

An Evaluation of ERwin

by David C. Hay

ERwin is arguably the most popular CASE tool on the market today. In the view of your author, this is unfortunate, because it has shortcomings that prevent users from producing really good models. Its widespread use bodes ill for the development of healthy modeling habits in the industry. To summarize, three of ERwin's shortcomings are particularly significant:

- Lack of user control over the aesthetics of diagrams.
- Failure to distinguish between a model and the various drawings representing that model.
- Failure to separate physical database designs from conceptual models.

In addition, there are unnecessary and wrong constraints on particular model configurations. The problems with the tool are itemized in detail below, in terms of graphic issues, modeling issues, and model management issues.

Graphic Issues

1. Users should have more control over the aesthetics of a diagram.

Lack of control over diagram aesthetics is one of the three most important shortcomings of ERwin.

- It should be possible for the user to control the size and shape of individual entity boxes. This is desirable both to eliminate bends in relationship lines, and to accommodate the number of relationship lines that may be attached to entities. [*enhancement*]
- It should be possible to attach a recursive loop (relating one occurrence of an entity to another occurrence of an entity) to any side of an entity. In Erwin, the "many" end of the relationship can only be placed at the right side, and the "one" end can only be either on the right or bottom side. [*enhancement*]
- It should be possible to draw diagonal lines. [*enhancement*]

For further information on this point, see the Auerbach article, "Making Data Models Readable"¹.

2. ERwin represents subtypes outside the supertype boxes, with specialized lines connecting them. It should be possible to represent sub-types as boxes within

¹ Hay, D., "Making Data Models Readable", *Information Systems Management*, 15(1), Winter, 1998, pp21-33.

The article is also available at <http://www.essentialstrategies.com/publications/makingrd.htm>.

super-type boxes, as described by Richard Barker in his data modeling book² and by James Martin in his information engineering books.³ Since some believe that the external approach is better, it would be best if that were an option. [enhancement]

3. The symbols on each end of a relationship should be independent of each other. That is, an end is mandatory or not, regardless of what is at the other end. In ERwin, a circle across the line shows that the end is optional if either this end or the other end is singular. There is no circle if the relationship is many-to-many. [modeling error – important]
4. It would be valuable to be able to select a relationship line, and then select "Edit/Redraw Diagram", to have just that line be straightened. If nothing is selected, "Redraw Diagram" would re-arrange the entire diagram. ERwin's reformat function doesn't work this way. It can only reformat the entire diagram. [enhancement]
5. When you change the color of an entity on a drawing (subject area to you), that color should only apply to that drawing. The way ERwin is now, any other subject area that contains the entity assumes that the entity is the color set for the first subject area. This is problematic in attempting to prepare a series of drawings. The first has one or two entities; the second adds a couple, the third adds a few more, and so forth. Because of the way ERwin works, one has to define a separate subject area for each drawing. In each drawing, the desire is to shade the entities that were added in that drawing. The problem is that the entities that were added in the previous drawing are still shaded when creating this new one. [enhancement]
6. Zooming in or out seems to cause straightened lines to become zig-zagged again. This is important for presentation purposes. [graphics error - important]

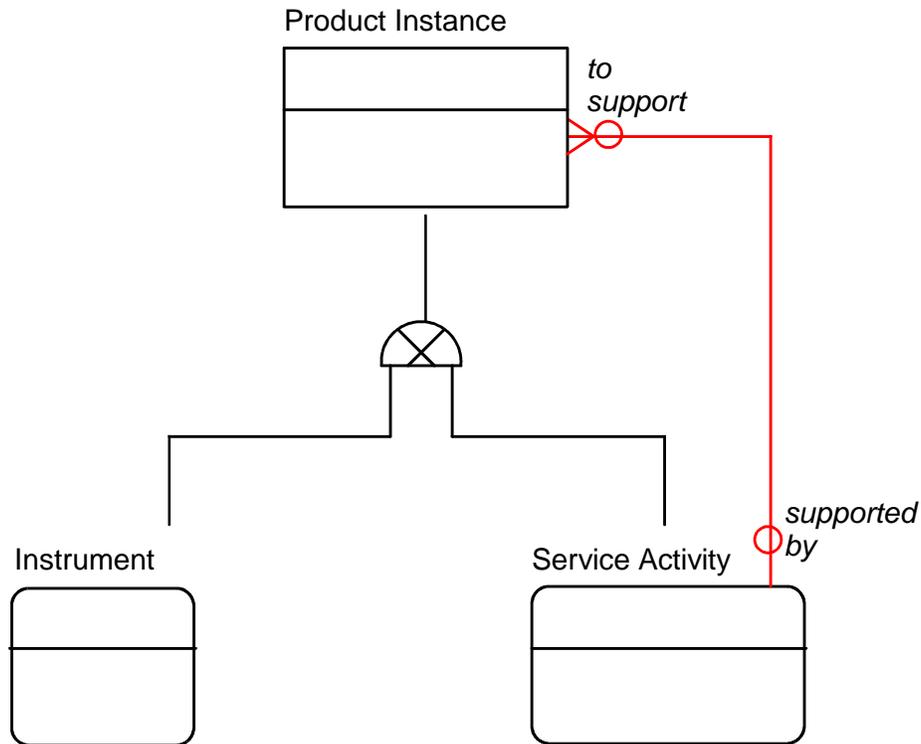
² Barker, Richard, *CASE Method: Entity Relationship Modelling*, Addison-Wesley Publishing Company, Wokingham, England, 1989.

³ For example, Martin, James and Carma McClure, *Diagramming Techniques for Analysts and Programmers*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

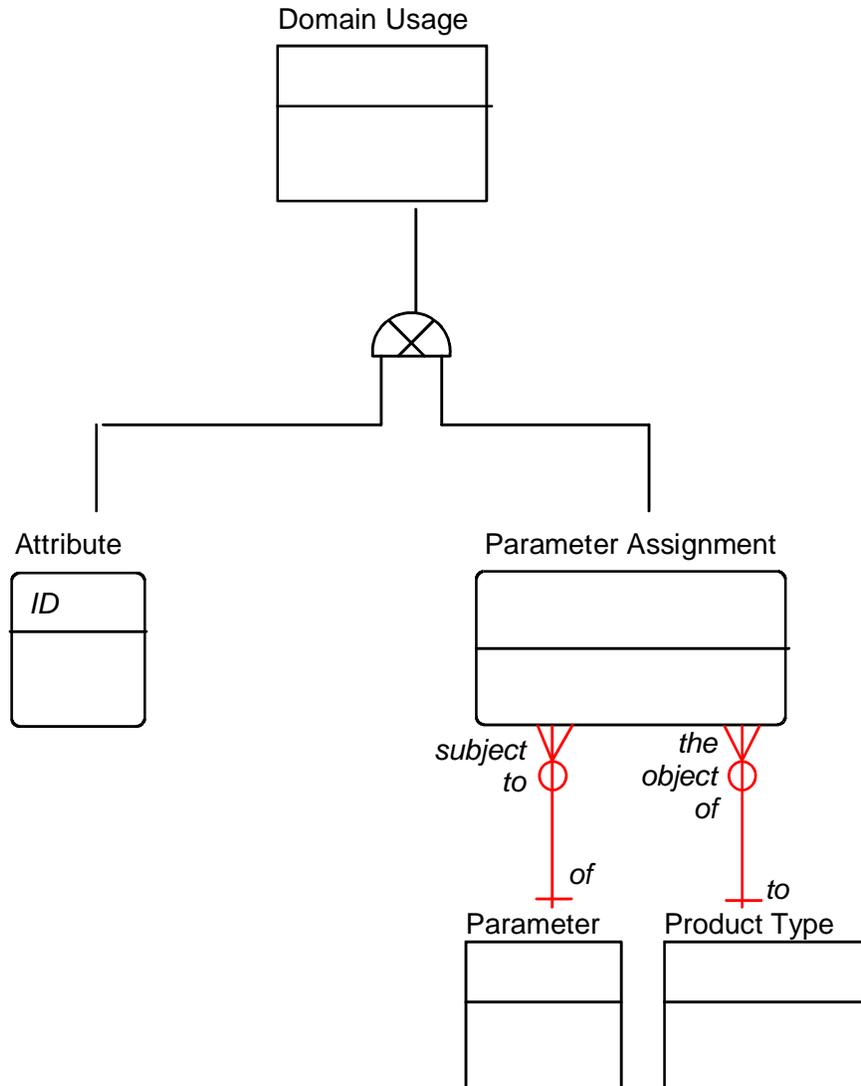
Modeling Issues

1. The following rules are imposed on models by ERwin that should not be:
[*modeling error – important*]

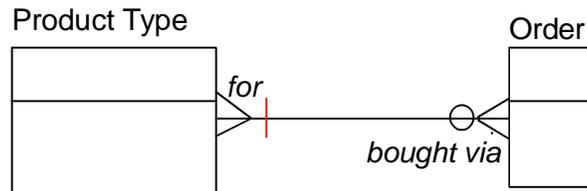
- A super-type may not be a child to a sub-type in a relationship. Figure 1 shows that each SERVICE ACTIVITY may be *to support* one or more PRODUCT INSTANCES, and each PRODUCT INSTANCE may be *supported by* one and only one SERVICE activity. In ERwin, this relationship is not permitted.



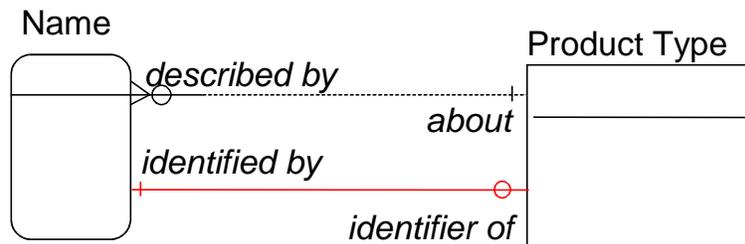
- A sub-type may not be a child of any other entity. Figure 2 shows that each PARAMETER ASSIGNMENT (kind of DOMAIN USAGE) must be *of* a PARAMETER and *to* a PRODUCT. This is not permitted in ERwin.



- One side of a many-to-many relationship may not be mandatory. (It is appropriate to assert that both sides may not be mandatory, although it is not really for the CASE tool to enforce this, either.) Figure 3 shows a case where each PRODUCT TYPE may be *bought via* one or more ORDERS, but each ORDER **must be for** one or more PRODUCT TYPES. ERwin cannot produce this model.



- It is not permitted to have two relationships between two entities (directly or indirectly) going in opposite directions. Figure 4, below, shows the case where each PRODUCT TYPE may be *described by* one or more NAMES, but each PRODUCT TYPE may be *identified by* a single NAME. This cannot be represented in ERwin.



In each case, we have a real logical requirement that cannot be modeled with these rules in place.

2. When a relationship in the logical model is converted to a foreign key in the physical model, the name of each component of the foreign key should be constructed by concatenating the name of the corresponding primary key component with the target table name. Better yet would be to concatenate the primary key component with the name of the entity from which the table was derived. Currently the primary key column name is simply replicated in the table with the foreign key, which is problematic if it already has a column by that name. [enhancement]
3. The tool should support the concept of an exclusive relationship. Usually represented by an arc drawn across two relationship lines, this asserts that <entity A> { must be|may be } { <relationship 1> <entity B> or <relationship 2> <entity C> }. For example, “Each LINE ITEM must be *for* one and only one PRODUCT TYPE, or *for* one and only one SERVICE.” This is a business rule type that occurs frequently and which could be represented by this simple graphic extension.

This is translated into the physical model as two foreign keys, with trigger logic that enforces use of only one. [*enhancement*]

4. When a logical supertype/subtype relationship exists, the physical model shows this as a supertype/subtype relationship, with a many to one relationship to each of the subtypes from the supertype. This is not correct. Each subtype should be a zero or one relationship to the supertype. Allan also called to mind that the supertype/subtype relationship should not exist in a physical model. But ERwin still shows it.

Actually, what you need is an “arc” (see previous point) showing that each occurrence of the supertype is related to exactly one occurrence of *either* subtype 1, subtype 2, etc. [*modeling error – important*]

Model Management Issues

1. The underlying model should be different from drawings made from that model. That is, there should be records of the set of entities and relationships, etc. that constitute the model in the database. Separately from this, it should be possible to create drawings that represent selected objects from the model.

In a modeling session, for example, we begin with a picture showing one entity only. We discuss that until everyone is comfortable with it, and then show a picture with two entities (including the one we just saw) and a relationship between them. Then we move on to four entities, six entities, and so forth until we have maybe fifteen entities on the diagram. Then we move on to another subject area and repeat the process.

Failure to distinguish between an underlying model and drawings representing portions of that model is one of the three most important shortcomings of ERwin.

This is different from the requirement to be able to manage the model and access to it. It should be possible to group model objects into “application areas” (sort of like your “subject areas” but different in important ways), where each person is given read/write or read-only access to one or more application areas. Each model object is “owned” by one and only one application area, although it can be “shared” with others. That is, an application area can make use of (and show on diagrams) an entity owned by another application area, but if changes to its definition are required, these must be taken up with someone who has read/write access to the owning application.

With ERwin, we have to use “subject areas” to perform both functions. This means we have to have a separate subject area for each diagram. There is no way, short of naming conventions, to recognize that 10 of the subject areas constitute one application area and another 15 subject areas constitute another. [*functional error – very important*]

2. Related to this point, it should be possible to “save as” a diagram under a new name. In ERwin, doing this duplicates all the entities in the model. What should happen is that the underlying entities, etc. are unaffected. You simply have a new

representation of the same objects. You can then remove elements from this drawing, leaving the others as they were. This is the most convenient way to make up the series of pictures described above. [*functional error – very important*]

3. The logical model and the physical model should be completely separate sets of objects. While the latest version of ERwin purports to support both logical models and physical models, the entities in a logical model are in fact the same objects as the tables in its corresponding physical model.

Lack of separation between the logical and physical models is one of the three most important shortcomings of ERwin.

This means, among other things, if you delete a table, you have in fact deleted the entity as well. If you change the structure of a table, you have changed the structure of the corresponding entity.

It also means that foreign keys exist in the logical model, even though they have no place there. What in the physical model is a foreign key is represented by the relationship in the logical model.

We have a group developing logical e/r models. When complete, these will be converted to first-cut physical models, which will then be passed to the database designers. The database designers must be free to denormalize the structure as required -- splitting entities into multiple tables, moving columns from one table to another, etc. We will have procedures to insure that physical departures do not conflict with the meaning of the logical model, but otherwise the physical designers should have a lot of latitude. None of the changes to the physical database design should affect the logical model, however.

This cannot be done with ERwin.

Moreover, before long it will be necessary to have multiple physical models for a given logical model. We are building a data warehouse and are in the process of mapping various legacy systems to the logical model. This cannot be done with ERwin.

This is unacceptable. What is needed is separate logical and physical objects in the ERwin repository, with "wizards" or some such utilities for converting entities to tables. Once the tables have been created, they would have a separate existence and could be manipulated as necessary.

With this approach, it would then not be necessary to give the database designers access to the logical models.

In addition, the "wizard" could do the correct foreign key renaming and provide the logic for converting sub-type/super-type combinations to tables, as described above. [*functional error – very important*]

4. If a model is checked out of Model Mart and entities are deleted from it, when the model is merged back in, the deletions are not recorded. It is necessary to go into model mart and delete the same entities again.

This is not how it should work. [*functional error – very important*]

5. When our model became very large, we could not use Model Mart. The response time for opening and saving diagrams became prohibitive. For this reason, we have not been using Model Mart since Christmas. [*functional error – very important*]