Facilitating the implementation of the TREND Temporal Conceptual Data Modelling Language

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

Stephan Maree
University of Cape Town
Cape Town, Western Cape, South Africa
mrxste013@myuct.ac.za

ABSTRACT

This software project aims to develop a tool that allows the user to design and implement a model based on the TREND Temporal Conceptual Data Modelling Language. There are currently no tools that are designed for Temporal Conceptual Data Modelling, and there exists no software that generates temporal database schemas based on a model. The project aims to address these problems by building a modelling tool, as well as designing and implementing an algorithm to generate a database schema based on the user's model. This is expected to lower the barrier of entry for temporal modelling, as well as increase productivity for experienced modellers.

1 INTRODUCTION

Temporal Databases are gaining increased attention in recent years, in both industry and academia [5]. Temporal data refers to any data that includes one or more time period for which that data is considered to be valid [14]. This contrasts atemporal data, in which the data stored is assumed to be currently valid. There is a common distinction between valid time and transaction time, the former referring to the period in which the fact is considered true in the real world and the latter refers to the time in which the fact is reflected in the database.

The added expressiveness that comes with these features have shown promise in fields like video surveillance [21] and clinical data analysis [3].

The standard temporal conceptual modelling languages (TCDML) for atemporal databases is the Entity Relationship (ER) Diagram (along with The unified modelling language - UML), however this does not allow for effective modelling for temporal data. The structure of a temporal database can me modelled using a TCDML, with many of these being extensions of the ER Diagram [1, 2, 10, 22]. This gives a high level overview of the information logic underlying the database in a graphical and accessible way.

TREND is a TCDML that emphasises usability and accessibility [4, 12]. 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 Figure 1.

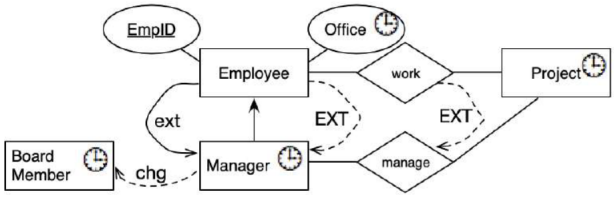

Figure 1: ER diagram extended by TREND. Taken from Figure 2 of [13].

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 modelling tool exists for TREND specifically, and this example was done manually - without the help of graph validator.

2 PROBLEM STATEMENT AND AIMS

Temporal databases have a high barrier to entry as temporal support needs to be implemented in the application logic [14]. There is no automated system that allows for the easy generation of temporal

[1]Note: A manager is still an instance of an employee

database schemas. This means that companies and organisations need to allocate developers, time, and money in order to build and maintain these features.

Additionally, there is no tool for graphical modelers to integrate temporal elements into their models. Existing graphical modelling tools are designed around general ER and not built with temporal databases in mind and thus provide no support for modelers (such as the ability to validate models and support for temporal SVGs in Trend) above simple diagramming features.

The goal of the project is to build a web application that makes it easier to design and implement temporal databases. It will accomplish this by allowing the user to draw a temporal conceptual data model using the Trend modelling language [4]. The tool will prevent the user from being able to make invalid diagrams, informing the user of their mistakes.

The user will be able to download this model and share it. If they intend to implement the model, they will be able to generate a schema that they can easily implement in an actual database management system (DBMS). This will create all the necessary tables and logical enforcement mechanisms for the database, removing the need to dedicate development time to implement these features.

3 RELATED WORK

For the schema creation aspect, there already exists a well-known algorithm for converting atemporal ER diagrams into relational databases [8, 15]. This algorithm will be implemented and extended for the project. Similar algorithms exist for different TCDMLs [11, 22], but these will not be implemented as they do not have overlapping features with Trend.

In 2011, an update to SQL was published that introduced temporal data support [14]. This included valid-time , transaction-time, and bitemporal table support, as well as referential integrity features. While this is a good first step, it does not include the transition constraints of Trend. For the project, we will be implementing our schemas in a database management system (DBMS) that implements these temporal features.

For the graphical modelling tool, we can consider three similar graphical modelling tools for other conceptual data modelling languages: ERD Tool, Crowd and Draw.io. The Entity-Relationship Diagram Tool (ERD Tool) is used to graphically model ER Diagrams [20]. It focuses on a simplified and streamlined user-interface and fulfilling all the necessary requirements of ER Diagrams, such as cardinality, participation, and recursive relationships [20]. 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.

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 [9]. The temporal ER graphical reasoning support exists in the form of checking for class consistency, checking for relationship consistency, discovering implied class and cardinally constraints [6]. This is more extensive than the ERD tool, yet does not contain any of the temporal constraint we would want to implement.

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 the existing modeling tools, Draw.io has the most extensive, polished, and feature rich user-interface. It includes detailed style customization for graphs (text, colour and arrangement), premade templates for modellers to use, and an extensive palette of objects. Reasoning support can be added using the Ontopanel plugin [7].

4 PROCEDURES AND METHODS

4.1 Modelling Tool

This part of the project deals with providing users a graphical interface to create their temporal conceptual models. As part of the requirements (the source and importance of these requirements will be discussed later), the following is a list of planned features for the modelling tool. Features are ordered from most essential to least essential.

- **Full Trend language support:** Support of all Trend features including full extended entity relationship support in addition to temporal elements (attributes, entities, relationships) and dynamic extension & evolution. Implemented using a palette of SVGs rendered on-screen.
- **Dragging and dropping:** Ability to drag and drop objects from the palette to a canvas where these objects can be moved.
- **Schema Generation:** The user will have the ability to generate a database schema based off their model, this forms a large component of the project and will be explained in its own section.
- **Ability to save and load models:** This will be in the serialisation chosen for our project (such as XML or JSON), models can be read from or written to disc.
- **Undo and Redo:** These are standard features expected in almost all modern software; they will help speed up model design.
- **Linking of Objects:** Users should be able to click and drag from one object to another to link them. The user will also be able to specify the type of link (relationship) between the two said objects.
- **Graph Validation:** This feature will be useful in helping new modellers with Trend. If the graph is edited, it will validate if the current state of the graph in accordance to Trend, if not, a relevant error will be thrown to the user.
- **Export to different formats:** The graphs should be able to be exported as more useful formats for the user, such as: PNG, SVG, XML, and JPEG
- **Shortcut support:** Support of common and expected shortcuts, such as: CRTL+A, CRTL+C, CTRL+V, CTRL+X, as well as some new shortcuts that are specific to our program.
- **Notation Conversion:** Users should have the ability to convert between the specific EER notation they want to use.

- **Canvas Grid:** The canvas that users interact to create their models on top of should support snap-to-grid and be infinitely expandable in every direction.
- **Other UI features:** User should be able to resize and rotate objects on the canvas.
- **Overview Window:** The final product should include an overview map that shows a zoomed-out high level view of the entire model to assist the user in navigating their model.

Additionally, we want the tool to have high performance and be able to handle large models. Since this project is being built with a strict deadline and flexible features, an AGILE development process will be followed, where the project production time will be split into five two-week sprints, which are outlined in Figure 2. Since the end project will be used by modellers, this project will also focus on client-driven development. The final result will be implemented using Model-View-Controller Architecture and a diagramming JavaScript library (such as GoJS or JointJS). This method will be preferred over extending an existing application, since existing applications are not designed to be extended off for new project creation (or most existing applications are closed-source). The final decision for the project is to choose XYFlow and react for the front-end, as other tools has high costs to use and XYFlow is free to use.

Understanding underlying tools and libraries (such as XYFlow, GoJS, and JointJS) enough to successfully create our own features will be a major design challenge. Additionally, the graph validation will be difficult as we will need to successfully check the graph syntax over the chosen serialisation of the graphs and also write these rules. Since our serialisation is likely to be JSON (due to the underlying library we will be using - which is XYFlow), it will be necessary to implement the ability to validate the graph using this serialisation. This will likely necessitate some form of standard that needs to be created and then enforced (such as Backus-Naur Form).

Since this tool is designed to be used by both beginner and expert modellers, we need input from both demographics in order to represent the full range of users. Thus, the graphical modelling tool will be evaluated on the following factors:

Representing the expert modellers, our project feature specification will be informed by our project supervisors (features included are ones that they deem important in their extensive experience in modelling), and the success of the tool will be our ability to implement all of the required features. Additionally, to test the validation feature, users will attempt to create mistakes on the graph which the validator cannot pick up (we already have an idea on common mistakes made by modellers that we must detect due to past questions answered in previous computer science exams).

On the other hand, beginner modellers also need to be represented in the user testing for our application. For them, it is important that the user interface is easy to understand and use. This will be evaluated by the users' ability to create models using the tool. To do this, we will conduct task-based tests in the form of questions that will be provided to beginner user testers. These questions will outline model specifications in controlled natural language, for example, in Figure 1, the temporal optional future dynamic extension could be described as follows: *Manager is an employee. An employee may also become a Manager.* The users must use these specifications to create conceptual data models in our tool, as well as another popular tool that would usually be used to create temporal conceptual data models, such as Draw.io or pen and paper. During these tests, any mistakes that users make will be recorded, as well as the time it took to create the model. It will be noted how fast beginners can model and how few mistakes they make when compared to an already existing tool and will be considered a success if users are quicker and more accurate when using out tool.

This will require us to recruit user testers, who will be recruited from the UCT computer science department. Computer science students fit the ideal for a "beginner modeller" as they have all done conceptual modelling in their undergrad, however do not posses extensive experience in the topic. Ideally we need to recruit as many students as possible, but we want a minimum of at least 12 user tester. To boost recruitment numbers, a form of remuneration or reward will be provided to user testers as an incentive.

Some performance testing will also be used, such as testing the loading, using, and saving of very large models.

4.2 Schema Creation

The principal task of this part of the project is generating a schema based on the model designed by the user. This will involve algorithm design, implementation, and testing.

A SQL file would be generated which, when run in a MariaDB server, would generate all the tables and logical enforcement mechanisms specified by the user. This would substantially reduce the barrier to entry of temporal databases by reducing the development costs associated with them.

The following alterations have been made to the standard algorithm for converting ER diagrams into schemas [8, 15]:

(1) Whenever an entity with a clock icon is added, its table needs to be an application time table.
(2) For one-to-one and one-to-many temporal relationships, create a new application-time table for each relationship. Have the combination of each participating entity's keys act as the primary key.
(3) For many-to-many and N-ary relationships, if the relationship is temporal then they need to have an application-time table.
(4) Temporal attributes should get their own application-time table that should include the primary key of the entity.
(5) Temporal multivalued attributes need to have an application-time table.
(6) For every temporal transition constraint in the model, add a trigger to the table that enforces it. The triggers need to enforce dynamic evolution and extensions, including if they are mandatory and optional. The quantitative temporal constraints should be enforced by checking the periods.

We made deliberate decision to have separate tables for temporal attributes and relationships. This avoids the confusion of having multiple periods in one row, although it does result in more tables.

This part would require extensive testing of the software to ensure correct output and performance. Unit testing for the serialisation input would be needed to ensure that the program works for

all diagrams. The resulting databases would also need to be evaluated using functional testing to ensure that the constraints function as expected. The conversion algorithm would need performance testing to ensure that it runs in a reasonable amount of time for various models.

All of the tests will be designed in the first phase of the project and be performed as the features are implemented. The individual features will be tested using many small models that only include specific temporal features. For example we could have a model that only includes a mandatory dynamic extension in the future between two entities.

After all the temporal features are developed, there is a dedicated period of testing to ensure proper functionality. The models used will be large and incorporate all the temporal constraints, allowing us to test the interactions between various constraints.

Since this feature has an inflexible scope, an AGILE approach would not be applicable. Designing a minimum viable product early on and spending most of the project improving it is not feasible for the amount of time given. As a result, an iterative development approach would need to be adopted, where the most important features are completely developed early in the project, and the less important features developed later. Focusing on the most important features early will allow us to still have a functioning product if we are unable to complete the project on time.

The core feature that needs to be implemented is supporting atemporal databases. Since the proposed algorithm builds upon the atemporal algorithm, we need to first implement it before we can extend it with temporal support. The next thing would be implementing the basic transition constraints (i.e. dynamic evolution and extensions). After this the quantitative transition constraints will be developed.

A big challenge anticipated is being able to implement the more advanced interactions between constraints. Some interactions might not even be possible to accommodate. For example, if an employee gets promoted to a manager, their relationships also need to be updated. This has the potential to substantially increase the complexity of the program.

MariaDB was chosen as the database management system (DBMS) for the project as it incorporates the necessary temporal features, while still being open source and free [17–19]. A detailed comparison of DBMSs was given in the literature review [16].

This feature will be coded in Javascript so that it is easier to integrate with the entire system.

5 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

For the graphical modelling tool, an ethical issue arises as user testing is needed in informing the best end product. Testers must be able to give informed consent and we should also ensure there data during participation is protected and kept private. We must remain transparent about the testing to the users at all time. For user testing, other computer sciences students will be chosen to conduct the testing. The main reason for this is that computer science students would not be considered as vulnerable persons due to their education; they posses the necessary computer systems understanding to be fully informed of the repercussions of joining user testing and thus can give informed consent for this project. All information about the user testers will be kept anonymous.

We also could run into a legal issue with utilizing other libraries (such as XYFlow, JointJS, and GoJS) to build our own tool. Although these libraries are open source, they are often protected under various licenses that impose specific restrictions and obligations. Our chosen tool, XYFlow, is protected under the MIT licence thus we can use the library provided our tool is not commercial. Additionally, since this tool is designed as a research tool and not for commercial use, the code for this project will be open source. Due to the above factors, we believe this project should receive ethics clearance, and we have considered the risk of not receiving ethics clearance in Table 2.

6 ANTICIPATED OUTCOMES

6.1 Impact

The goal of this project is to build a tool that makes designing and implementing temporal databases easier. This has the potential to lower the barrier to entry for temporal modelling and temporal databases and increase interest in the field from both industry and academia. The tool will additionally help increase productivity for more experienced modellers. This project will also help cement Trend as a standard language for temporal conceptual data modelling.

6.2 Key Success Factors

The graphical tool will be considered a success if it can satisfy all its requirements. This includes succeeding in the user testing by being more efficient and easy to use compared to existing alternatives and fulfilling all planned features and other non-functional requirements (explored through performance testing) required by more experienced modellers.

The schema creation tool will be considered a success if it can implement all of the temporal features of Trend. These features should be implemented with minimal bugs and should be able to run on a DBMS. The algorithm also need to be able to work for large models and take a reasonable amount of time to run.

Given a list of n nodes and e edges, the algorithm processing these should have a time complexity smaller than $O(n \cdot e)$ i.e. for every node, looping through all the edges would not be acceptable. The ideal time complexity would be linear time, i.e. $O(n + e)$, however solutions that have a worse time complexity would be accepted, as long as they are better than $O(n \cdot e)$.

7 PROJECT PLAN

7.1 Risks

Refer to Table 2 for a Risk Assessment Matrix.

7.2 Timeline

The project divided into four phases. The first phase is the Design Phase. This phase includes designing the serialisation, user tests, the SQL implementations for the schemas, and the testing associated with them. This is followed by the Development Phase, in which we build the tool and perform the testing. The Writing Phase is dedicated to the final project papers. The last phase is the Poster

and Website Phase, in which we design and develop the poster and website. A detailed overview of the timeline can be seen in the Gantt Chart in Figure 2. The deliverables are stated in Table 1.

Table 1: Deliverables And Milestones

Deliverable/Milestone	Date
Ethics Application Deadline	6 May 2024
Serialisation Complete	13 May 2024
Initial Combined Prototype	24 June 2024
Fully Functional Prototype	22 July 2024
Project Progress Demonstrations	22-26 July 2024
Code Finalised	12 August 2024
Project Paper Complete Draft	23 August 2024
Project Paper Final Submission	30 August 2024
Project Code Final Submission	9 September 2024
Final Project Demonstrations	16-20 September 2024
Poster Due	27 September 2024
Website Due	4 October 2024

7.3 Resources Required

Since the graphical tool is being designed as a web-app and will be able run on almost any machine, no additional resources are required for its development. However, people are needed for the user testing (specifically beginner modellers). All software tools built upon are open-source and free, thus no licensing or payments are needed.

The database schemas generated will need a database client that can run a MariaDB server to function. MariaDB is open-source and free, meaning that acquiring it will not present any challenges.

7.4 Work Allocation

The project work is allocated on the two main components that need to be produced. Richard Taylor will focus on building the graphical modelling tool while Stephan Maree will focus on the schema creation aspect. These tools will use a shared serialisation. These two tools will be designed to function separately, but ideally they will be integrated into one final product, where users' conceptual models can be converted into temporal databases.

REFERENCES

[1] Alessandro Artale and Enrico Franconi. 1999. Temporal ER Modeling with Description Logics. In *Conceptual Modeling — ER '99*, Jacky Akoka, Mokrane Bouzeghoub, Isabelle Comyn-Wattiau, and Elisabeth Métais (Eds.). Springer, Berlin, Heidelberg, 81–95.

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

[3] Andreas Behrend, Philip Schmiegelt, Jingquan Xie, Ronny Fehling, Adel Ghoneimy, Zhen Liu, Eric Chan, and Dieter Gawlick. 2015. Temporal State Management for Supporting the Real-Time Analysis of Clinical Data. *Advances in Intelligent Systems and Computing* 312 (01 2015), 159–170. https://doi.org/10.1007/978-3-319-10518-5_13

[4] S Berman, C Maria Keet, and Tamindran Shunmugam. 2024. The temporal conceptual data modelling language Trend. *Under review at Data Intelligence* (2024).

[5] Michael H. Böhlen, Anton Dignös, Johann Gamper, and Christian S. Jensen. 2018. Temporal Data Management – An Overview. In *Business Intelligence and Big Data*, Esteban Zimányi (Ed.). Springer International Publishing, Cham, 51–83.

[6] Germán Braun, Christian Gimenez, Pablo Rubén Fillottrani, and Laura Cecchi. 2017. 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).*

[7] Yue Chen. 2022. Ontopanel: a diagrams. net plugin for graphical semantic modelling. (2022).

[8] R. Elmasri and S.B. Navathe. 2004. *Fundamentals of Database Systems.* Addison-Wesley.

[9] Christian Gimenez, Germán Braun, Laura Cecchi, and Pablo Rubén Fillottrani. 2016. 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).*

[10] Heidi Gregersen. 2005. Timeer plus: A temporal eer model supporting schema changes. In *British National Conference on Databases.* Springer, Springer, Berlin, Heidelberg, 41–59.

[11] Heidi Gregersen, L. Mark, and Christian Søndergaard Jensen. 1998. *Mapping Temporal ER Diagrams to Relational Schemas.* Number TR-39 in Technical Report. Aalborg Universitetsforlag.

[12] C Maria Keet and Sonia Berman. 2017. 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 (Lecture Notes in Computer Science, Vol. 10650).* Springer, Cham, 437–450.

[13] C Maria Keet and Sonia Berman. 2021. A Brief Introduction to Temporal Modelling with TREND.

[14] Krishna Kulkarni and Jan-Eike Michels. 2012. Temporal features in SQL: 2011. *ACM Sigmod Record* 41, 3 (2012), 34–43.

[15] Vincent S Lai, Jean-Pierre Kuilboer, and Jan L Guynes. 1994. Temporal databases: model design and commercialization prospects. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 25, 3 (1994), 6–18.

[16] S Maree. 2024. Literature Review: Realising Temporal Information Modelling. Literature review for CSC4019Z. University of Cape Town, South Africa. 6 pages.

[17] MariaDB. 2024. *Application-Time Periods.* https://mariadb.com/kb/en/application-time-periods/

[18] MariaDB. 2024. *MariaDB Server: the innovative open source database.* https://mariadb.org/

[19] MariaDB. 2024. *System-Versioned Tables.* https://mariadb.com/kb/en/system-versioned-tables/

[20] Ron McFadyen. 2015. AN ERD TOOL. *Submitted to an international journal* (2015).

[21] Fabio Persia, Fabio Bettini, and Sven Helmer. 2017. An Interactive Framework for Video Surveillance Event Detection and Modeling. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Singapore, Singapore) *(CIKM '17).* Association for Computing Machinery, New York, NY, USA, 2515–2518. https://doi.org/10.1145/3132847.3133164

[22] C Theodoulidis, Pericles Loucopoulos, and Benkt Wangler. 1991. A conceptual modelling formalism for temporal database applications. *Information Systems* 16, 4 (1991), 401–416.

Table 2: Risk Assessment Table

Risk	Explanation	Likelihood	Mitigation	Impact	Response
Group member leaves	This could be a result from them leaving honours, or taking a leave of absence.	Low	Keep software design modular & remove dependencies	Very High	Arrange meeting with supervisor to discuss new scope for project.
Scope Creep	The continuous expansion of the project's scope as more requirements are added after the project has started	High	Take the Agile "Onion" approach, where important features are prioritised	Medium	Same as mitigation, prioritise important features before adding superfluous features
Dependency and Technology Risks	Libraries or frameworks used in our project become deprecated or cause unexpected issues that slow progress	Low-Medium	Limit unnecessary dependencies and extend stable & Modern libraries	High	Change dependencies and potentially re-evaluate scope
Technical Debt	Shortcuts and short-term bug fixes lead to less organised code that must be later reorganised	Medium	Regular Refactoring	Medium	Same as mitigation - focus on refactoring
Project falls behind schedule	We are late on achieving milestones and deliverables	High	Regular review with project supervisor as part of the AGILE design process	High	Re-evaluate and potentially reduce project scope
Unable to acquire Ethics Committee approval	Our user testing would require approval from the Ethics Committee.	Low	Stick to widely used standards and guidelines	Very High	Evaluate design based on heuristics instead of user tests.

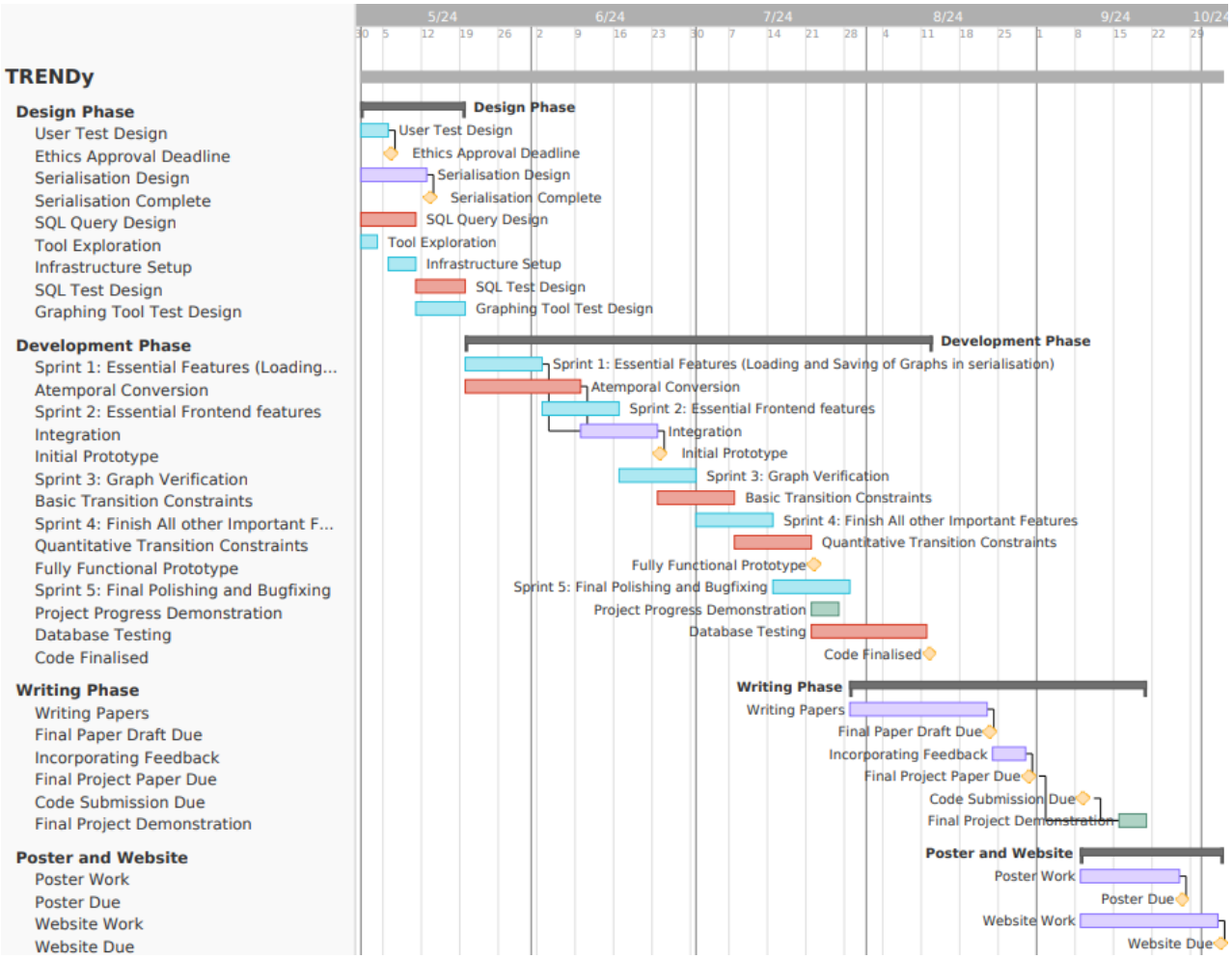

Figure 2: The Gantt Chart for the project. In this the blue bars are assigned to Richard, the red bars belong to Stephan, the purple bars are done by both team members, and the green bars correspond to the project's proposals.