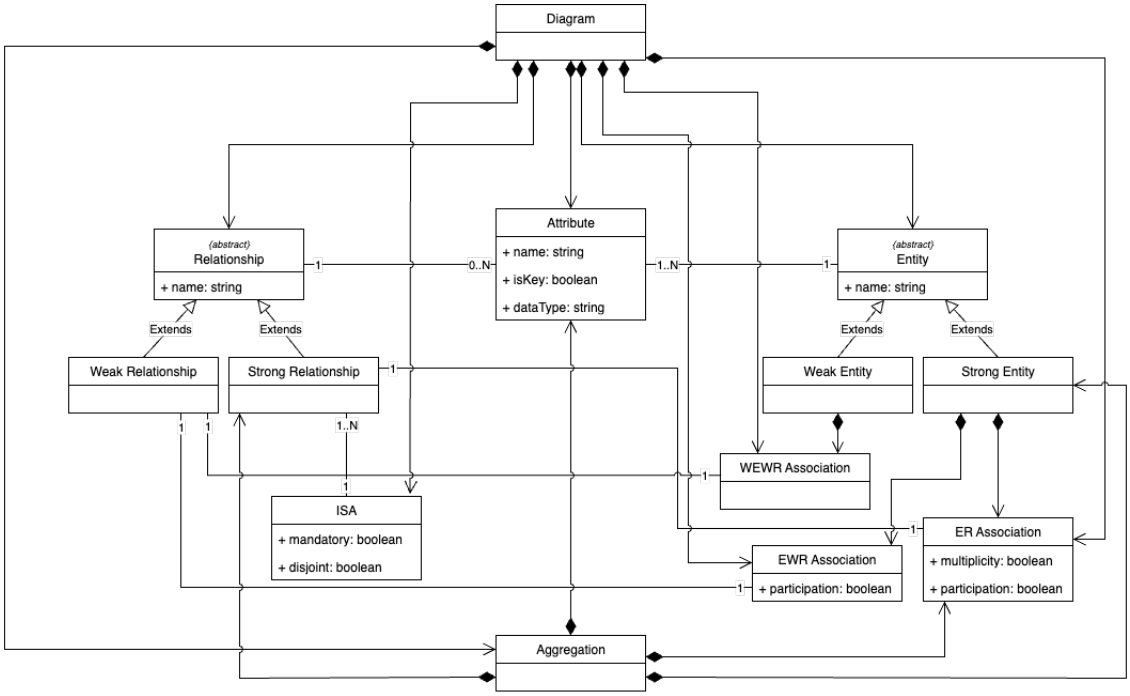


Figure 5.2: Domain diagram

- A domain, which is responsible for defining the elements which will exist in the models and how they are connected with each other - a meta-model's abstract syntax;
- And a view, which is responsible for defining the visual representations of the elements defined in the domain - a meta-model's concrete syntax.

The domain was defined in code, in Sirius Web's backend, having in consideration all of the elements an ER model may contain. When compiling the code and running it locally, it was possible to extract the visual representation of the domain defined in the Sirius Web application, as can be seen in Figure 5.3. A clearer and more readable diagram can be seen in Figure 5.2 to have a better understanding of the domain built in Sirius Web.
The view was also defined in code, and an example snippet of the code, where the view for a "Strong" Entity Node for an ER model is defined, can be seen in Figure 5.4. When testing the application, it was possible to create diagrams just like the one shown in Figure 5.5, where the thicker lines are meant to represent the double lines.
In ER modelling, there are some restrictions and rules, such as:

- Entities only connect to Relationships and Attributes, not being able to connect directly to other Entities;
- Weak Entities only connect to Weak Relationships;
- Relationships only connect to Attributes;
- Generalisations have only one parent/super Entity;

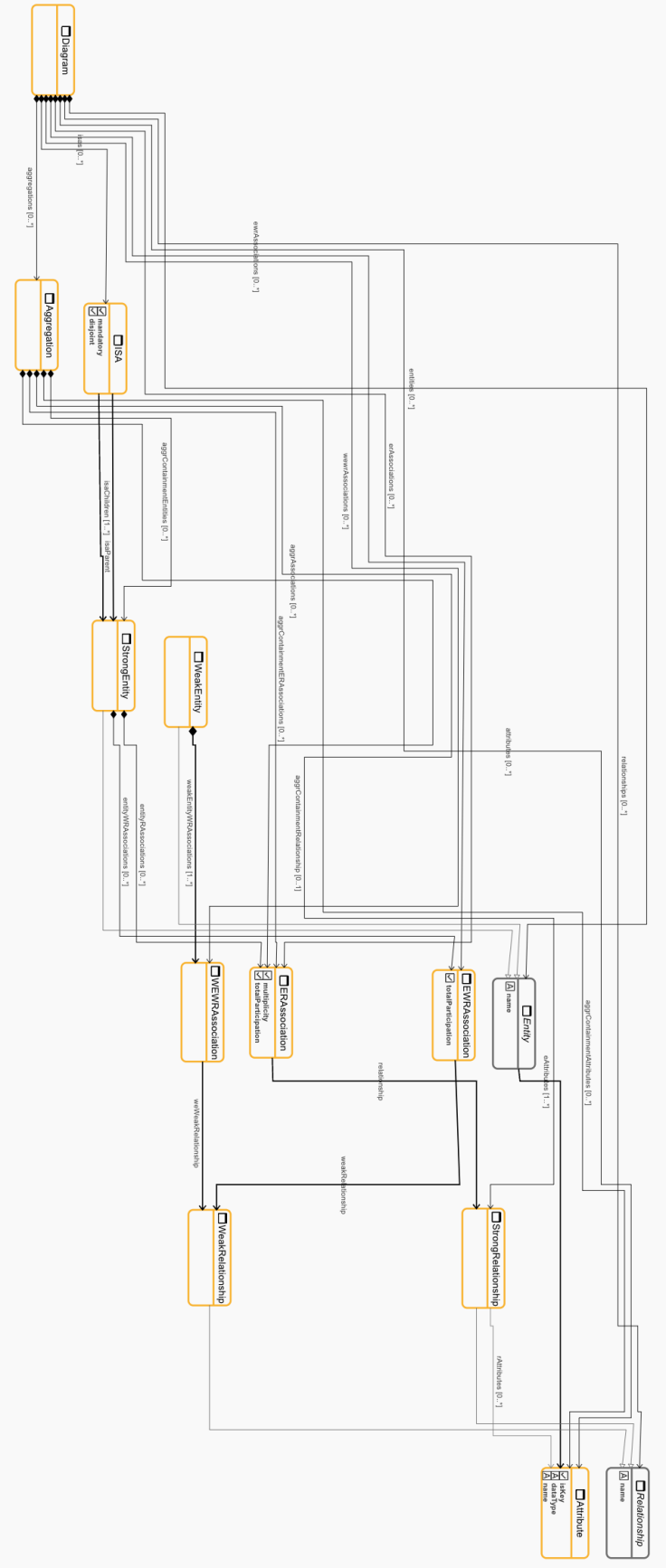


Figure 5.3: Defined domain of an Entity-Relationship Diagram in Sirius Web

```
public class StrongEntityNodeDescriptionProvider implements INodeDescriptionProvider {
    public static final String NAME = "Entity";

    @Override
    public EObject create() {
        var nodeStyle = ViewFactory.eINSTANCE.createRectangularNodeStyleDescription();
        nodeStyle.setColor(value:"#FFFFFF00");
        nodeStyle.setBorderColor(value:"black");
        nodeStyle.setLabelColor(value:"black");
        nodeStyle.setWithHeader(value:false);

        var builder = new ERDEditorViewBuilder();
        var domainType = builder.domainType(builder.entity(name:"StrongEntity"));

        var nodeDescription = ViewFactory.eINSTANCE.createNodeDescription();
        nodeDescription.setName(NAME);
        nodeDescription.setDomainType(domainType);
        nodeDescription.setUserResizable(value:false);
        nodeDescription.setSemanticCandidatesExpression(value:"aql:self.entities");
        nodeDescription.setChildrenLayoutStrategy(ViewFactory.eINSTANCE.createFreeFormLayoutStrategyDescription());
        nodeDescription.setLabelExpression(value:"aql:self.name");
        nodeDescription.setStyle(nodeStyle);
        nodeDescription.setSynchronizationPolicy(SynchronizationPolicy.SYNCHRONIZED);

        return nodeDescription;
    }
```

Figure 5.4: Snippet of code where an Entity node's view is defined

- Generalisations' parent/super Entity and children Entities cannot be Weak Entities;
- Associations/connections between an Entity and a Weak/Identifying Relationship have only participation constraints, there are no multiplicity constraints possible since at most one Entity can participate in the relationship.
- Associations/connections between a Weak Entity and its Weak Relationship have no participation constraints, since a Weak Relationship always has total participation from the Weak Relationships.

These listed restrictions were possible to implement in the code, and in a very straightforward way, because as the domain was being built and defined, these restrictions came up naturally. Some of more complex restrictions, were planned to be implemented through Epsilon's Validation Language, restrictions such as:

- The same Attribute cannot be connected to two different Entities/Relationships;
- Each Entity should have only one key Attribute;
- A Relationship must have at least two Entities connected to it;
- A Weak Entity must have always be connected to a Strong Entity through a Weak/Identifying Relationship;

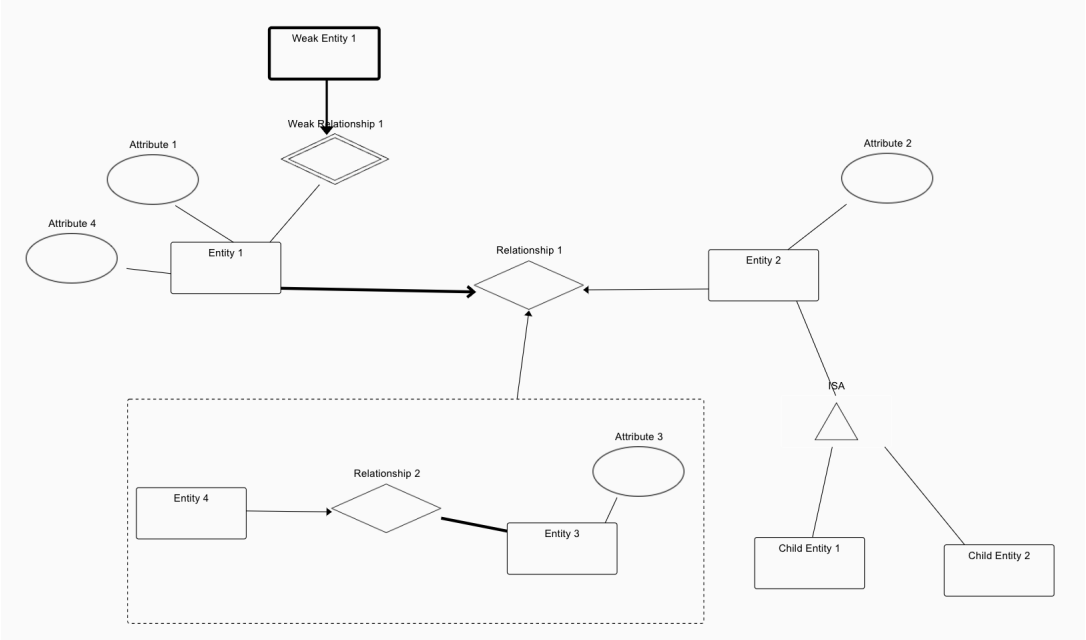


Figure 5.5: Example of an Entity-Relationship Diagram modelled through the application

5.2.3 Limitations

The Sirius Web framework ended up showing some limitations in terms of usability, even from a developer's perspective. Moving elements around in the canvas was challenging, creating new elements was not very intuitive, organising the diagram was not easy. The overall user experience would not be pleasant.
That is the main reason why this feasibility study did not move forward with development. The usability of the platform was not satisfactory, and could not be delivered as an option to model Entity-Relationship Diagrams.

5.3 Possible Alternatives

Given the usability limitations encountered with Sirius Web, other frameworks and technologies can be explored as potential solutions for building web-based diagram editors. The next two alternatives presented are very recent technologies, still under development, but they seem very promising.
Eclipse GLSP (Graphical Language Server Platform) [18] is an open-source framework for creating diagram editors based on web technologies. It follows a client–server architecture, where the server handles the language logic (loading, interpreting, editing) and the client manages rendering. Instead of abstracting away rendering, GLSP builds on technologies such as Eclipse Sprotty, SVG, and CSS, enabling scalable, interactive, and fully tailored diagrams. Beyond core editor features like editing, layout, shapes, and palettes, GLSP provides flexible integration options: it can be embedded directly into web pages, deployed as a stand-alone web application, or integrated with popular IDEs such as Eclipse Theia,

VS Code, and the Eclipse desktop environment.

EMF Cloud [17] is a collection of open-source technologies for building web-based modelling tools. It provides modular components such as a model hub for API-based access and manipulation, an editing domain for managing model state, and libraries for integrating graphical, form-based, or textual editors. EMF Cloud is designed around modern web technologies, using Typescript, Node.js, and React, and integrates seamlessly with frameworks like Eclipse GLSP for graphical representations. Its flexible architecture allows components to be combined, customized, or replaced, supporting diverse deployment scenarios ranging from frontend-only browser applications to cloud-based backends. EMF.cloud also offers integration with platforms such as VS Code, Eclipse Theia, or standalone web applications.

Several other modelling workbench options besides Sirius Web have been proposed in the literature and in practice. However, each of the following options presents its own limitations regarding feasibility, adoption, or suitability for the intended purpose of supporting ER diagram editors in the web.

MetaEdit+[1] is a mature tool but does not provide web-based support, which limits its integration in modern cloud-native applications.

JetBrains' MPS[2] is a robust platform for defining domain-specific languages, but it also lacks web support, limiting its use in scenarios where web-based accessibility is a key requirement.

Gentleman[3], on the other hand, does provide web support, but relies on a projectional approach where models are defined in a tree-like view and only later visualized in a concrete syntax. This is less aligned with the intended activity of directly creating and modifying ER diagrams in their native visual notation.

CINCO Cloud[4] is an academic product, with limited evidence of adoption and no direct integration with EMF-based technologies, particularly regarding model generation and validation.

Jjodel[5], is more suited for small projects, with some performance limitations and does not integrate with EMF. [46]

Finally, **WebGME**[6] is also an academic product, with reduced built-in graphical richness and limited adoption, while likewise lacking integration with EMF technologies for generation and validation.

These alternatives demonstrate the breadth of options available but also highlight the challenges in finding a solution that is both technically robust, web-enabled, and well-supported in the modelling community.

[1] https://www.metacase.com/products.html
[2] https://www.jetbrains.com/mps/
[3] https://gentlemancp.org/about
[4] https://scce.gitlab.io/cinco-cloud/
[5] https://www.jjodel.io/
[6] https://webgme.org/

6

Conclusions and Future Work

6.1 Conclusions

This study set out to evaluate Entity-Relationship (ER) modelling tools in terms of effectiveness/expressiveness, user experience, and effort (time required), with the aim of identifying their relative strengths and weaknesses when used in academic and professional contexts.
The research began with a Systematic Literature Review (SLR), which served to map the current state of ER modelling tools in the market. The SLR highlighted a variety of available solutions, ranging from simple, free tools to advanced, commercially licensed software. Based on some constraints - namely, cost (free or offering a free tier), adherence to Chen's notation, and similarity in functionality - three tools were selected for experimental evaluation: Draw.io, ERDPlus, and BigER.
A controlled experiment was designed following best practices in software engineering experimentation. The experiment adopted a within-subjects design, in which 20 participant used all three tools to complete ER modelling tasks, and the order of tool usage was counterbalanced to minimise learning and fatigue effects.
The dependent variables under study were:

- Effectiveness, measured through the System Usability Scale (SUS) adapted for functional aspects.
- User experience usability, measured through SUS items focusing on ease of navigation, aesthetics, and overall satisfaction.
- Time taken to complete predefined ER modelling tasks, to measure effort.

Appropriate statistical methods were selected according to the data characteristics. For data meeting assumptions of normality and sphericity, a one-way repeated measures ANOVA was applied; for non-parametric cases, Friedman's test was used. Post-hoc analyses employed paired-samples t-tests or their non-parametric equivalents, with both Holm's correction and Bonferroni correction applied to control the family-wise error rate.
The results revealed distinct patterns across the three variables:

- **Effectiveness**: The null hypothesis was rejected, with statistical evidence showing differences among the tools. Post-hoc analysis indicated that BigER consistently outperformed the other two tools in this dimension.
- **User experience usability**: The null hypothesis was retained, indicating no statistically significant differences among the three tools in terms of navigation, aesthetics, or perceived satisfaction.
- **Time**: The null hypothesis was also retained, suggesting that, within the scope of this experiment, none of the tools led to significantly faster or slower task completion times.

These findings suggest that while BigER offers clear advantages in functional usability, this does not necessarily translate into better perceived user experience or measurable efficiency gains in short-term modelling tasks. This highlights the complexity of tool evaluation, where a strong performance in one dimension may not guarantee superiority in others.
The qualitative insights found with the open feedback presented in Section 4.4.2, complement the statistical results, revealing practical trade-offs that pure numbers cannot capture. For instance, while BigER scored higher on effectiveness, its learning curve and weaker visual output could hinder adoption for beginners or visually-driven tasks. In contrast, Draw.io offers flexibility but lacks ERD-specific features, and ERDPlus provides a clean interface but can feel slow and repetitive in usage.
For contexts prioritising functional completeness and advanced export options, BigER is the strongest choice, since it offers model validation and SQL generation from the models - this because it is a tool developed using model-driven development, and contains a meta-model behind the construction of diagrams/models. For tasks requiring diagram flexibility across modelling types, Draw.io is a strong option despite its ERD limitations. For simplicity and ERD focus, ERDPlus can be effective, especially in beginner teaching scenarios.
The main conclusion emerging from this study is that, at present, there is no ER modelling tool that fully satisfies all key requirements simultaneously. Specifically, none of the tools evaluated respect Chen's notation in full, provide built-in diagram validation, support automatic SQL code generation, and offer a user-friendly, intuitive interface, all in one. This gap highlights the need for further development in this area to deliver a tool that integrates these capabilities in a consistent and usable way.
As part of this research, a technological feasibility study was conducted (in Chapter 5) with the initial aim of developing such a tool using Sirius Web. However, this approach was ultimately set aside for the experiment presented in this dissertation due to significant usability limitations encountered during development. Nevertheless, the lessons learned from that preliminary work reinforce the conclusion that future research should continue to explore both the design and technical feasibility of building a complete ER modelling

environment that bridges conceptual modelling, validation, and code generation.

6.2 Future Work

While this study offers valuable insights, several opportunities remain to expand the research:

- Include more tools from the Systematic Literature Review to broaden the comparison and generalisability.
- Increase participant diversity to better reflect different backgrounds, including industry professionals and novice users.
- Conduct longitudinal studies to assess how tool preference and efficiency evolve with long-term use.
- Use broader effort measures, such as NASA-TLX, to capture mental workload beyond time metrics.
- Conduct a comparative study between UML and ER modelling to understand the pros and cons of each modelling language.
- Explore the possibility of integrating Artificial Intelligence into a solution to generate ER diagrams from pre-defined requisites.

This study lays a foundation for further exploration of ER modelling tools, both in academic and professional settings. Future research can be built on these findings in several ways. First, the methodology applied here - combining SUS-based usability assessments, time/effort measurements, and qualitative feedback - can be replicated or adapted to evaluate other modelling tools or notations beyond Chen's. This allows researchers to directly compare results across different contexts and user populations.
Second, the detailed strengths and weaknesses identified for each tool provide valuable guidance for tool developers. For example, BigER's functional advantages could be further enhanced by addressing graphical interface concerns, while Draw.io could benefit from incorporating missing ER-specific elements and features such as automated diagram organisation. Both ERDPlus and Draw.io could incorporate some type of validation for further development and possible SQL generation capabilities.
Ultimately, a good and complete ER modelling tool is necessary. It is not just a piece of software - it is the bridge between abstract ideas and clear, shared understanding on database modelling.

Bibliography

[1] *About draw.io*. URL: https://www.drawio.com/about (cit. on p. 23).

[2] *About Visual Paradigm*. URL: https://www.visual-paradigm.com/aboutus/ (cit. on p. 23).

[3] *Architecture Overview*. URL: https://www.eclipse.org/sirius/doc/developer/Architecture_Overview.html (cit. on p. 13).

[4] V. R. Basili, G. Caldiera, and H. D. Rombach. "The Goal Question Metric Approach". In: 1994. URL: https://api.semanticscholar.org/CorpusID:13884048 (cit. on p. 28).

[5] C. Batini, S. Ceri, and S. Navathe. *Conceptual Database Design: An Entity-relationship Approach*. Benjamin/Cummings series in computer science and engineering. Benjamin/Cummings Publishing Company, 1992. ISBN: 9780805302448. URL: https://books.google.es/books?id=HeFQAAAAMAAJ (cit. on p. 1).

[6] M. Brambilla, J. Cabot, and M. Wimmer. *Model-Driven Software Engineering in Practice*. Vol. 1. 2012-09. DOI: 10.2200/S00441ED1V01Y201208SWE001 (cit. on pp. 10, 11).

[7] J. Cabot. *The Entity Relationship language (ER) is stronger than ever*. URL: https://modeling-languages.com/entity-relationship-language-er-stronger-than-ever/ (cit. on pp. 1, 4).

[8] P. P.-S. Chen. "The Entity-Relationship Model—toward a Unified View of Data". In: *ACM Trans. Database Syst.* 1.1 (1976-03), pp. 9–36. ISSN: 0362-5915. DOI: 10.1145/320434.320440. URL: https://doi.org/10.1145/320434.320440 (cit. on pp. 1, 4, 6, 22).

[9] P. P.-S. Chen. "The Entity-Relationship Model: A Basis for the Enterprise View of Data". In: *Proceedings of the June 13-16, 1977, National Computer Conference*. AFIPS '77. Dallas, Texas: Association for Computing Machinery, 1977, pp. 77–84. ISBN: 9781450379144. DOI: 10.1145/1499402.1499421. URL: https://doi.org/10.1145/1499402.1499421 (cit. on p. 4).

[10] Eclipse-Sirius. *GitHub - eclipse-sirius/sirius-web: Sirius Web: open-source low-code platform to define custom web applications supporting your specific visual languages*. en. URL: https://github.com/eclipse-sirius/sirius-web (cit. on pp. 49, 51).

[11] R. Elmasri and S. Navathe. *Fundamentals of database Systems*. 2016-01 (cit. on p. 1).

[12] *Epsilon*. URL: https://eclipse.dev/epsilon/ (cit. on p. 50).

[13] *ER Diagram (ERD) Tool | Lucidchart*. URL: https://www.lucidchart.com/pages/examples/er-diagram-tool (cit. on p. 21).

[14] K. Fergusson. "Entity Relationship Diagrams with draw.io". In: *draw.io* (2021-12). URL: https://drawio-app.com/blog/entity-relationship-diagrams-with-draw-io/ (cit. on p. 23).

[15] A. Field. *Discovering Statistics using IBM SPSS Statistics*. 4th. Sage Publications Ltd., 2013. ISBN: 1446249182 (cit. on pp. 37, 38).

[16] C. T. Force. *Computing Curricula 2020: Paradigms for Global Computing Education*. New York, NY, USA: Association for Computing Machinery, 2020. ISBN: 9781450390590 (cit. on p. 1).

[17] E. Foundation. *EMF Cloud*. URL: https://eclipse.dev/emfcloud/ (cit. on p. 56).

[18] E. Foundation. *GLSP*. URL: https://eclipse.dev/glsp/#overview (cit. on p. 55).

[19] E. Foundation. *Sirius Web 101: Create a Modeler With No Code*. 2022-11. URL: https://www.youtube.com/watch?v=p_tDEzGtS0o (cit. on pp. 13, 49).

[20] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Signature Series (Fowler). Pearson Education, 2010. ISBN: 9780131392809. URL: https://books.google.pt/books?id=ri1muolw_YwC (cit. on pp. 7, 8).

[21] S. I. Frankel et al. "Model Driven Architecture (MDA)". In: (2001) (cit. on p. 11).

[22] P. Glaser and D. Bork. "The bigER Tool - Hybrid Textual and Graphical Modeling of Entity Relationships in VS Code". In: *2021 IEEE 25th International Enterprise Distributed Object Computing Workshop (EDOCW)* (2021), pp. 337–340. URL: https://api.semanticscholar.org/CorpusID:244831429 (cit. on p. 19).

[23] *Introduction | dbdiagram Docs*. URL: https://dbdiagram.io/docs/ (cit. on p. 22).

[24] A. Jedlitschka, M. Ciolkowski, and D. Pfahl. "Reporting Experiments in Software Engineering". In: 2008-01, pp. 201–228. ISBN: 978-1-84800-043-8. DOI: 10.1007/978-1-84800-044-5_8 (cit. on pp. 2, 3).

[25] N. Jukic et al. "Data modeling in the cloud". In: (2013) (cit. on p. 21).

[26] N. Juristo and A. Moreno. *Basics of Software Engineering Experimentation*. Springer US, 2013. ISBN: 9781475733044. URL: https://books.google.pt/books?id=iJTkBwAAQBAJ (cit. on p. 3).

[27] S. Kesh. "Evaluating the quality of entity relationship models". In: *Information and Software Technology* 37.12 (1995), pp. 681–689. ISSN: 0950-5849. DOI: https://doi.org/10.1016/0950-5849(96)81745-9. URL: https://www.sciencedirect.com/science/article/pii/0950584996817459 (cit. on p. 6).

[28] B. Kitchenham and S. Charters. "Guidelines for performing Systematic Literature Reviews in Software Engineering". In: 2 (2007-01) (cit. on p. 14).

[29] Y. Levy and T. Ellis. "A Guide for Novice Researchers on Experimental and Quasi-Experimental Studies in Information Systems Research". In: *Interdisciplinary Journal of Information* 6 (2011-01). DOI: 10.28945/1373 (cit. on pp. 28, 32).

[30] S. W. Liddle. "Model-driven software development". In: *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. Springer, 2011, pp. 17–54 (cit. on pp. 9, 11).

[31] J. Lopes. et al. "Entity-relationship Modeling Tools and DSLs: Is It Still Possible to Advance the State of the Art from Observations in Practice?" In: *Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, INSTICC. SciTePress, 2022, pp. 179–186. ISBN: 978-989-758-569-2. DOI: 10.5220/0011044500003179 (cit. on p. 21).

[32] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[33] R. McFadyen. "AN ERD TOOL". In: (2015) (cit. on p. 19).

[34] S. J. Mellor et al. "Model-driven architecture". In: *International Conference on Object-Oriented Information Systems*. Springer. 2002, pp. 290–297 (cit. on p. 11).

[35] P. Melly. "Real-time multi-user collaborative diagramming". In: *draw.io* (2022-10). URL: https://drawio-app.com/blog/real-time-multi-user-collaborative-diagramming/ (cit. on p. 23).

[36] P. Mohagheghi et al. "Where does model-driven engineering help? Experiences from three industrial cases". In: *Software and Systems Modeling* 12.3 (2013-07), pp. 619–639. DOI: 10.1007/s10270-011-0219-7. URL: https://doi.org/10.1007/s10270-011-0219-7 (cit. on p. 10).

[37] *Object Language (EOL) - Epsilon*. URL: https://eclipse.dev/epsilon/doc/eol/ (cit. on p. 50).

[38] S. Patig. "Evolution of entity–relationship modelling". In: *Data & Knowledge Engineering* 56.2 (2006), pp. 122–138. ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2005.03.010. URL: https://www.sciencedirect.com/science/article/pii/S0169023X05000339 (cit. on p. 4).

[39] D. Phung. "Implementation of graphical editor using sirius". In: (2018) (cit. on p. 13).

[40] S. Robinson et al. "Conceptual Modeling: Definition, Purpose, and Benefits". In: 2015-12. DOI: 10.1109/WSC.2015.7408386 (cit. on p. 2).

[41] D. Rosca and L. Domingues. "A Systematic Comparison of Roundtrip Software Engineering Approaches applied to UML Class Diagram". In: *Procedia Computer Science* 181 (2021). CENTERIS 2020 - International Conference on ENTERprise Information Systems / ProjMAN 2020 - International Conference on Project MANagement / HCist 2020 - International Conference on Health and Social Care Information Systems and Technologies 2020, CENTERIS/ProjMAN/HCist 2020, pp. 861–868. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.01.240. URL: https://www.sciencedirect.com/science/article/pii/S1877050921002830 (cit. on p. 9).

[42] *Sirius | Overview*. URL: https://www.eclipse.org/sirius/overview.html (cit. on p. 13).

[43] *Sirius | Sirius WebHome*. URL: https://eclipse.dev/sirius/sirius-web.html (cit. on p. 49).

[44] B. Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer, 2000. ISBN: 9783540654704. URL: https://books.google.pt/books?id=Xc3HAnPVQAYC (cit. on pp. 5, 6).

[45] R. Tiwari. *9 ER Model Tools to use in 2023: A Comprehensive List - Learn | Hevo*. 2023-01. URL: https://hevodata.com/learn/er-model-tools/#l9 (cit. on p. 21).

[46] J.-P. Tolvanen et al. "A framework for evaluating tool support for co-evolution of modeling languages, tools and models". In: *Software Systems Modeling* (2024-10). DOI: 10.1007/s10270-024-01218-5. URL: https://doi.org/10.1007/s10270-024-01218-5 (cit. on p. 56).

[47] V. Viyović, M. Maksimović, and B. Perisić. "Sirius: A rapid development of DSM graphical editor". In: *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*. 2014, pp. 233–238. DOI: 10.1109/INES.2014.6909375 (cit. on p. 13).

[48] *What is Creately? - Creately Help Center*. URL: https://support.creately.com/hc/en-us/articles/360001418815-What-is-Creately (cit. on p. 24).

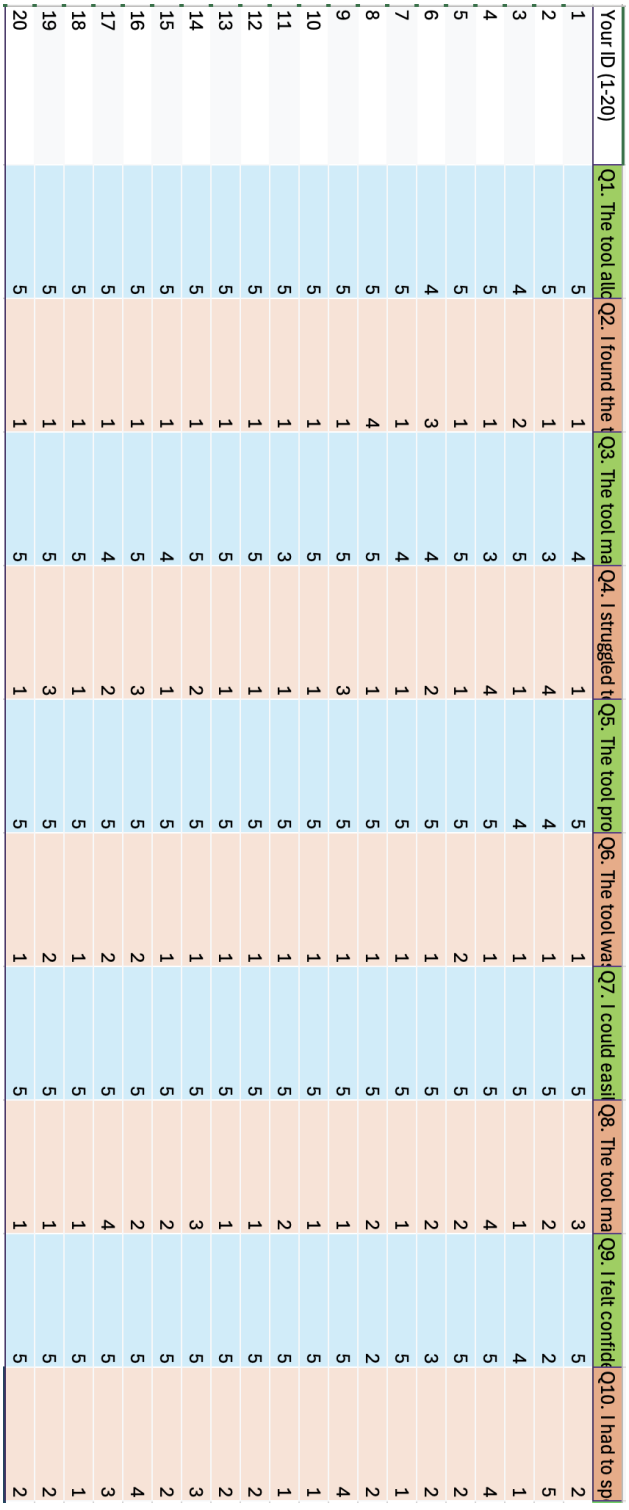
[49] C. Wohlin et al. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. ISBN: 3642290434 (cit. on pp. 46, 47).

I

Raw SUS Scores for Effectiveness

	Your ID (1-20)	Q1. The tool all	Q2. I found the	Q3. The tool ma	Q4. I struggled t	Q5. The tool pro	Q6. The tool wa	Q7. I could easi	Q8. The tool ma	Q9. I felt confid	Q10. I had to sp
1		5	1	3	2	1	1	5	2	3	1
2		3	5	5	4	1	2	3	4	2	1
3		5	1	3	1	1	2	4	1	5	4
4		5	4	2	4	1	4	5	1	2	2
5		5	1	4	2	1	4	4	4	5	1
6		4	3	4	4	1	4	3	4	2	3
7		5	4	4	1	1	1	4	1	4	1
8		5	2	5	2	1	4	3	2	2	2
9		5	1	4	2	1	1	4	1	2	2
10		5	3	3	4	1	2	5	2	3	2
11		5	1	2	2	1	4	4	4	5	1
12		5	1	5	1	1	1	5	1	5	1
13		4	2	5	3	1	2	3	2	4	2
14		5	3	4	3	1	4	2	1	2	1
15		5	2	5	4	1	2	5	1	4	1
16		5	1	5	2	1	2	5	1	5	1
17		5	1	2	1	1	2	4	2	4	1
18		5	1	5	2	1	1	5	1	5	1
19		5	4	2	4	1	4	2	4	2	3
20		5	4	2	2	1	2	3	2	4	1

Figure I.1: Draw.io's raw SUS scores for effectiveness



Your ID (1-20)	Q1. The tool allo	Q2. I found the t	Q3. The tool ma	Q4. I struggled t	Q5. The tool pro	Q6. The tool wa	Q7. I could easil	Q8. The tool ma	Q9. I felt confid	Q10. I had to sp
1	5	1	4	1	5	1	5	3	5	2
2	5	1	3	4	4	1	5	2	2	5
3	4	2	5	1	4	1	5	1	4	1
4	5	1	3	4	5	1	5	4	5	4
5	5	1	5	1	5	2	5	2	5	2
6	4	3	4	2	5	1	5	2	3	2
7	5	1	4	1	5	1	5	1	5	1
8	5	4	5	1	5	1	5	2	2	2
9	5	1	5	3	5	1	5	1	5	4
10	5	1	5	1	5	1	5	1	5	1
11	5	1	3	1	5	1	5	2	5	1
12	5	1	5	1	5	1	5	1	5	2
13	5	1	5	1	5	1	5	1	5	2
14	5	1	5	2	5	1	5	3	5	3
15	5	1	4	1	5	1	5	2	5	2
16	5	1	5	3	5	2	5	2	5	4
17	5	1	4	2	5	2	5	4	5	3
18	5	1	5	1	5	1	5	1	5	1
19	5	1	5	3	5	2	5	1	5	2
20	5	1	5	1	5	1	5	1	5	2

Figure I.2: BigER's raw SUS scores for effectiveness

Your ID (1-20)	Q1. The tool allowe	Q2. I found the	Q3. The tool ma	Q4. I struggled t	Q5. The tool prc	Q6. The tool wa	Q7. I could easi	Q8. The tool ma	Q9. I felt confid	Q10. I had to sp
1	5	3	3	3	1	1	4	3	4	4
2	5	4	4	3	1	4	4	4	2	3
3	5	1	2	3	1	1	3	3	4	5
4	5	4	5	2	1	1	5	1	5	1
5	5	1	5	2	1	1	5	1	4	1
6	4	3	3	2	1	4	2	4	3	5
7	5	1	4	4	1	2	5	4	5	2
8	5	2	5	2	1	1	5	1	4	1
9	5	1	4	1	1	2	5	1	4	1
10	5	1	1	3	1	4	4	5	2	1
11	5	1	5	3	1	4	5	1	5	1
12	5	1	5	1	1	1	5	1	5	1
13	5	1	4	1	1	4	5	1	4	2
14	5	2	5	2	1	1	4	2	4	2
15	5	1	5	2	2	4	5	1	5	2
16	5	4	5	4	1	2	5	1	5	1
17	5	1	5	1	1	1	2	1	4	1
18	5	2	4	1	1	1	5	1	5	1
19	5	1	4	2	1	1	4	2	5	2
20	4	1	5	2	1	1	5	1	5	1

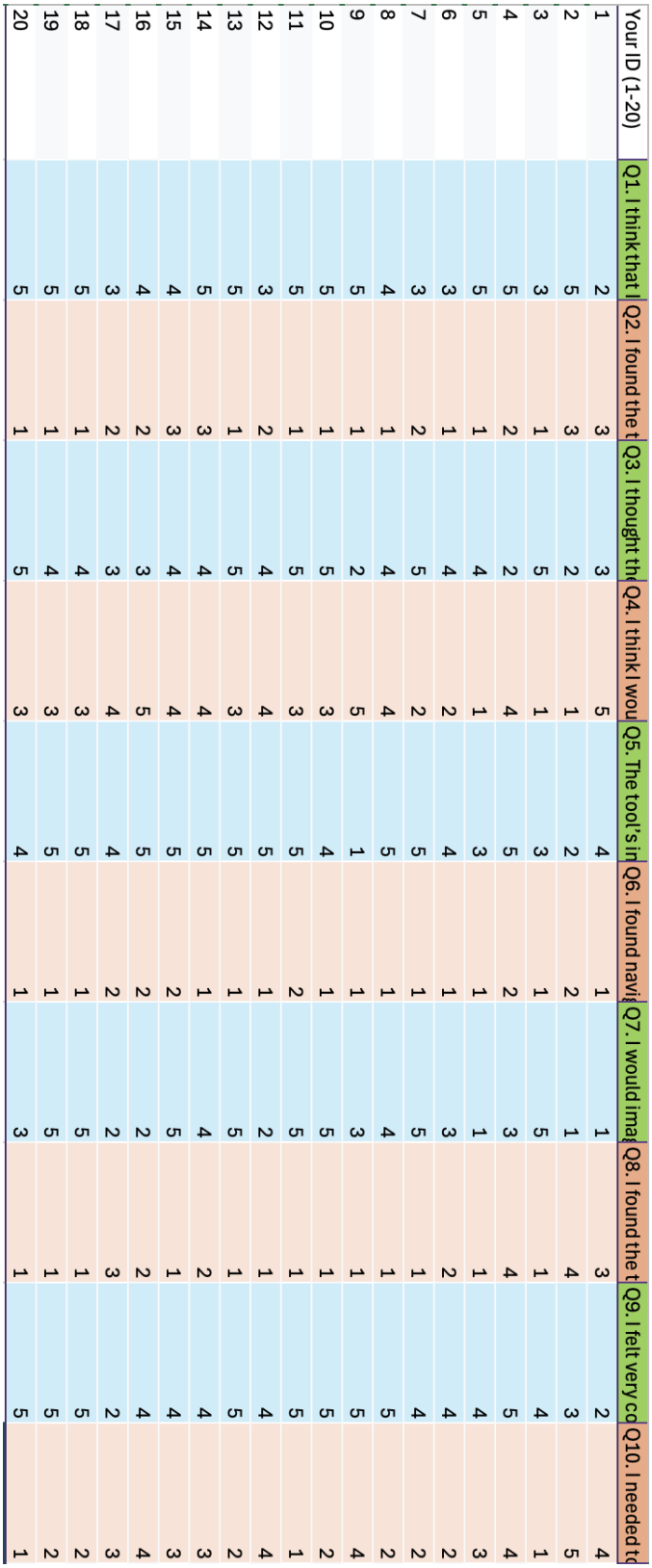
Figure I.3: ERDPlus' raw SUS scores for effectiveness

II

Raw SUS Scores for User Experience

Your ID (1-20)	Q1. I think that I	Q2. I found the t	Q3. I thought th	Q4. I think I wou	Q5. The tool's in	Q6. I found navi	Q7. I would ima	Q8. I found the t	Q9. I felt very co	Q10. I needed t
1	4	2	4	3	4	1	3	1	4	2
2	3	1	5	1	4	1	5	2	3	1
3	1	2	3	1	2	1	5	1	5	1
4	2	2	4	1	3	4	4	2	4	1
5	2	2	4	1	4	1	4	1	3	1
6	2	4	2	1	3	2	3	2	3	2
7	5	1	5	1	3	1	5	1	5	1
8	1	4	3	2	1	4	2	2	4	2
9	4	1	5	1	5	4	5	1	5	1
10	3	2	4	1	4	2	4	2	4	1
11	3	2	4	2	4	2	4	3	4	2
12	4	1	5	1	5	1	5	1	5	1
13	3	2	4	3	4	2	3	1	4	2
14	1	2	5	1	3	1	5	1	5	1
15	3	3	4	1	4	2	5	1	5	2
16	5	1	5	2	4	1	4	1	5	2
17	4	4	4	2	2	2	4	2	4	1
18	1	1	5	1	5	2	5	1	5	1
19	1	4	3	3	3	4	3	2	1	2
20	2	2	3	1	1	3	2	3	4	2

Figure II.1: Draw.io's raw SUS scores for user experience



Your ID (1-20)	Q1. I think that I	Q2. I found the t	Q3. I thought th	Q4. I think I wou	Q5. The tool's in	Q6. I found navi	Q7. I would ima	Q8. I found the t	Q9. I felt very co	Q10. I needed to
1	2	3	3	5	4	1	1	3	2	4
2	5	3	2	1	2	2	1	4	3	5
3	3	1	5	1	3	1	5	1	4	1
4	5	2	2	4	5	2	3	4	5	4
5	5	1	4	1	3	1	1	1	4	3
6	3	1	4	2	4	1	3	2	4	2
7	3	2	5	2	5	1	5	1	4	2
8	4	1	4	4	5	1	4	1	5	2
9	5	1	2	5	1	1	3	1	5	4
10	5	1	5	3	4	1	5	1	5	2
11	5	1	5	3	5	2	5	1	5	1
12	3	2	4	4	5	1	2	1	4	4
13	5	1	5	3	5	1	5	1	5	2
14	5	3	4	4	5	1	4	2	4	3
15	4	3	4	5	5	2	5	1	4	3
16	4	2	3	5	5	2	2	2	4	4
17	3	2	3	4	4	2	2	3	2	3
18	5	1	4	3	5	1	5	1	5	2
19	5	1	4	3	5	1	5	1	5	2
20	5	1	5	3	4	1	3	1	5	1

Figure II.2: BigER's raw SUS scores for user experience

Your ID (1-20)	Q1. I think that I	Q2. I found the t	Q3. I thought th	Q4. I think I wou	Q5. The tool's in	Q6. I found navi	Q7. I would ima	Q8. I found the t	Q9. I felt very co	Q10. I needed to
1	2	4	3	4	5	1	4	3	3	4
2	1	3	4	1	2	2	5	2	4	1
3	1	3	1	1	3	2	4	2	2	2
4	2	1	5	1	3	1	4	1	4	1
5	5	1	5	1	4	1	4	1	5	1
6	2	4	2	4	3	3	2	4	2	3
7	1	2	4	1	4	1	5	2	5	1
8	3	1	5	2	5	2	5	1	4	2
9	5	1	5	1	4	2	5	1	5	1
10	1	3	3	2	3	5	4	3	1	1
11	4	1	5	2	5	1	5	1	5	1
12	4	1	5	1	5	1	5	1	5	1
13	4	1	4	2	5	2	5	1	5	1
14	3	2	5	1	5	2	5	1	4	1
15	4	1	5	2	5	3	5	1	4	2
16	4	1	5	1	4	2	5	1	5	2
17	4	1	5	1	5	1	5	1	4	1
18	3	1	5	1	5	1	5	1	5	1
19	5	1	4	1	4	3	5	1	5	2
20	5	1	5	1	5	1	5	1	5	1

Figure II.3: ERDPlus' raw SUS scores for user experience

III

Raw Times for each tool

Participant	ERDPlus	Draw.io	BigER
P1	230	437	284
P2	259	280	247
P3	205	287	229
P4	274	203	277
P5	92	190	185
P6	216	202	136
P7	150	186	134
P8	215	248	176
P9	146	270	146
P10	211	174	206
P11	159	185	210
P12	223	209	163
P13	145	218	133
P14	234	217	212
P15	145	188	237
P16	252	191	149
P17	124	218	129
P18	200	195	148
P19	129	208	213
P20	172	200	167

Figure III.1: Task 1 times for each tool, in seconds

Participant	ERDPlus	Draw.io	BigER
P1	96	120	75
P2	94	40	68
P3	75	43	48
P4	112	59	41
P5	43	53	25
P6	61	27	53
P7	53	40	48
P8	74	53	23
P9	57	59	29
P10	40	33	33
P11	21	40	42
P12	62	39	34
P13	51	42	28
P14	38	28	28
P15	41	27	51
P16	43	33	40
P17	44	57	21
P18	25	33	29
P19	32	27	34
P20	38	25	23

Figure III.2: Task 2 times for each tool, in seconds

Participant	ERDPlus	Draw.io	BigER
P1	145	223	162
P2	201	188	160
P3	139	180	140
P4	140	146	125
P5	87	148	88
P6	128	146	115
P7	108	134	97
P8	179	146	115
P9	110	208	94
P10	116	93	137
P11	84	113	130
P12	175	110	118
P13	93	181	83
P14	167	134	166
P15	86	129	171
P16	151	110	93
P17	78	136	59
P18	114	109	105
P19	135	133	145
P20	91	127	103

Figure III.3: Task 3 times for each tool, in seconds