Literature Review: Researching graphical design tools for temporal models

Richard Taylor
The University of Cape Town
Cape Town, South Africa
tylric007@myuct.ac.za

ABSTRACT

When designing conceptual data models', modelers have a wide range of choices of tools to assist them. However, there remains a distinct lack of tools for modelers to integrate temporal elements into their models, with full reasoning support. This paper will be an exploration into the existing technologies for conceptual data modelling, temporal reasoning and graphical tools to get an understanding of what state-of-the-art is in order to assist with the building of a novel tool that will assist users in their modeling, specifically in the conceptual modelling language TREND, which is an extension of the extended-entity diagram.

KEYWORDS

Conceptual modelling, temporal data, graphical tooling.

1 INTRODUCTION

The significance of temporal data has grown with the advent of cost-effective long-term data storage, particularly noticeable in the surge of business process modelling and analytics. More specifically, temporal data is essential for a wide range of applications, including historical data tracking, trend analysis, forecasting, and compliance with legal and regulatory requirement or in other words temporal data allows for better decision making [19]. Since the mid-1990s we have seen the introduction new database technologies that facilitate temporal functionalities, and this shift necessitates the integration of temporal capabilities into the corresponding models. Even since before the turn of the century, most databases applications have been temporal [11].

When we create our conceptual data models, we are trying to create a simplified map of our information domain in order to aid understanding of our problems for experts and novices alike. Due to the scope set for this project, we will consider entity-relationship (ER) diagrams as our conceptual model of choice, thus look only ER/EER temporal extensions.

In ER/EER we are not interested in how the data is stored yet, but we are interested in what needs to be represented and how the relationships between entities in that data are represented. This is unlike , for instance, Object-Role Modelling, which focuses on the "how", which makes it useful for formulating, transforming or evolving designs [10]). It will be assumed that the reader has a basic understanding of the language of ER/EER diagrams, and extended entity-relationship diagrams (EER) – which in summary are essentially an augmented version of an ER diagram that includes the functionality of inheritance an composition, [14] provides great further reading. It is also worth mentioning that there are one-to-one mappings between most specific ER/EER notations (such as Chen's original proposed notation or the crow's feet notation), the result being that they are isomorphic and will not be discerned from one-another. It will be discussed how it could be useful in a fully realized graphical tool to have notation as a simple toggle option allowing users to choose their preferred notation, as there is still not a wide consensus on which (notation) is the best [14].

This review stands as a basis for a novel tool, thus must thoroughly investigate the current knowledge that exists around temporal modelling, and graphical modelling in general and provide an understanding on what the current state-of-the art is for said modelling in section 3. In doing this however, we must also come to an agreement on what temporal modelling language to design tooling for in section 2.

2 TEMPORAL DATA

Elmasri and Navathe (REF) provide an introduction to temporal database concepts, which describes the notion of time in this context as an ordered sequence of points (often referred to as chronon) which are arbitrarily defined by the user depending on what granularity is needed. For example, if 1 second was chosen as the time chronon, two events occurring within the same second would be considered as simultaneous events, even if the events are not in fact simultaneous. We are also given a distinction between the concepts of valid time and transactional time. Valid time is the actual time an event or fact occurs, while transactional time is the time the time a fact is recorded as valid in the database [7]. In practice most often both transactional time and valid time are used in together.

3 TREND

Temporal information Representation in Entity-Relationship Diagrams (TREND) is the temporal conceptual data modelling language (TCDML) we are designing tooling for. What makes TREND unique is its extensive testing into the understandability and usability by modellers, novice and expert alike. This was done through iterating TREND, as to make it more usable, though feedback from over 1000 participants from a series of 11 experimentations. This makes it the perfect language on which to base potential novel graphical tools.

We have also included additional metrics set out in [1] to evaluate TREND in the form of the following requirements:

- Orthogonality, which is that it's important that temporal features be defined distinctly and independently across classes, relationships, and attributes, and that The level of

temporal support should be tailored by the designer based on the specific needs of the application, this will prove true when the temporal extensions TREND implements are introduced.
- Upward compatibility, which is the ability to maintain the original non-temporal meanings of traditional (legacy) conceptual schemas even when they are incorporated into temporal schemas. Since TREND is an extension of the ER/EER notation, it maintains the meaning of non-temporal ER/EER diagrams.

Non-graphical attempts at temporal modelling have been made, but as we learn in [2], graphical notations are clearly preferred over textural notations from the modellers' perspective, so languages without graphical notations are ruled out as in general they are not a usable as their graphical languages. Keet and Berman tell us that natural language is preferred by some for more complex temporal constraints, which makes a case for having both graphical and natural language features in a potential tool [15].

3.1 Modelling with TREND

Trend implements its temporal extensions through the core constraints of dynamic extension and dynamic evolution [13]. These constraints outline the primary ways in which temporal entities in a model can change over time. For dynamic extension (EXT/ext), the extended entity is still an instance of of the original entity. For example, an *Employee* may get promoted to *Manager*, but still stays an *Employee*. On the other hand, for dynamic evolution (CHG/chg), the extended entity is no longer the same instance and the entity it has extended from. For example, in a law firm, when an *Article Clerk* becomes a *lawyer*, it stops being an *Article Clerk* [13]. Other important notation distinctions must be made for TREND:

- **Icons:** Temporal entities as described above are labeled with a clock icon, while atemporal entities (legacy ER/EER entities) are labelled with a clock.
- **Mandatory or Optional**: Mandatory temporal constraints (that must have in the past or must occur in the future) are denoted by a solid line and optional constraints (that may happen in the future or could've happened in the past) are denoted by a dashed line.
- **Past or future**: If a temporal constraint is on a past condition, lowercase letters are used (i.e. chg, ext) and capital letters are used (i.e CHG, EXT) on future constraints.
- **Quantitative**: If a constraint has a quantitative value attached to it (for example, an *employee* must have been an *assistant-manager* for 2 years before coming a *manager*), then that value is displayed in time chronon, next to the constraint type.

These elements on notation can be summarized in the following example.

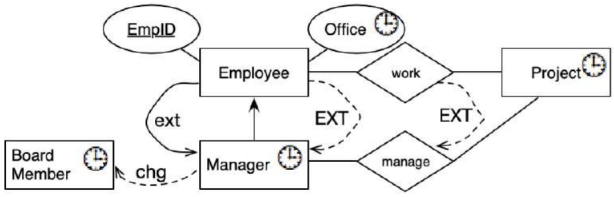

Figure 1: EER diagram extended by TREND. Taken from [13].

Objects that can change with time (such as an employee's office) are labelled as temporal elements with the clock icon. Employees can potentially, with time, become managers, and to be a manager you must have been an employee [1]. This is represented by the past mandatory and future optional dynamic extensions, with the relationships also extended. A manager can also in time optionally become a Board Member, which is not an instance of a manager, thus is represented by the future optional dynamic evolution. Note that no graphical tool exists for TREND specifically, and this example was done without the help of a constraint checker.

4 GRAPHICAL MODELLING TOOLING

From pen and paper to specialised software, we have the ability to model systems in general. ER/EER diagrams are one of the most popular among modellers [5], as they aid in the understanding for complex systems. Modelers in practice are represented by varying expert levels, from novices to experts who have been modeling for over 10 years [5]. Using a survey, Davies et al. originally found the most popular use cases were Database design and management first, Business process documentation second and then Improvement of internal business processes third [5], however in a more recent survey [8], there were some minor changes to what were the most popular use cases for conceptual modelling:

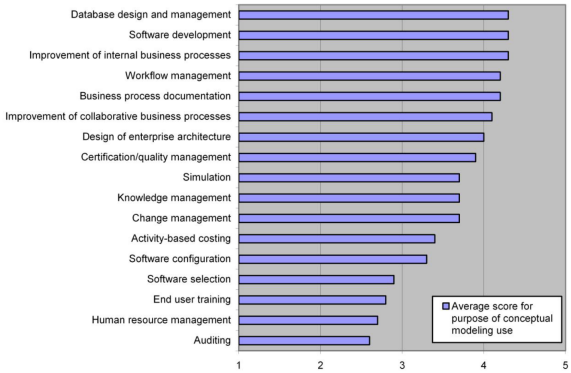

Figure 2: Average Score for Purpose of Conceptual Modeling Use [2]. Taken from [8].

[1]Note: A manager is still an instance of an employee
[2]This was surveyed as a five-point question, where 5 meant very frequent use, and 1 meant not used at all.

4.1 Extension of a preexisting tool.

In research for this topic, many modeling tools were discovered which are rich in features and have effective graphical user interfaces, thus rather than building our own tool from scratch, given the development time of this project, it would be better to augment an already existing - open source - tool to create our end result. This will further be elaborated on in the Discussion section. No tool was found that possessed temporal reasoning of any kind, which exposed the gap in tools where a new tool is needed.

5 REASONING

As per the scope of this project, reasoning on our temporal models is an essential feature. This will take the form of constraint checking over temporal models to alert the user if they have made any mistakes - in terms of model structure (for a simple example, two attributes can not be connected, thus a connection between two attributes should raise an error to the user). This reasoning will aid modelers in their learning and usability of TREND diagrams, which is hypothesized to increase productivity and lower the barrier to entry for temporal modelling. Thus, in the final section discussing tools, not only will the graphical tooling be taken into account, but also any reasoning support the tool might have. Any reasoning feature could be augmented from reasoning on TREND models.

6 DISCUSSION

Comparison of existing graphical modelling tools

We will do an in-depth analysis into different approaches taken for other tools. The successful approaches that pertain to our application domain will be a base of inspiration for our graphical TREND tool. Note that this review was limited to open source papers which had corresponding studies or relevant papers published which restricted the majority of tools that have been built.

6.1 ERD Tool

The Entity-Relationship Diagram Tool was, as the name suggests used to graphically model entity-relationship diagrams (ERD) [17]. It focuses on a simplified and streamlined user-interface and fulfilling all the necessary requirements of ERD, such as cardinality, participation and recursive relationships [17]. These requirements do not align with what we are looking for in a potential tool, but in analysing the underlying framework this tool uses, we can apply the same frameworks to our application domain.

The Graphical Editing Framework (GEF) is useful in responsibilities such as configuring and creation and saving of models and creating a palette for modellers to use. This palette consisted of all the necessary tools for model building, such as attributes and entities, and the relationship between these models [17].

GEF implemented the Model-View-Controller design pattern, where the canvas the the user sees is connected to the underlying model data through the use of a controller, so if the user changes a model or canvas, the controller will update the other accordingly. In this design, there is a separate controller for each type of object on-screen (entity, attribute and relationship). GEF also supports notifications with the use of the observer design pattern. These design decisions will be noted when designing a similar tool. This tool does ERD validation, but this validation is no extensive. If a user makes an obvious mistake, such as connecting two attributes, they will be notified, however this error checking does not apply to more sophisticated ERD rules. Additionally, this product misses our on much of the usability testing and features users would expect, such as snap-to geometry [17].

6.2 Crowd

Crowd is a research tool that allows for temporal ER modelling, and is assisted by automated reasoning. It implements Client-server architecture, where each client supports Model-view-controller architecture and queries the server for reasoning support. The temporal ER graphical reasoning support in the form of checking for class consistency, checking for relationship consistency, discovering implied class and cardinally constraints. This is more extensive than the ERD tool, yet does not contain any of the temporal constraint we would want to implement. In this case we could leverage a similar technique to apply our temporal constraints. The older versions of Crowd makes use of Owllink to interface with OWL (Web Ontology Language) reasoners to achieve its graphical reasoning support, using a Knowledge Base (KB), which is collection of axioms and facts we want enforced [16]. OWL enables the precise definition of terms in vocabularies and their relationships. This process of defining terms and their connections is known as creating an ontology [18]. As it stands, no version of Crowd supports the constraint checking for temporal extensions to ER [3], and instead the most recent version of crowd [4] supports EER, UML (unified modelling language) and ORM 2 along with connection to the RACER and Konlcude reasoners [9]. More rules could be added in to create our TREND tool, in using this version of crowd, support for temporal graphical objects would also need to be added in.

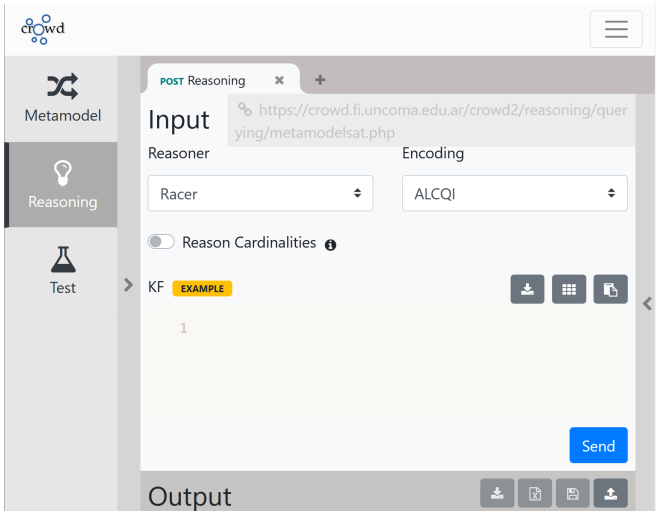

Figure 3: Crowd's Reasoning feature. Screenshot taken from http://crowd.fi.uncoma.edu.ar:3335/api.

[3] And only crowd 1.0 has support for building temporal ER extensions.
[4] Which is crowd 2.0

Crowd is based of the principle that interoperability between different conceptual modelling languages, this gives modellers the freedom to choose their preference of modelling language [3]. This can inspire a feature in our own tool. Although this tool is planned to only be available for TREND applications so far, a feature could be introduced where modellers could have the ability to swap between different ER/EER notations as per their preference. This is possible due to the one-to-one mappings between entities in different notations.

The newest version of crowd also has and updated user-interface, with much needed features added to make the tool work-flow more similar to other advanced modeling tools. This includes features such as draggable objects (with snap-to geometry), where the user can create relationships and specify cardinalities between relationships. It also possess a layout feature which automatically sets the layout of objects in your model (including but not limited to grid layout, circle layout, and concentric layout) and zoom control. The disadvantages to crowd include a faulty reasoner, and as said a lack of temporal features in the newer versions, so the palette lacks any temporal objects. Crowd also does not support keyboard shortcuts.

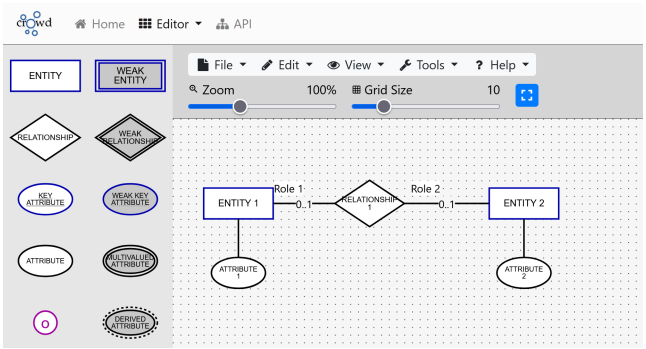

Figure 4: Crowd's user interface, using EER. Screenshot taken from http://crowd.fi.uncoma.edu.ar:3335/editor/eer.

6.3 Draw.io

Draw.io (formally diagrams.net) is a web-based tool (also with a desktop version) for general modeling for diagrams, such as flowcharts, ER and UML diagrams, and network diagrams. It too has open source code that would allow it to be extended into the tool we want. Out of existing modeling tools, Draw.io has most extensive, polished and feature rich user-interface. It included detailed style customization for graphs (text, colour and arrangement), premade templates of modellers to use and an extensive palette of objects [5]. The tool also fully utilises shortcuts [6] and mouse drag-to-select (which groups objects that can now be moved simultaneously), and allows for the insertion of custom made objects (and images). Also included are an integrated spellchecker, extensive language support, automatic backup, find-and-replace, plugin support, zoom, and layers for occluding objects. This extensive feature list makes Draw.io the best tool to extend for our own tool - as a graphical tool it is state-of-the-art.

[5]Although upon selecting "Entity relationship diagrams", modelers are greeted with a template that is not an ER diagram, and that more resembles a UML class diagram.
[6]It has custom shortcuts found at https://shorturl.at/fhzGP

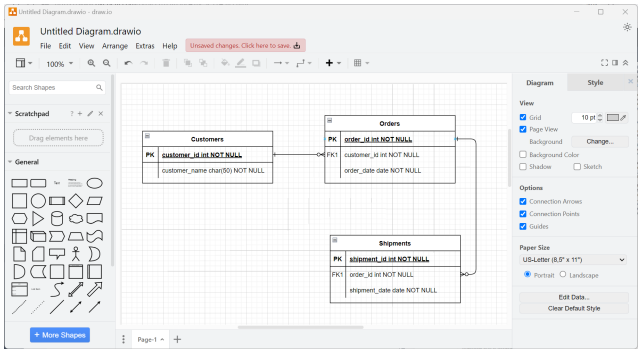

Figure 5: Draw.io's user interface, using ER. Screenshot taken from draw.io native: https://github.com/jgraph/drawio-desktop.

This makes it attractive as a base to extend for our novel TREND tool. In practice, our tool would have the user limited in the types of objects they could use from the palette, and temporal objects must be added. The main disadvantage of Draw.io is its lack of reasoning support of any kind. OntoPanel [4] provides an understanding into how a potential reasoning extension to draw.io could be implemented. OntoPanel is a plugin for draw.io which is designed to assist domain experts in creating and mapping visual ontologies for materials testing (with a focus on ensuring data is compliant with FAIR principles (Findable, Accessible, Interoperable, and Reusable)). It includes pipeline tools for importing and reusing existing ontologies, converting diagrams to OWL, verifying diagrams against OWL rules, and mapping data for interoperability [4].

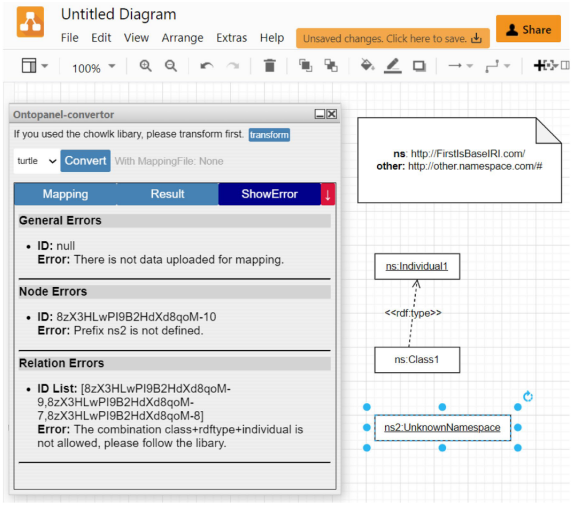

Figure 6: Ontopanel's reasoner working in Draw.io. Taken from [4]

The front-end is built in JavaScript and interfaces with Draw.io's mxGraph library, and interacts with the back-end through REST-APIs. The back-end is built using the Django REST framework, it contains a module that converts the models into a form that the OWL reasoners can perform constraint checking over. Any errors are then send back to the user [4]. This gives us insight into how we could implement TREND temporal reasoning over our novel tool

using Draw.io as a base. This leaves us in a position to begin work on our tool, and depending on scope, consider more ambitious features, such as controlled natural language conversions into models [12] or the conversion of hand-drawn models to be converted into digital temporal models [6].

CONCLUSIONS

To conclude, there is a need for a novel graphical tool to assist in the creation of temporal models. Despite the existence of various tools for general conceptual modeling, none fully meet the specialized needs of temporal modeling with intuitive graphical interfaces and comprehensive reasoning support. Given time and project scope, this tool would be best fitted as an extension to an already existing tool, and augmented to support our TREND temporal needs. In the discussion section of this review, three tool were explored for their potential in this regard. The ERD tool and Crowd both posseed reasoning support for non-temporal constraints, however, it was decided to extend Draw.io for its state-of-the-art user-interface. We also explored OntoPanel, a plugin for Draw.io, which outlines an implementation of reasoner support which informs us in the building of our own novel tool.

REFERENCES

[1] Artale, A., Parent, C., and Spaccapietra, S. Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence 50*, 1 (2007), 5–38.

[2] Berman, S., Keet, C. M., and Shunmugam, T. The temporal conceptual data modelling language trend.

[3] Braun, G., Gimenez, C., Fillottrani, P. R., and Cecchi, L. Towards conceptual modelling interoperability in a web tool for ontology engineering. In *III Simposio Argentino de Ontologías y sus Aplicaciones (SAOA)-JAIIO 46 (Córdoba, 2017)*. (2017).

[4] Chen, Y. Ontopanel: a diagrams. net plugin for graphical semantic modelling.

[5] Davies, I., Green, P., Rosemann, M., Indulska, M., and Gallo, S. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering 58*, 3 (2006), 358–380.

[6] El-abbassy, A. Tablet pc applications for education: database design using hand-drawn erd. *Journal of the ACS 3* (2009).

[7] Elmasri, R., and Navathe, S. B. Fundamentals of database systems seventh edition.

[8] Fettke, P. How conceptual modeling is used. *Communications of the Association for Information Systems 25*, 1 (2009), 43.

[9] Gimenez, C., Braun, G., Cecchi, L., and Fillottrani, P. R. crowd: A tool for conceptual modelling assisted by automated reasoning. In *II Simposio Argentino de Ontologías y sus Aplicaciones (SAOA 2016)-JAIIO 45 (Tres de Febrero, 2016)* (2016).

[10] Halpin, T. Object-role modeling: an overview. *white paper,(online at www. orm. net)* (1998).

[11] Jensen, C., and Snodgrass, R. Temporal data management. *IEEE Transactions on Knowledge and Data Engineering 11*, 1 (1999), 36–44.

[12] Keet, C. M. Natural language template selection for temporal constraints.

[13] Keet, C. M. *A Brief Introduction to Temporal Modelling with TREND.* Submitted to an international journal, 2021.

[14] Keet, C. M. Conceptual data models. In *The What and How of Modelling Information and Knowledge: From Mind Maps to Ontologies*. Springer, 2023, pp. 49–79.

[15] Keet, C. M., and Berman, S. Determining the preferred representation of temporal constraints in conceptual models. In *Conceptual Modeling: 36th International Conference, ER 2017, Valencia, Spain, November 6–9, 2017, Proceedings 36* (2017), Springer, pp. 437–450.

[16] Liebig, T., Luther, M., Noppens, O., Rodriguez, M., Calvanese, D., Wessel, M., Horridge, M., Bechhofer, S., Tsarkov, D., and Sirin, E. Owllink: Dig for owl 2. In *OWLED* (2008), vol. 48.

[17] McFadyen, R. An erd tool. *Submitted to an international journal* (2015).

[18] McGuinness, D. L., Van Harmelen, F., et al. Owl web ontology language overview. *W3C recommendation 10*, 10 (2004), 2004.

[19] Zimdars, A., Chickering, D. M., and Meek, C. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320* (2013).