

---

---

# METHODS & TOOLS

---

---

Practical knowledge for the software developer, tester and project manager

ISSN 1661-402X

Fall 2013 (Volume 21 - number 3)

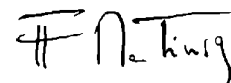
[www.methodsandtools.com](http://www.methodsandtools.com)

## Looking for Renaissance Software Development

As Agile is becoming a mainstream software development approach, I have seen on the Web more content about the importance of having cross-functional people in Scrum teams. Another denomination for this type of developer is to call them "T-shaped" people. The "T" symbolizes the deepness of the expertise in a particular topic and the broadness of knowledge in other areas of software engineering.

Large software development organizations have often chosen to have a tayloristic approach where people are separated according to their expertise: programmers, testers, DBA, business analysts. This separation in silos had also an implied "hierarchy" in the mind of developers: you are a tester or a business analyst only because you cannot code ;O) There is nothing wrong with being an expert and you cannot be expert in everything. This division, that was often increased by the fact that people were working in different offices, has created a "us versus them" mentality. I had the chances to work mainly in software development projects where I was able to perform different roles and I have found this situation beneficial. This is also because I have witnessed the loss of information when communication involved multiple people between the end-user and the software developer.

People like Leonardo da Vinci that were active in multiple fields from the art to engineering inspire the Renaissance reference in the title. We are not all geniuses like Leonardo da Vinci, even if some of us might think so, but we should be inspired by the broadness of his activities. Successful software applications are the combination of many elements that span from requirements to database performance and developers should know how they work together. This is why you find in our quarterly PDF issues articles that cover all the areas of software engineering. If you are a software renaissance developer, Methods & Tools is for you.



## Inside

Experiential Learning: What Does it Have to do with Agile? .....	page 3
Use the Debugger, Stupid! .....	page 9
What is the Mikado Method? .....	page 17
Zucchini – a Visual Testing Framework for iOS Applications .....	page 24
Portofino - A Java Web Application Framework .....	page 31
TerraER - an Academic Tool for ER Modeling .....	page 38
UnQLite - an Embedded NoSQL Database .....	page 42
Karma - a Javascript Test Runner .....	page 48
CodernityDB - a Fast Pure Python NoSQL Database .....	page 54

## **TerraER - an Academic Tool for ER Modeling**

Henrique Rocha, henrique.rocha [at] gmail.com, Ricardo Terra, rterrabh [at] gmail.com

TerraER is a free open-source learning tool designed to aid students in the creation of entity-relationship models. Our main goal is to provide students with a tool that reflects exactly the data modeling concepts learned in the classroom.

**Web Site:** [http://www.terraer.com.br/index\\_en.html](http://www.terraer.com.br/index_en.html)

**Version tested:** TerraER 2.02

**System requirements:** Java 1.5 or higher

**License & Pricing:** Open Source GNU Public License, Freeware.

**Support:** Documentation available at our website and issue tracker system at <https://github.com/rterrabh/TerraER>

### **Introduction**

Data modeling is one part of the database conceptual design process. The entity-relationship model (ER model) is a largely used conceptual model proposed by Chen [1]. It defines the entities and the relationships between these entities. The ER model is simple and easy to understand, making it intelligible to both database designers and end users [2, 3, 4].

Academic database courses still adopt ER models to teach conceptual design. Nevertheless, we noticed a lack of modeling tools for this purpose. In practice, most modeling tools support logical design, which is a detailed model the designer proceeds after the conceptual model is complete. In view of such circumstances, academics (students and professors) are forced to use logical design tools instead, such as DBDesigner ([www.fabforce.net/dbdesigner4](http://www.fabforce.net/dbdesigner4)), ERWin ([www.erwin.com](http://www.erwin.com)), etc.

The use of existing logical design tools - rather than conceptual design ones - has two major issues: (i) they may confuse students who are learning about conceptual modeling; and (ii) they were not developed for academic purposes. To address these shortcomings, we designed TerraER, a free open-source learning tool designed to aid students in the creation of ER models. Our main goal is to provide students with a tool that reflects exactly the data modeling concepts learned in the classroom.

### **Installation**

TerraER is distributed in a single JAR file publicly available for download at our web site [www.terraer.com.br/download\\_en.html](http://www.terraer.com.br/download_en.html). The JAR file is auto-contained, i.e., it can be placed in any folder, does not require the installation of additional libraries, and does not change operating system files (e.g., windows registry). In a nutshell, TerraER requires only a Java Runtime Environment (JRE) previously installed on the target computer ([www.java.com/en/download/index.jsp](http://www.java.com/en/download/index.jsp)).

### **Main Features**

TerraER 2.02 has the following relevant features:

- Free and open-source (GPL);
- Simple and small but complete application (around 5 MB);

- Cross-platform (only needs a JRE);
- Internationalization (English and Portuguese);
- Persist models to XML files (a cross-platform format);
- Clipboard transfer;
- Model printing; and
- Easy to learn and intuitive interface.

**User Interface & Feature description**

TerraER was initially designed for academic purposes, i.e., to help students and professors in the task of creating ER models. Therefore, we developed the graphical user interface to be practical, intelligible, and intuitive (i.e., easy to learn and use).

The tool is a free, open-source application under the GNU Public License. Thereupon, we encourage users to directly contribute to the TerraER project (<https://github.com/rterrabh/TerraER>). As an example, a B.Sc. student has contributed to the project by developing the internationalization for the English language.

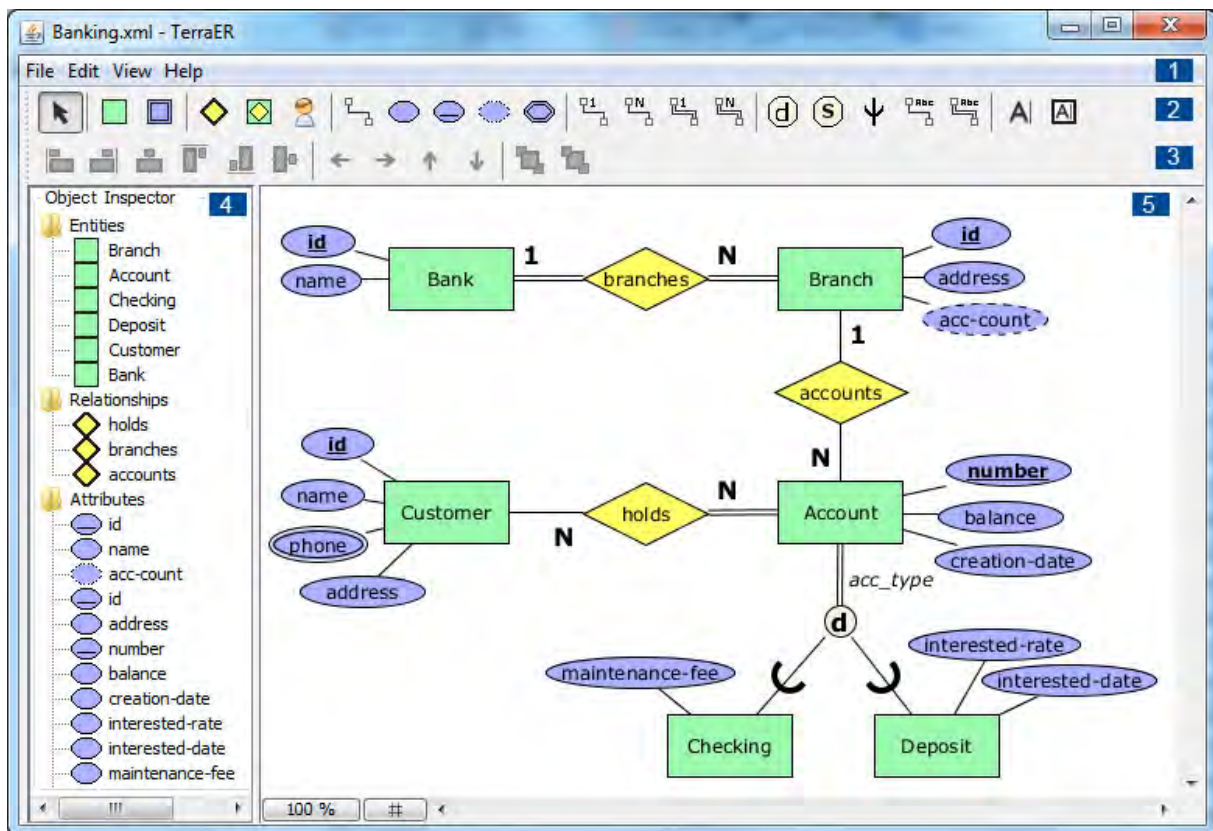


Figure 1 – Screenshot of TerraER 2.02 modeling a hypothetical Banking ER

The tool is developed in Java language, which makes TerraER a cross-platform application. In addition, new releases are always verified on Windows, Linux, and Mac OS.

Figure 1 illustrates the TerraER interface, which is organized into five main parts:

1. *Menu bar*: It provides the users with basic functionalities, such as saving, loading, and printing the models. More important, the models are saved in an XML-based format, which directly contributes to the cross-platform feature. In practice, models saved using an operating system can be loaded on a different one without any issue;
2. *Objects toolbar*: It provides commands to create ER model elements - such as entities, relationships, attributes, etc. - on *Chen's* notation, as adopted by Elmasri and Navathe [3];
3. *Position toolbar*: It provides users with means to manipulate elements' position - such as alignment, overlapping, etc. - in order to support an elegant model design;
4. *Object inspector*: It lists the elements of the current ER model and allows user to select, remove, or edit them. In practice, this feature provides a quick and precise way to handle model's elements;
5. *Drawing area*: It shows the graphical view of the ER model under creation. The user may add and remove elements to the model. There is a *zoom* feature, which can be very useful when dealing with large models. Also, there is a *grid* feature that displays a grid to help users to position elements.

## **Evaluation Experiment**

TerraER was initially designed as a teaching aid tool for academic database courses. In order to evaluate the applicability of our tool, we conducted a study in a Brazilian university to obtain the feedback from undergraduate students, which represent our target public.

In the study, we defined three assignments and divided the database course class in 10 groups. In the first assignment, five groups were required to use DBDesigner (a popular database design system) and the other five groups to use TerraER. In the second assignment, we swapped the groups. In the third and last assignment, each group could opt for which tool they prefer. In the last day of the class, the students filled an evaluation form.

The results clearly indicate a preference for TerraER as the model design tool. In fact, nine out of ten groups preferred TerraER to DBDesigner. According to the answers of the evaluation forms, the students argue that TerraER is easier to use and understand because it reflects the conceptual design learned inside the classroom.

## **Supporting TerraER**

We welcome any help from the open-source community to support TerraER. For those not comfortable with their programming skills, it is possible to contribute with suggestions, donations, and even helping in the creation of a better documentation. TerraER source-code is available at <https://github.com/rterrabh/TerraER>, for those willing to play a more active roll.

## **Conclusion**

Academic courses still adopt ER model for database modeling. However, there is a severe lack of tools designed for this particularly purpose. In view of such circumstances, academics (professors and students) do not have other option than to rely on tools that do not follow the concepts learned inside the class. Even though these are usually popular tools, they do not draw conceptual models but logical models instead. In practice, it may negatively impact the learning curve of the students.

To address this shortcoming, we developed TerraER with the specific purpose to be adopted in academic courses. Thereupon, we made the tool simple, easy to use, and complete w.r.t. ER modeling using *Chen's* notation. More important, the study we conducted has shown students' preference for TerraER because it reflects the concepts learned in the classroom.

Last but not least, TerraER is a small, multi-platform, free, and open-source software system, which make the tool the proper choice for conceptual data modeling. As far as we can guarantee, TerraER is currently employed in databases courses in ten Brazilian universities.

## **References**

- [1] P. P. Chen. *The entity-relationship model – towards a unified view of data*. ACM Transactions on Database System, pages 9–36, 1976.
- [2] S. Bagui and R. Earp. *Database Design Using Entity-Relationship Diagrams*. CRC Press LLC, 1964.
- [3] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 6th edition, 2010.
- [4] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 6th edition, 2010.