Cleaning up the clutter in interactive visualisations of labelled transition systems

Wike Johanna Duivenvoorden
wike.duivenvoorden@student.uva.nl

June 30, 2025, 34 pages

Academic supervisor: prof. dr. M. Worring, m.worring@uva.nl
Daily supervisor: David Smelt, david.smelt@axini.com
Host organisation/Research group: Axini B.V. , https://www.axini.com

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Abstract

This study investigates the visualisation methods used to depict Labelled Transition Systems used in Model-Based Testing. Visualisation is currently done with node-link diagrams. These diagrams are insufficient for the growing amount of data to be visualised. The depiction of real-world systems with abundant data is often characterised by a high degree of complexity, which can impede the interpretation of graphs and the effective presentation of data. Consequently, the exploration of alternative approaches is imperative. Design criteria were established, and alternative visualisation methods were selected based on those. Five novel visualisations were made using in-house data from Axini, a Dutch Model-Based Testing company. A survey was sent out to compare the original situation and the five new visualisations with the design criteria. Our analysis shows that novel visualisation approaches outperform traditional methods in terms of clarity and user engagement. Furthermore, visualisation methods with interactive features have a positive impact on usability and expressiveness.

Contents

List of abbreviations

AML	Axini modelling language
AMP	Axini modelling platform
DAG	Directed acyclic graph
DFD	Data flow diagram
ER	Entity relation
InfoVis	Information visualisation
LTS	Labelled transition system
MTB	Model-based testing
STS	Symbolic transition system
SUT	System under test
UML	Universal modelling language

Chapter 1

Introduction

Software testing is an important technique to assess the quality of systems. There are many ways to test a system. One of those ways is Model-Based Testing (MBT), which has gained popularity among academia and in industry [1–4]. MBT generates test cases based on a model of the system under test (SUT). The behaviour of the SUT is modelled with a variation of a state-transition model, where the states in the model represent an abstraction of the states of the system, and the transition between the states represent the actions that the system may perform [3].

To facilitate the development of these models for testing, they can be expressed as a Labelled Transition System (LTS) or as a Symbolic Transition System (STS). An LTS can depict any conceivable execution path and an STS combines that with data, variables, and data-dependent logical conditions [3]. Visualisations of the model, which provide insight into the circumstances generating the (un)desirable behaviour, give the user a visual representation of the system and aid in rapidly identifying the model's flaws [5]. Typically, node-link diagrams (state-transition models) are used to visualise LTS models [1, 3, 5–7]. These diagrams are widely used to model systems in physics, linguistics and computer science and are a form of Directed Acyclic Graphs (DAG) that model relationships between objects [5].

A problem arises when the model is to represent a complex, real-world system [8]. Encoding all possible data values in transitions and mapping the data to a DAG, leads to *state space explosion* and causes the visualisation to become too complex to create a good overview for the users [3]. To try and reduce the state space explosion, most studies working with large models in the field of information visualisation have found that graphs can become difficult to navigate rather quickly.

Additionally, large visualisations are associated with a lot of visual clutter; we aim to identify visualisation methods that minimise this. We base our research on explored and mature solutions, i.e. clustering [5, 6] and semantic zooming [4].

1.1 Problem statement

The specific problem at hand is the challenge of how to visualise a large amount of information without overwhelming the user. As systems grow, exploring information collections becomes increasingly difficult, and a textual representation becomes increasingly harder to comprehend [9]. Previous studies have looked at models several times larger than the models used in MBT, averaging around 1000+ visualised nodes, and the resulting visualisations are only for displaying or presenting. Our research is about graphs and models, where 100 visualised nodes are a considerable amount. We discuss some studies which utilised a different visualisation paradigm/technique to display LTS models. We want to create comprehensive visualisations that can be used effectively during testing, presenting, and debugging. For those purposes, we need to look beyond node-link diagrams. To achieve this, we turn to established visual notations in software engineering: Universal Modelling Language (UML) models and tree diagrams.

1.1.1 Research questions

The present study sets out to achieve two objectives: firstly, to investigate an alternative way to visualise complex LTSs, and secondly, to ascertain the effectiveness of adding such a new feature into the already existing MBT environment of Axini. This leads to the following research questions:

> **RQ 1** How can alternative visualisation types enhance the comprehension and analysis of complex Labelled Transition Systems for users?

RQ 2 How can interactive features and user-driven exploration be incorporated into Labelled Transition Systems visualisations to support model checking or analysis more effectively?

And the following sub-research questions:

- **RQ 1.1** In what ways does a novel visualisation approach for Labelled Transition Systems outperform traditional graph-based visualisations in terms of scalability, clarity, and overall user experience?
- **RQ 2.1** How does the inclusion of interactive features impact the usability, performance, and expressiveness of the Labelled Transition Systems visualisation tools for expert and new users?

1.1.2 Research method

This study uses the methodology to observe users doing their work with the visualisation while thinking aloud and sharing their thoughts. This way we could inspect and question how the users go about using the visualisation and where they get stuck, or what their wishes are for an alternative.

Based on this information, design criteria for the alternative visualisations were set up. Additionally, the requirements set by users were used to design a prototype of the ideal visualisation situation.

We selected a known LTS model from Axini to use as input data for an open-source tool called `cytoscape-expand-collapse`. With the tool, a proof of concept was constructed that showcases the functionality of the novel visualisation method employing semantic zooming combined with glyphs. At the same time, alternative visualisations were being realised in D3 and MermaidJS.

To compare and validate the visualisation methods, a survey was sent out to expert users connected to Axini, developers who designed the current situation and to students of the MSc Software Engineering. Participants were asked to provide feedback via a variety of questions about the visualisations.

1.2 Contributions

This research aims to expand on previous research about visualising LTS models in the MBT domain. Our research makes the following contributions:

1. Design criteria for a visualisation of Labelled Transition Systems used for Model-Based Testing.
2. A comparison of alternative visualisation methods for Labelled Transition Systems, selected based on user criteria.
3. A novel way of visualising labelled transition systems by combining semantic zooming with glyphs.

1.3 Outline

Our work is structured in the following way: in Chapter 2, we describe the background of this thesis and explain the related domains and the key terms. Chapter 3 contains the previous work done in the relevant domains related to this thesis. Chapter 4 describes the design criteria we established and the visualisation methods chosen to implement. Results are shown in Chapter 5 and discussed in Chapter 6. Finally, in Chapter 7 we present our concluding remarks together with future work.

Chapter 2

Background

This chapter serves as an overview of relevant topics, offering necessary background information for this thesis. We will introduce the related topics and domains such as information visualisation, Model-Based Testing, Labelled Transition Systems, visualisation construction, and aesthetics. Together with Chapter 3 on related work, these sections provide context for the project.

2.1 Model-Based Testing

Model-Based Testing (MBT) is a method that allows for automatic testing of software by creating a model of the desired behaviour of the system under test (SUT) [2]. From this model, test cases are automatically created and executed [3]. In [1, 2], Tretmans defines MBT as formal, specification-based, active, black-box, functionality testing.

It is formal because the model describing the expected behaviour of the SUT is given in a formal language that upholds the **ioco** relation [1]. Ioco is short for input-output conformance, and Tretmans introduced the ioco relation as a way to formalise when an implementation conforms to a specification.

The properties being tested concern functionality, as opposed to, for example, performance or usability properties. Functionality is defined in this context as testing whether the system correctly does what it should do in terms of correct responses to given stimuli.

The specification is the basis and starting point for testing. It prescribes what the SUT should and should not do [1, 2]. The specification is given in the form of a natural language description of the behaviour of the model, to which the behaviour of the SUT must conform. This specification is assumed to be correct and valid and will be turned into a Labelled Transition System by human interpretation.

Finally, the SUT is seen as a black box without internal detail, which can only be accessed and observed through its external interfaces. The externally observed behaviour is then checked against the specification.

2.1.1 Labelled transition systems

A Labelled transition system (LTS) is a model used in formal testing that consists of a set of states and a set of transitions between those states. These represent the system's behaviour and depending on the current state the system is in, it can perform certain actions that are either inputs or outputs [1]. It is formally defined as:

Definition 1 *a Labelled transition system is a 4-tuple* $\langle Q, L, T, q_0 \rangle$ *where*

- Q*, countable, non-empty set of states;*
- L*, countable set of labels;*
- $T \subseteq Q \times (L \cup \{\tau\} \times Q)$*, with* $\tau \notin L$*, is the transition relation;*
- $q_0 \in Q$ *initial state.*

If the system is in state A, it may perform action B and go to state C. The composition of transitions expresses that the system can perform a sequence of actions. Because of non-determinism, it may be the case that the system performs the same sequence of actions, but ends up in a different state.

An LTS with inputs and outputs is called an Input-Output transition system and is similar to a normal LTS [1]. This variation is always ready to perform any input action. Inputs are usually denoted with "?" and outputs with "!".

In Figure 2.1, an input-output LTS is depicted as a node-link diagram. The nodes are all the machine's states and the links represent the transition. In this example, a person should insert a certain amount of coins to be able to receive a coffee. It showcases *state space explosion* because of all the different combinations listed [3].

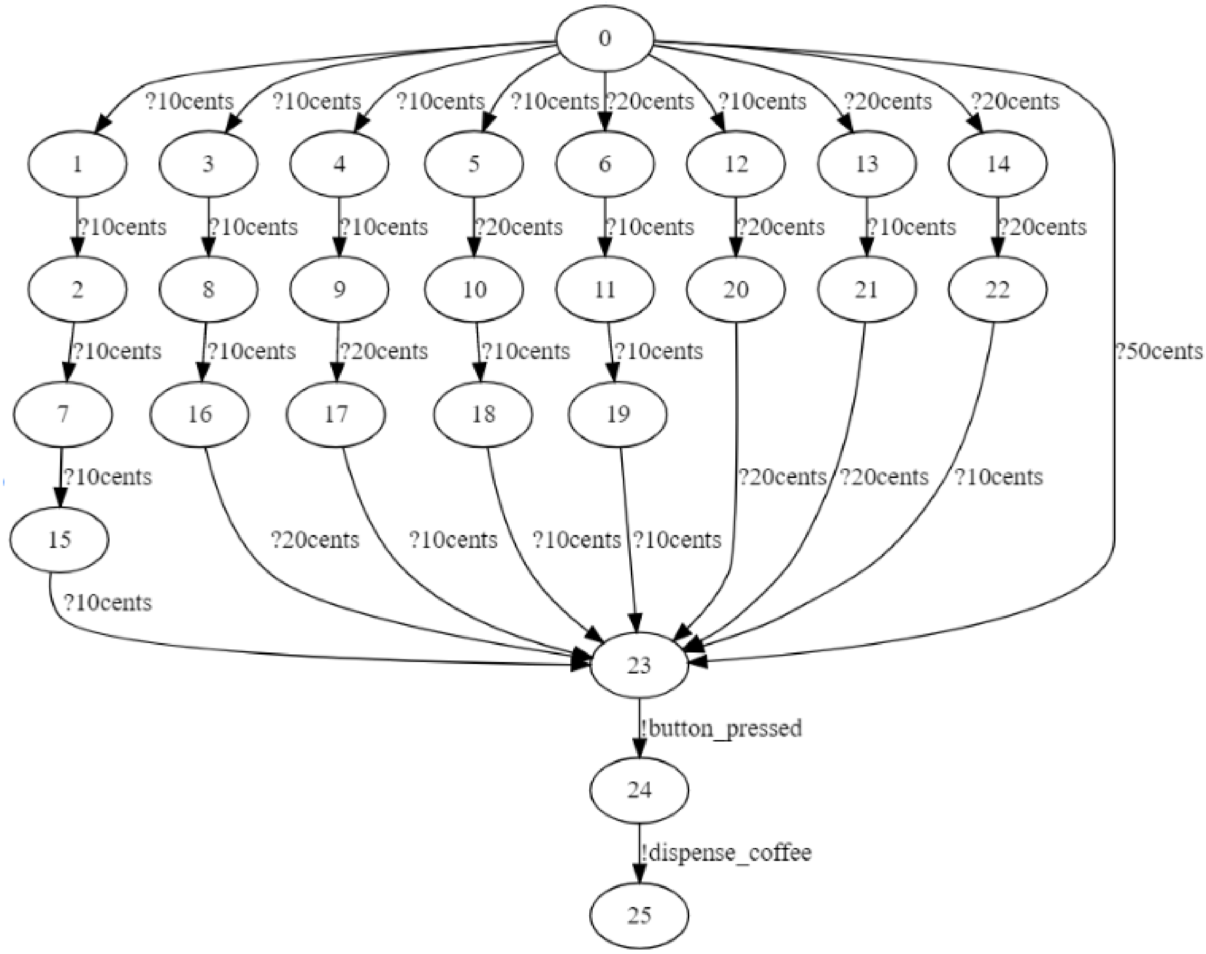


Figure 2.1: An example of a coffee machine as an LTS with state space explosion.

2.2 Information Visualisation

The field of Information Visualisation (InfoVis) is described as "the use of computer-supported, interactive visual representations of data to amplify cognition" [10, 11]. It refers to the way data is presented, in a way that users can understand it without needing to look at the statistical details [12]. InfoVis systems are used for exploratory tasks [11], such as exploring a large information space in an efficient and effective way, but also for explaining findings to a larger audience [11, 13]. They provide a less cognitively loaded method of exploration, which is possible because humans have unique visual system capabilities, which allows them to detect patterns and features in visualised data in a short amount of time [13].

There are two main uses for visualisations: exploration and presentation, highlighted by Wijk in [13]. Exploration for when users have a limited understanding of the system and prefer a less cognitively loaded way of exploring [11]. Presentation is used to communicate results [13]. Though researchers in the field consider exploration more important than presentation, Wijk states that presentation is just as valuable. In this thesis, the resulting visualisations are aimed to be used for both purposes.

2.2.1 Diagrams over words

Paired often with visualisations are the expressions "Seeing is believing" and "A picture is worth a thousand words" [11]. Research shows that pictures do help create an overview of data that otherwise is hard to comprehend at a glance. Therefore, a diagram is better than words [11, 14].

To convey information, there are two kinds of representations: sentential and diagrammatic. Sentential representation expressions form a sequence corresponding to a natural language description of a problem. More simply stated, sentential representations are a textual description of a problem or a model. Diagrammatic representation expressions map to a diagram depiction of a problem. As a logical consequence, diagrammatic representations are diagrams. Larkin and Simon explain that graphical visualisation can facilitate more efficient task performance [14]. This is achieved by replacing complex logical conclusions with rapid perceptual ones and by reducing the amount of information required to complete a task.

2.3 Visualisation design

The Visual Information-Seeking Mantra of Shneiderman summarises many visual design guidelines and provides a framework for designing InfoVis applications. It is a well-known sequence in explorative visualisation design: overview, zoom and filter, details on demand [9].

Overview

To get an overview of the entire collection of data in the network, a layout algorithm is used, and it plays an important role in generating the visualisation [12]. There are two types of layout algorithms, iterative and static. An iterative layout algorithm improves the layout with each iteration until an acceptable layout is achieved, whereas a static layout generates the layout in a single computation [5].

It has become a well-known problem that when a collection of information becomes significantly large that creating an overview becomes difficult. The bigger the graph, the harder it is to make it look nicely structured [9, 15, 16].

As Heer *et al.* states in [16]:

> "As networks get large and highly connected, node-link diagrams often devolve into giant hairballs of line crossings." (p. 67)

An example of an iterative layout algorithm is a force-directed algorithm, modelling the graph as a force. Each node gets a weight, and each transition is modelled as a spring, attracting connected nodes and repelling other nodes. The simulation is run iteratively until a stable equilibrium is found [16]. The resulting graph has achieved the optimal layout with the minimal amount of line crossings. It is a popular algorithm for LTS visualisations [5].

Zoom and filter

The amount of data can be overwhelming, so a user must be allowed to zoom in and filter out items of interest[6]. Users typically have an interest in a subset of a collection, and they require tools to control the zoom focus and factor. Smooth zooming enables users to maintain their sense of position and context [9]. As described by Ham *et al.* in [6], the user can simplify the visualisation by selecting a cluster of nodes, and only that cluster and its descendants are then shown.

Filtering can be achieved by control widgets for dynamic queries. This affords users agency over the displayed content; in this way, they are enabled to focus on content that aligns with their interests by removing superfluous objects [9].

Details on demand

In selecting an item or group, the user may request further information as required. Once a collection has been reduced to a few dozen elements, it should be simple to navigate the specifics of the group or individual objects. The conventional approach entails the selection of an item, which initiates the opening of a pop-up window containing the values of each attribute [9]. Related to this is the concept of semantic zooming, further explained in section 3.2.

2.4 Aesthetics

What defines a good visualisation? In the field of graphs, it means that the visualisation is readable and comprehensible [15]. This will not only make the resulting graph look nice but also make the structure and meaning visible. To do so, there are two areas to look at: graph structure and semantics.

The structure of a graph is about the layout of nodes, edges and graphs. Sometimes nodes overlap or edges become long and complex [15]. This is not only bad for the overall aesthetics, but also for the comprehensibility of the graph. So, selecting a layout that works for a particular graph is important for the user experience.

Semantic is about the labelling, colour, and shapes of nodes and edges, the elements that might draw the eye of the user at first glance [15].

How a graph is perceived is influenced by its presentation [17]. The look also determines whether people will dive deeper to fully understand the visualisation. We are first drawn to certain elements of the visualisation that stand out due to colour, size or shape, then to the functionality, performance and interaction.

The amount of knowledge acquired by using visualisation depends on the data, the specification, the perceptual skills, and the prior knowledge of the observer [13, 17]. Generally, for experts, visualisations will be more useful than for novices [12, 17]. In an ideal scenario, the knowledge extracted from data in general is an objective process, and the same results are produced by all who repeat the analysis. However, the interpretation of visualisations is a subjective process.

2.4.1 Visual encodings

Visual encodings, also known as visual variables or visual channels, are an important part of designing a visualisation (illustrated in Figure 2.2). They define the dimensions of the graphic design space [18]. The node shape, colour, and size all influence how a visualisation is perceived. Those encodings map the data to the visualisation and define a set of primitives for constructing visual notations. From that set, graphical symbols can be constructed by specifying particular values for visual variables (e.g., shape = circle, colour = blue) [18]. Common consensus about the effectiveness of the encodings is colour $\prec$ size $\prec$ shape $\prec$ orientation (the symbol $\prec$ reads as *precedes*) [19]. These encodings facilitate a one-to-one

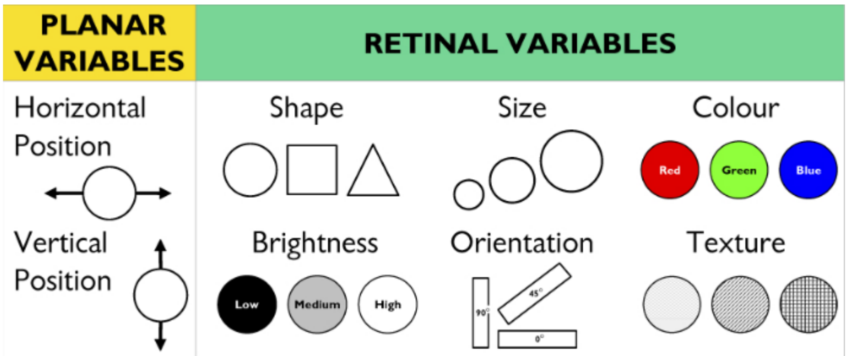


Figure 2.2: The visual variables that combine to make visual notations [18]

mapping of data into graphical features. Moody explains this as the principle of semiotic clarity [18]. This is one of the principles used for designing effective visual notations. Designers can make unlimited combinations with the visual variables; however, simple encodings are often better than complicated ones [16].

2.4.2 Glyphs

A glyph is a small visual marker, often used in multivariate visualisations, that grants the user access to multiple levels of the visualisation [19]. It differs from icons, symbols, and pictograms because a glyph is constructed from single or multiple data attributes, and thus it also adapts to said data.

A glyph is made by a set of visual encodings, and each visual encoding represents a different variable from the multivariate data. The visual encodings, colour and size, are more dominant and help focus the user's attention. Therefore, effective glyph visualisation should carefully choose and combine different visual encodings [19].

Chapter 3

Related work

While researching the domain related to this thesis, we came across related work that assisted us in gaining insights into the current state-of-the-art and served as inspiration for features of this project. We divide the related work into the following categories: Graph visualisation, Semantic zooming, and Clustering and grouping.

3.1 Graph visualisation

Various sources have researched how the layout of a graph affects the overall visualisation [6, 15, 20, 21]. Although their research was done on much larger graphs than the use case for this study, we present the found heuristics and divide them into node placement and edge placement.

Nodes

For the positioning of the nodes, Ham *et al.* recommend three practices. Firstly, they advise keeping edges between nodes short, so that nodes are close to their neighbours. Secondly, the distance between nodes of the same cluster should be maximised. In this way, the clusters of nodes do not get too crowded. Lastly, if it is possible, highlight the symmetry of the graph. This could be either local or global symmetry. An example of a symmetric graph is a tree diagram because of the hierarchical layout, which also provides local symmetry in the subsections of the graph [6].

Supporting this view, Bennett, Chris *et al.* write that an even distribution of nodes over the whole graph gives a regular impression and makes the visualisation more attractive [15]. Nodes should not be layered or stacked, except for nesting or clustering. Clustering, however, clashes with the recommendation of an even distribution of nodes.

Edges

Minimising the number of edge crossings is the most logical and most agreed-upon heuristic for edge placements, as having a significant number of edge crossings could lead to occlusion and hiding of potential patterns [6, 7, 15].

In [6], Ham *et al.* explain their layout for the edges. Making all the arrows point one way, signalling the flow, causes large *back pointers*, which span several ranks when nodes of a lower rank are connected to those of a higher rank. Alternatively, making arrows point both ways is useful to highlight connections between clusters, but this arrangement obscures the flow. To make a difference between flow arrows and backpointers, Ham *et al.* decide on the option to curve the backpointers, which also highlights the cyclic properties of the backpointers.

3.2 Semantic zooming

Semantic zooming, also called level-of-detail rendering, is a way to hide/show the complexity of a visualisation. World maps are a well-known implementation of this visualisation: the user can view both the world as a whole, but also zoom in to street level. This interpretation of semantic zooming supports the details on demand from the mantra for visualisation design.

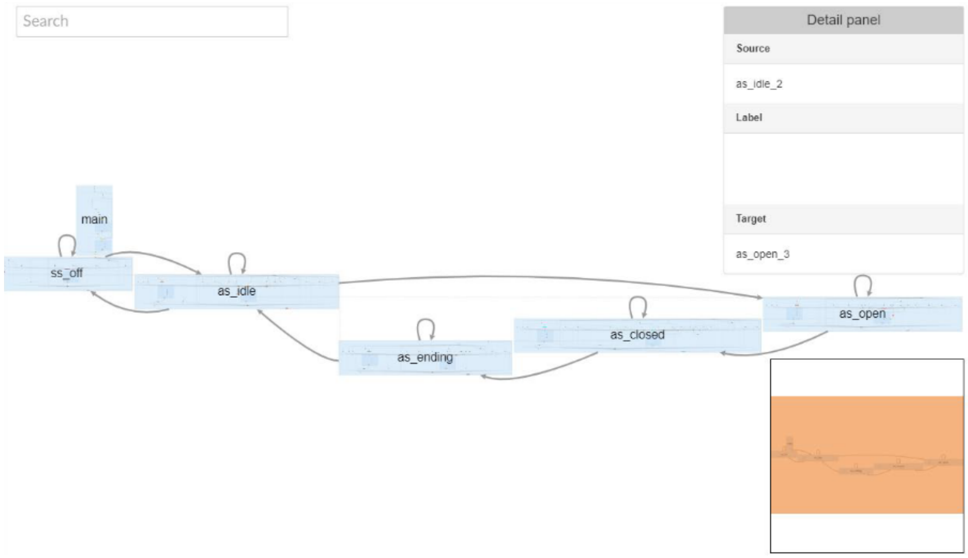


Figure 3.1: Image taken from [4], showing the overview of a model with semantic zooming.

Earlier work on visualising LTS models done at Axini uses semantic zooming. In [4], Werf defines semantic zooming as a form of details-on-demand that lets the user adjust the graph complexity by zooming in and out. This paper aimed to reduce the complexity of the visualisation by hiding irrelevant information. Zooming out on the generated graph will show a less detailed view of the LTS (illustrated in Figure 3.1), by blurring the nodes and edges and overwriting them with text on what that part does, but without changing the dimensions of the graph. So, if a user zoomed out, the graph complexity would be low; however, zooming back in would bring back the complexity.

Similarly, Luca *et al.* describe semantic zooming as a multi-level visualisation for exploring large graphs. In [22], they present a zoomable Multi-Level Tree algorithm for a map-like visualisation of large graphs. The multi-level visualisation utilises multi-level trees, and increasing the level of zoom shows the next tree.

3.3 Clustering and grouping

A common technique to reduce complexity in large graphs is to group nodes together based on structural or behavioural similarities [5, 6, 20, 21].

In [5], Ieperen describes clustering on actions and clustering on confluence. Clustering on actions means that paths from node to node that are identical are clustered together. This creates a smaller graph, although not entirely the same as the original in terms of the transitions. Clustering on confluence is clustering of nodes that represent internal behaviour of the system. These nodes get clustered together if their branching behaviour is identical. While both types of clustering aid in reducing the complexity of the graph, results show that attribute clustering may be preferred in some use cases, as it improves the readability of the graph and is easier to compute for large graphs [5].

Likewise, in a study investigating large state spaces, Groote and Ham [20] present a method to visualise very large state spaces with millions of nodes. To do so, they reduce the state space by clustering on an equivalence relation. This clustering method is taken from [6] and is similar to clustering on confluence because only when the paths down from separate nodes are identical, they may be clustered together [6, 20]. In a continuation of this study, Ploeger and Tankink argue that other methods of clustering should be explored [21]. With equivalence clustering, phases and branches of the graph are visible, but only after the LTS is generated. The LTS is generated breadth-first, and the clustering of the lower levels depends on the clustering of the higher levels [21].

Chapter 4

Design

This chapter describes the current visualisation of our use case and its limitations as specified by end-users and developers. We discuss the research process undertaken to inform the design decisions, followed by the final design choices made prior to implementing our solution. The primary objective of these designs is to at least contain the same amount of information as the original visualisation while presenting it in a less cluttered view.

4.1 Use case: Axini's models

Axini is a Dutch company that specialises in MBT to test software for other companies. For this, they developed the Axini Modelling Platform (AMP). Models are created in the Axini modelling language (AML) and they reflect the expected behaviours of the system under test (SUT). From those LTS/STS models, a visual abstract, read-only representation is generated.

The current design situation consists of a DAG with a force-directed layout. In Figure 4.1, a simple visualisation of an LTS depicting a coffee machine can be seen. This LTS portrays pressing the button for coffee on a coffee machine. This visualisation consists of one behaviour `main` with three states and a clear starting state. The starting state is a black dot, while the other states are rounded squares. The arrows connecting the states depict the transition it takes going from one state to another.

4.1.1 Limitations

The limitations are coming to light when visualising medium to large system models. Figure 4.2 is a medium-sized model, and because it is so big, it develops some uncertainty. This visualisation consists of two behaviours `session` and `subscription`, depicted as rectangles. Both behaviours have two nested behaviours, again shown as rectangles. This stacking of behaviours is visualised by the blue rectangles getting more saturated. As a consequence of the overlay, the light-grey dotted lines, which signal the connection between the last state in the nested behaviour to the first state of the other nested behaviour, are blending into the blue background.

In Figure 4.2, it can be seen that the directions of the arrows/edges are all morphed together, because of the horizontal layout of the states/labels. The first reaction of the user is drawn to those dark patches of crossed lines instead of the layout of the whole system. As this is the only view available, it is not very presentation-friendly. Furthermore, the amount of knowledge acquired by looking at and using this visualisation is limited to the user's prior knowledge of the system.

This suggests that the problem in using this visualisation lies in its inability to scale with the data and create a proper overview of the whole SUT.

4.2 Design criteria

In order to identify the alternative visualisations we want to research, we held interviews with clients and end-users who work regularly with AMP visualisations (the questionnaire can be found in appendix A.1). According to the interviewees, the visualisation tool was mainly used to present their models in such a way that they can be easily explained. Another use case resulting from the interview was to check if the visualised model turns out as expected. However, this only works if the model is small. When

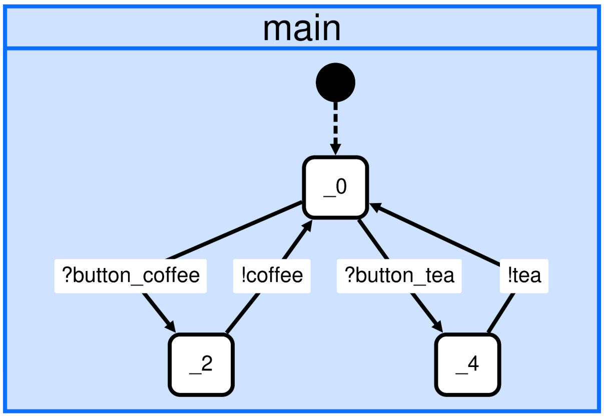


Figure 4.1: The AMP visualisation of the Coffee Machine LTS.

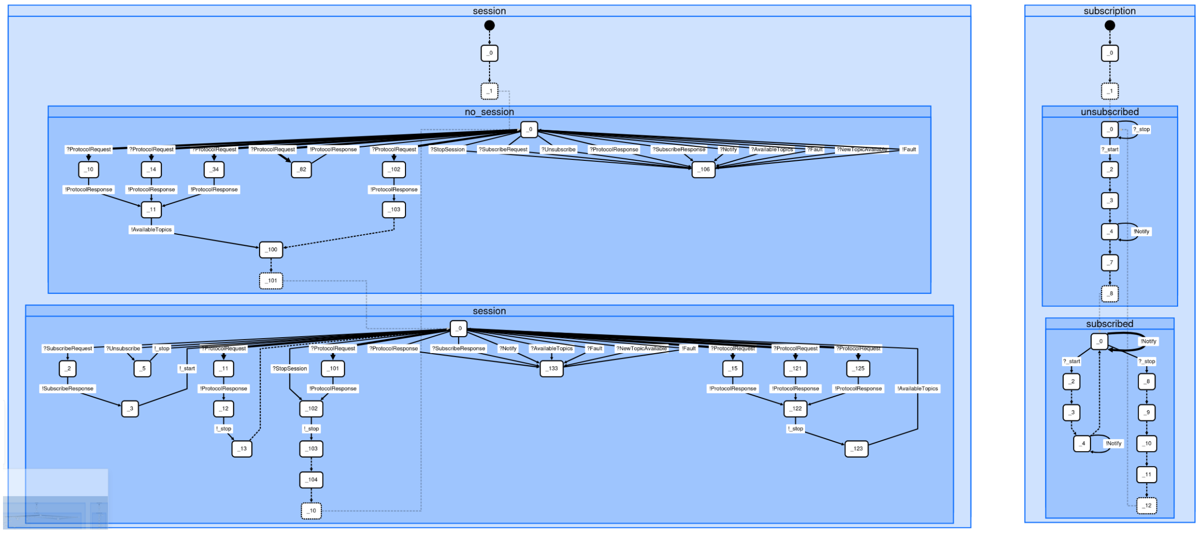


Figure 4.2: The AMP visualisation of the Newsfeed LTS.

the model scales up, it gets more difficult to check. Users with large models reported that they run into performance issues with the visualisation. For some models, it can take up to 30 seconds to load the visualisation, which has a negative impact on user experience. This also results in the visualisation being slower to respond to interactive elements. The full requirements can be found in Appendix A.2. The following criteria are the most important ones for our design:

1. **Flow**
   The flow of the visualisation should be clear. The structure of the visualisation should support nested nodes. Edges connecting nodes should be directed, and there should be a clear starting point. This allows the user to easily gather information about the flow of the visualisation.
2. **User interaction**
   The user should be able to interact with the visualisation to get an overview of all the depicted data. Such interactions are zooming in or out, clicking on nodes to display details, clicking on behaviours to expand or collapse them, and dragging nodes around for better presentation. To emphasise the flow of the visualisation, the path to a node should be highlighted when clicking on that node.
3. **Scalability**
   The visualisation method should allow users to navigate through both small and large datasets with equal ease.
4. **Aesthetics**
   The visualisation should look nice, such that all the criteria mentioned before are intuitive and

that the visualisation is readable and comprehensible. This will clarify the structure and meaning of the visualisation.

5. **Clarity**
   The focus of the design is clarity. This is achieved by a one-to-one mapping correspondence between symbols and the concepts they refer to, following the principle of semiotic clarity [18].
6. **Modularisation**
   Another focus is modularisation, defined by Moody as the reduction of the complexity of large systems into smaller parts or subsystems [18]. We will use this for expanding or collapsing nested behaviours and thus reducing the visual clutter of the visualisation.

4.3 Visualisation methods

For the visualisation methods, some fields of the Axini data were selected for use. According to Definition 1 of an LTS, we have visualised at least the following: node source representing Q, node target, T and a set of labels L. The following visualisation methods reflect the design criteria to a greater or lesser extent; simple versions are depicted in Figure 4.3. We will now explain the selected visualisation types based on the selected design criteria.

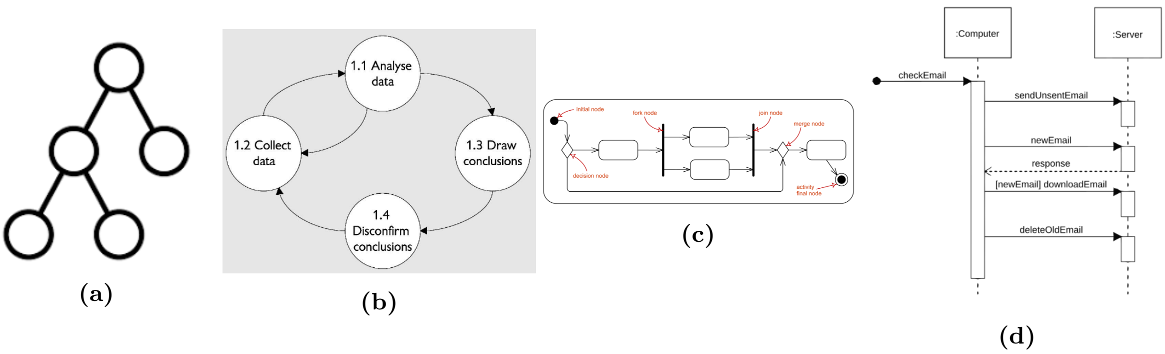


Figure 4.3: An overview of the types of visualisation methods selected for this study. a) tree diagram, b) data flow diagram, c) activity diagram, and d) sequence diagram.

4.3.1 Tree diagram

The first diagram type examines the impact of hierarchy in the visualisation. A tree diagram was used to analyse the relationship between the nodes. Tree diagrams have been widely used as a graphical model for probabilistic reasoning [23]. A tree diagram is a variation of a DAG with a hierarchical ordering. This type of visualisation should help to present an abstract overview of the data because hierarchy is one of the most effective ways of organising complexity [18].

The aesthetics of this visualisation method meet the bare minimum of this design criterion. As it is constructed of simple lines and shapes, it is readable and comprehensible, but not very pleasant to look at. Every node is connected to exactly one parent node, except the root node, which has no parent node. However, there are no ways to indicate connections between different parts of the system, and (self-)loops and multiple parents per node are not allowed in this layout structure. There is a way around those shortcomings, and that is considered in section 6.4.1.

The tree diagram supports user-driven exploration because the child nodes of every parent node can be expanded by clicking on the parent node. This way, a user can explore the diagram at their own pace and understanding.

For scalability, this diagram is generated via a JSON file, so the construction of the diagram will not increase the complexity. However, we hypothesise that navigating the diagram will become more complex when the dataset expands.

4.3.2 Data flow diagram

A data flow diagram (DFD) is an example of a diagram that shows the flow of data through a program by modelling the system as a network of transformations linked by paths of data [24]. It is one of the two most commonly used modelling techniques in computer science to visualise programs, the other one being ER (entity-relationship) modelling [18]. There are two variations with different shapes and arrow styles: the DeMarco dialect with rounded shapes and curved lines, and the Gane and Sarson dialect with rectangles with rounded corners and right-angled lines.

For this study, this visualisation method was chosen because of its strengths in showing the flow of a system. Furthermore, it is a useful diagram for deconstructing data into smaller parts, so for modularisation, we will be looking at this diagram. This type of diagram also promotes intuitive visualisation of the model. It provides a clear, organised overview at a glance.

4.3.3 UML activity diagram

A UML activity diagram models the dynamic aspects of a system or the flow of an object as it moves from state to state [25]. An activity diagram is similar to a state machine and may thus contain similar characteristics of state machines, like simple and composite states, branches, forks, and joins. The flow is emphasised by the directed edges and a clear starting point.

This visualisation method creates an abstraction of a model and has the ability to create multiple swim lanes to split a larger model into smaller subsystems, which promotes clarity and modularisation [26]. The core of a system remains relatively unchanged if it is rendered abstract, a process which is facilitated by this visualisation method. We hypothesise that even with large amounts of data, the system remains relatively uncluttered and thus scales well.

4.3.4 UML sequence diagram

A UML sequence diagram shows the interaction logic between the objects in the system in the order in which the interactions take place. Whereas activity diagrams emphasise the flow of control from activity to activity, interaction diagrams emphasise the flow of control from object to object [25]. This is useful for creating a clear flow in the visualisation and guides a beginner trying to understand the system. A sequence diagram is mostly used to visualise networking and message sending within computer science. This is visible in the aesthetics of the diagram: there is little variation in colour or shape, but the line types used are of significance. Solid lines represent requests and dotted lines represent responses. This helps the user differentiate the direction of the flow of the requests and responses. The elements used to construct our sequence diagram are taken from and explained in [27].

4.3.5 Semantic zooming method

For our novel visualisation method, called the semantic zooming method, we have combined all of our design criteria. This method creates a one-to-one mapping from the data of the Axini Newsfeed LTS. The concept is taken from an already established Cytoscape library called "expand-collapse". This code library uses semantic zooming, and by offering user-driven exploration, the view is interactive and adaptable for multiple use cases. For example: a developer looking for bugs, an experienced user presenting the model to others, or a beginner exploring the model.

Initially, the user is presented with a detailed, zoomed-out view of the system as an overview. The user can then zoom in and filter out areas of interest. Further details can be accessed by expanding or collapsing sections as required. This presentation, with an overview at first glance, followed by details on demand, is the basis of semantic zooming.

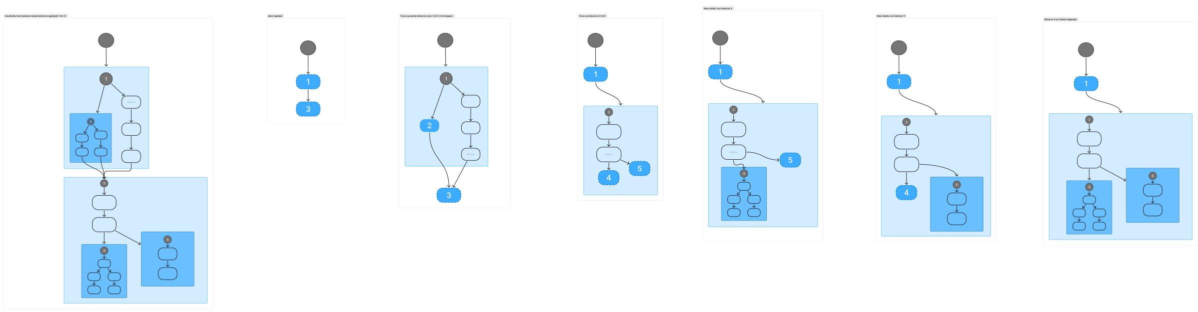


Figure 4.4: Draft of visualisation with expanding and collapsing behaviours.

We included glyphs in the visualisation: within a collapsed node, a bar chart and a pie chart are visible, describing the collapsed behaviour. This way, the user has access to information about nodes that are hidden from view.

We matched the aesthetics of the original situation by keeping the colour, shape and size of the nodes the same. Figure 4.4 presents a prototype of the collapsible nodes.

We expect that in this way, the visual clutter of a big, complex program can be reduced.

Chapter 5

Results

In this chapter, we present the results of our visualisation methods and the user survey we sent out. Together, they provide important insights into the use cases of the visualisation.

5.1 Visualisation methods

Depicted below are the resulting diagrams of the selected visualisation methods. They were constructed with the D3, MermaidJS, and Cytoscape libraries available for JavaScript.

Tree diagram

Figure 5.1 shows the breakdown of the Newsfeed LTS according to the rules of a tree diagram. The diagram is expandable and starts small and collapsed. In Figure 5.2, a part of the tree is still collapsed. As explained before, this diagram type examines the effect that hierarchy has on the model, which is reflected in the figure by the single starting point that splits into two child nodes of a lower rank. One of the shortcomings of the diagram, i.e. no ways to indicate connections between different parts of the system, is illustrated in Figure 5.1 and 5.2. To demonstrate the consequence, we have added dashed red lines to connect nodes across the system, spanning multiple ranks.

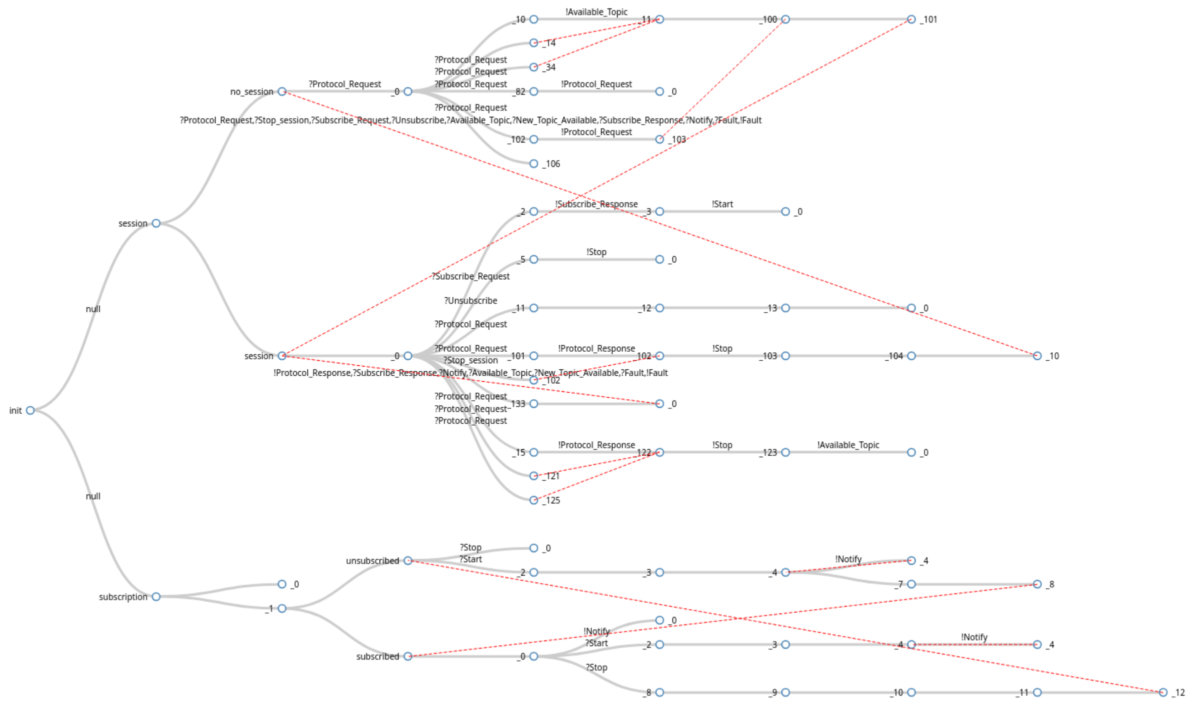


Figure 5.1: Tree diagram of the Newsfeed LTS.

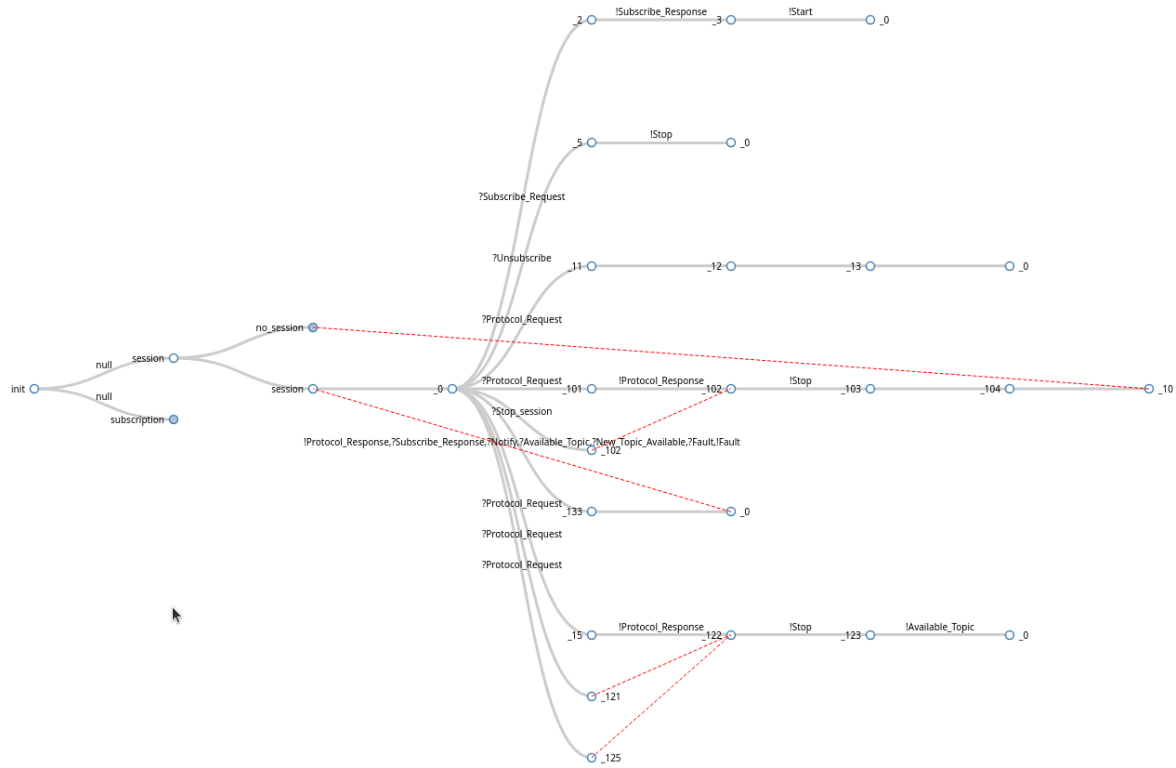


Figure 5.2: The tree diagram, partially extended, so that only one of the flows is visible.

Data flow diagram

The second diagram type examined the modularisation of the system. The data flow diagram shows how data flows through the Newsfeed LTS, see Figure 5.3. To show the modularisation, the system is divided into different processes, depicted by circles in the DeMarco dialect. The databases in the diagram are not the shape of the DeMarco dialect, but rather the database shape of UML.

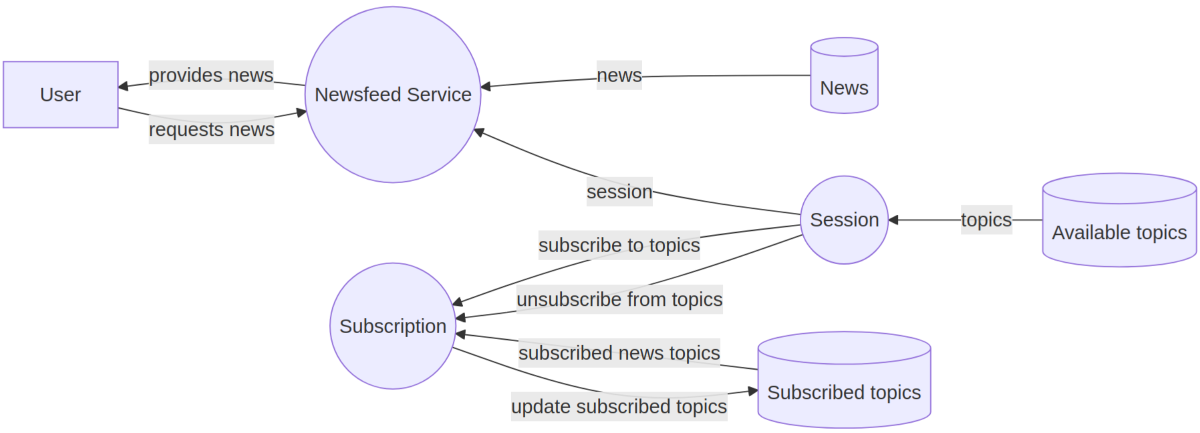


Figure 5.3: Data flow diagram of the Newsfeed LTS.

UML activity diagram

The third diagram type examined an abstracted version of the system. The activity diagram in Figure 5.4 gives a high-level overview of the actions of the Newsfeed LTS. The diamond shapes represent a decision-making point for the system.

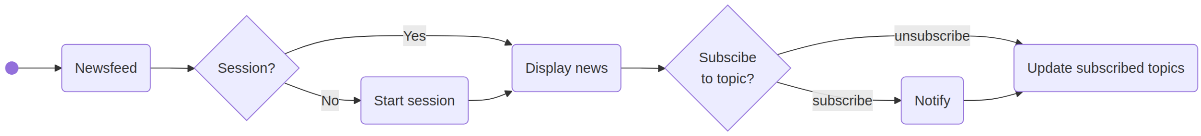


Figure 5.4: The activity diagram of the Newsfeed LTS.

UML sequence diagram

The last diagram type examines the logic and the flow of a system. The sequence diagram in Figure 5.5 shows the Newsfeed LTS. The user is shown as an external factor to the system, and the behaviours are turned into processes. The dotted boxes labelled `alt` depict the choice for the session and the subscription process. The user is included in the subscription process.

Semantic zooming method

Figure 5.6 shows the implementation of semantic zooming on the Newsfeed LTS and shows the system on start-up. The default is the detailed view, with every behaviour expanded. There are a few buttons on the sidebar. The explore button collapses all the nodes and shows an abstract overview of the system, as can be seen in Figure 5.7. The user is able to dive into detail by expanding all the behaviours as they work their way through the visualisation. Figure 5.8 shows a partly expanded view. The highlight trace button enables a mode where clicking on a node in the graph, the path to that node is accentuated (Figure 5.9). Lastly, the sequence diagram button and the activity diagram button show the same picture as in Figure 5.5 and 5.4, respectively.

5.2 Survey

5.2.1 Set up of the survey

A survey questioning the design criteria of all the visualisation methods was sent out to expert users connected to Axini, developers who designed the current situation, and to students of the MSc Software Engineering, who in their studies have come across an assignment working with the visualisation of Axini. The participants were asked to give feedback on the visualisation methods and how they reflect the design criteria described in section 4.2. Additionally, participants were asked which visualisation method they would employ for different use-cases. The questions can be found in Appendix A.3.

5.2.2 Results of the survey

Thirteen individuals returned the questionnaire. The following results were obtained for each of the specified use cases: 76.9% of the respondents voted for the semantic zooming method to use for debugging, while 15.4% opted for the original and 7.7% chose the tree diagram.

With regard to the exploration of the models, the semantic zooming method was selected by 38.5% of respondents, the data flow diagram and the activity diagram were each selected by 23.1%, and both the original situation and the sequence diagram were selected by 7.7%.

Concerning the presentation of models, the sequence diagram and the activity diagram got equal votes with 30.8%, the semantic zooming method and the data flow diagram followed with both 15.4%, and 7.7% voted for the original situation.

In Figure 5.10, the results from the survey are presented per design criterion. Table 5.1 summarises the results in a heatmap. In the following section, we will highlight the most notable results for each design criterion.

Flow

Respondents agree that there is a clear flow in every visualisation method. What stands out is that the activity diagram and the semantic zooming methods both have no negative responses.

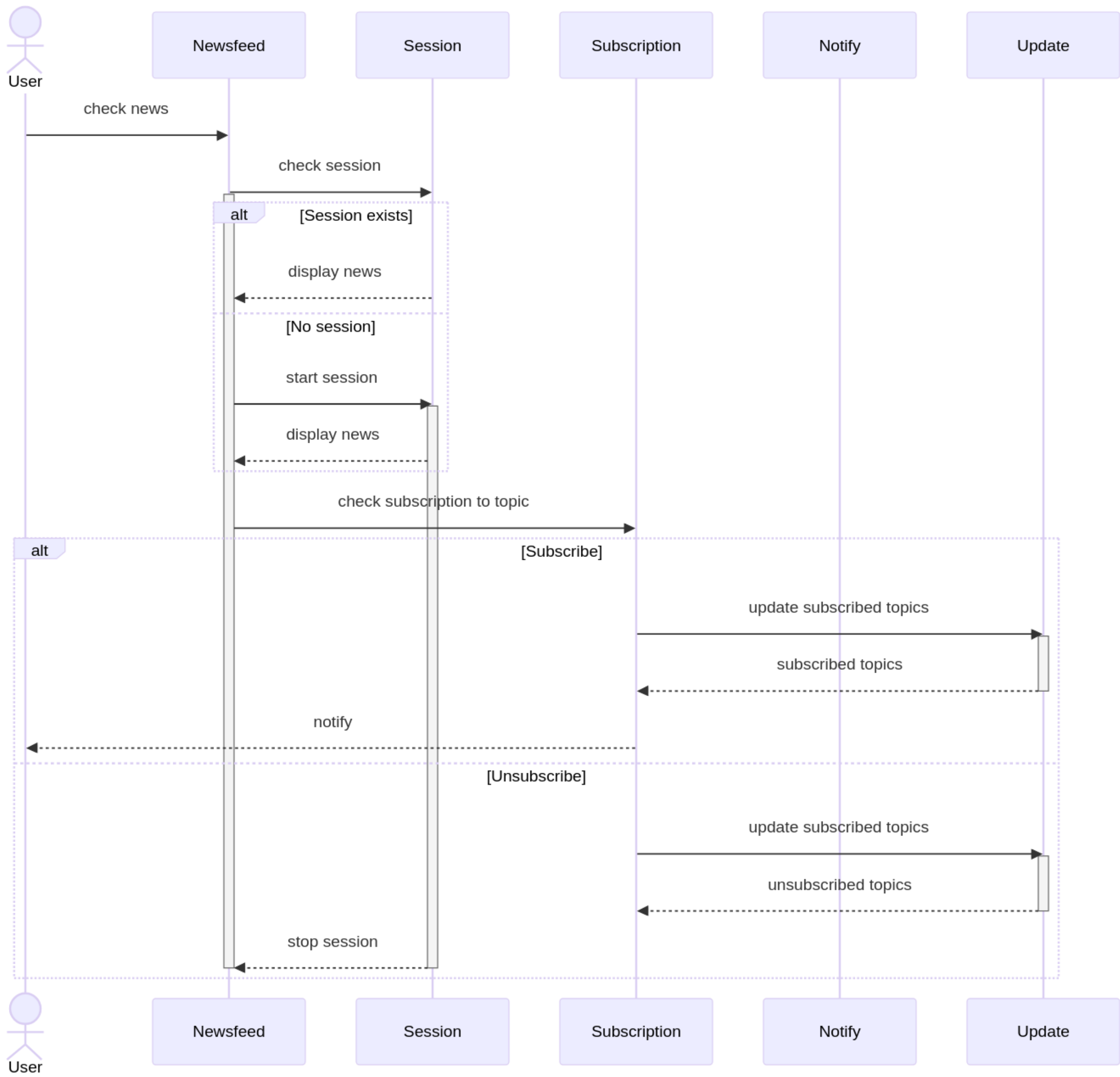


Figure 5.5: The sequence diagram of the Newsfeed LTS.

User interaction

Respondents are overall indifferent about the user interactivity of the visualisation methods. Only a minority of respondents are negative about the user interaction. Interestingly, the survey reported an overwhelmingly positive response to the interactivity of the semantic zooming visualisation method.

Scalability

The survey results for this design criterion report a mix of results. Responses for the original situation, tree diagram, and sequence diagram are quite negative-leaning. The other visualisation methods have more positive responses.

Aesthetics

Respondents are mostly indifferent and slightly positive about the aesthetics of the various diagrams. The activity diagram visualisation method has the most positive votes, with only a minority being indifferent. The original situation has received the most negative and indifferent votes.

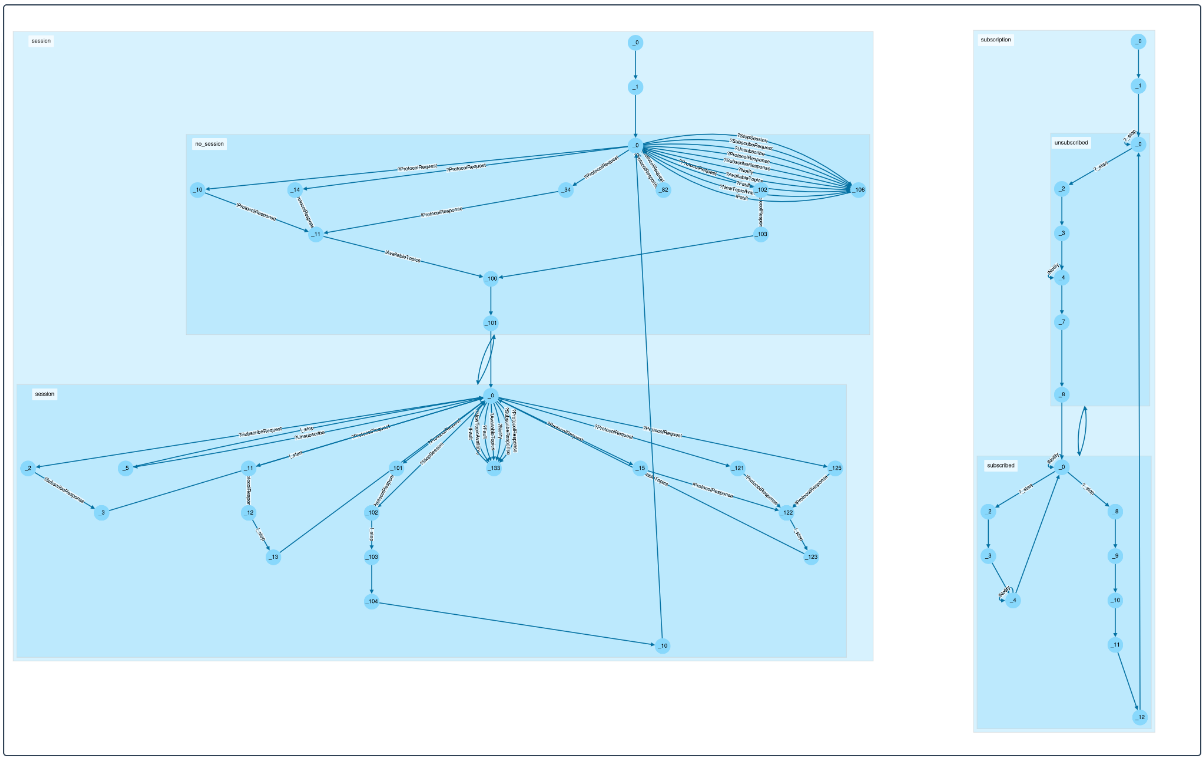


Figure 5.6: The visualisation method with semantic zooming, depicting the whole model of Newsfeed LTS.

Figure 5.7: The most zoomed out view of the LTS.

Clarity

Clarity in this survey is defined as a one-to-one mapping correspondence between symbols and the concepts they refer to. Respondents show a mix of results and are mostly negative about the activity diagram and the dataflow diagram.

Modularisation

Opinions differ as to whether the visualisation methods were capable of modularisation. Taking a closer look at Figure 5.10f, it can be seen that respondents were positive about the data flow diagram, activity diagram and the semantic zooming, but less so about the original situation. The tree and sequence diagrams received both positive and negative feedback.

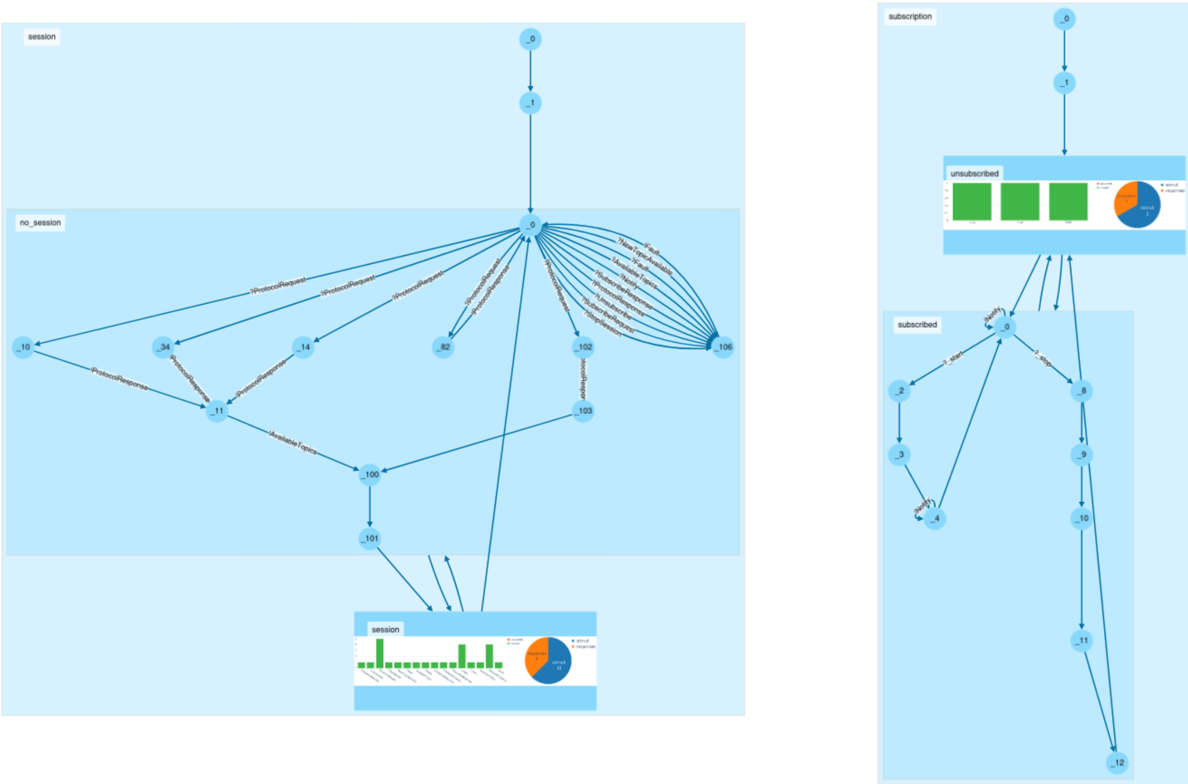


Figure 5.8: The visualisation method with semantic zooming, depicting the model of Newsfeed LTS with some behaviours collapsed.

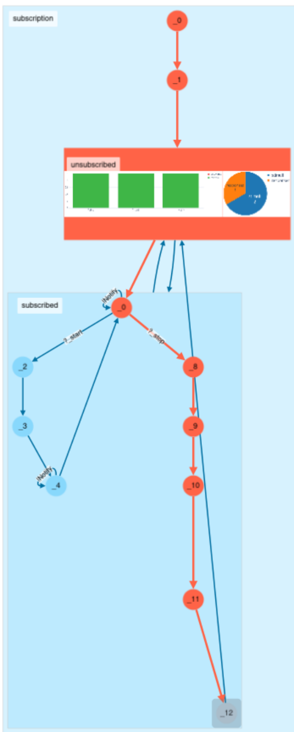
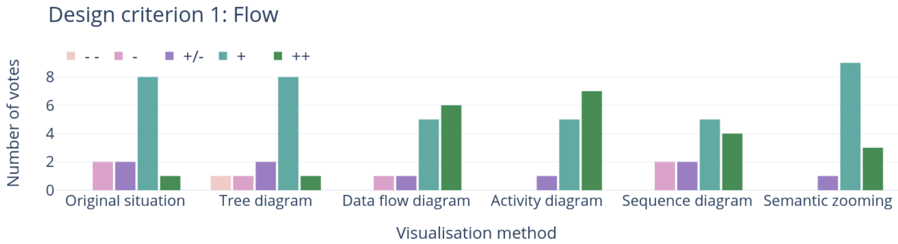
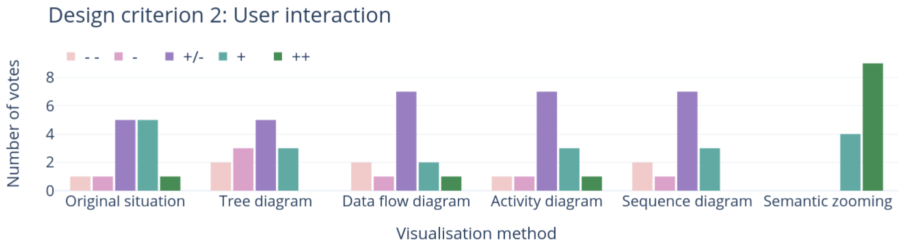


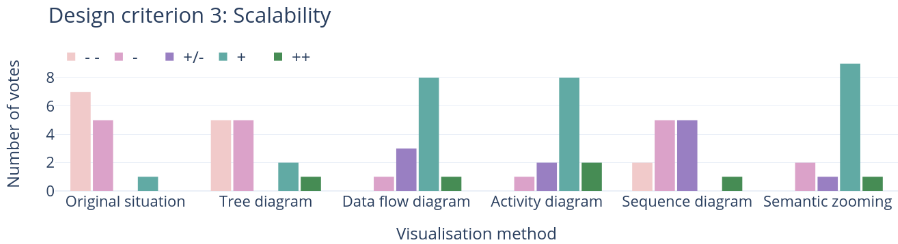
Figure 5.9: The trace to a node is highlighted.



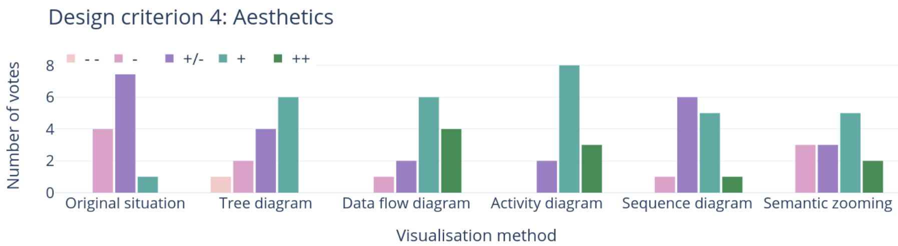
(a) Flow design criterion



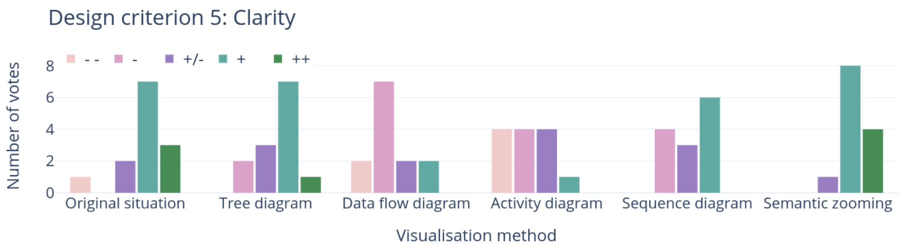
(b) User interaction design criterion



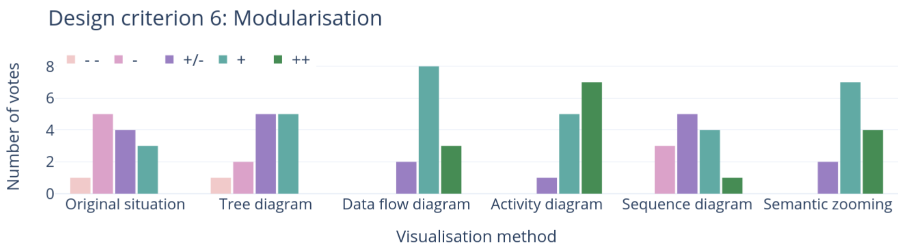
(c) Scalability design criterion



(d) Aesthetics design criterion



(e) Clarity design criterion



(f) Modularisation design criterion

Figure 5.10: Results from the survey.

	original situation	tree diagram	data flow diagram	activity diagram	sequence diagram	semantic zooming method
flow	47	46	55	58	50	54
user interaction	43	35	38	41	37	61
scalability	21	28	48	50	32	48
aesthetics	36	41	52	53	45	45
clarity	50	46	30	28	41	55
modularisation	35	40	53	58	42	54
total score	**232**	**236**	**276**	**288**	**247**	**317**

Table 5.1: Heatmap of survey results. The rating scale was converted to a score of 1-5. Scores were added and mapped to colour values.

Chapter 6

Discussion

In this chapter, we discuss the results of our experiment(s) concerning our research questions. The results of the previous chapter indicate that not one of the alternatives has a clear preference on all the design criteria among the respondents of the survey. This chapter, therefore, moves on to discuss the reasons why this could be.

6.1 Scalability, clarity and user experience

The first sub-question in this study sought to determine in what ways a novel visualisation approach for LTSs outperforms traditional graph-based visualisation in terms of scalability, clarity and overall user experience.

Looking at scalability, the results from the survey and Table 5.1 show that the data flow diagram, activity diagram, and the semantic zooming method are the preferred visualisation methods for an LTS visualisation, which does not get harder to navigate when depicting a larger data set. One potential explanation for this result is that these methods present an abstraction of the system, as is also the case with data flow and activity diagrams. Alternatively, they may possess the capability to simplify the model, as is achieved through the utilisation of the semantic zooming method. This results in reduced visual clutter on the screen, a goal previously identified by studies as being essential for an effective visualisation.

For clarity, the results showed a mix of opinions, with the activity diagram, sequence diagram and the data flow diagram ending up at the bottom. This outcome may be attributed to the fact that, when assessing clarity, it is essential to ensure a one-to-one mapping between all elements of the system and the visualisation of the model. In the lowest-scoring visualisation methods, this mapping might be incomplete. We reason that this is because the rules of the visualisation method do not allow for a perfect mapping, and that during the construction of the visualisations, some elements of the model got lost in translation. In contrast, the reason why the other diagrams, including the original situation, are well-received on this criterion might be because they are generated directly from the data provided by Axini. Thus, they do not lose any information while mapping from the model to the visualisation.

The user engagement results state that respondents of the survey were indifferent about the user interaction of the different visualisation methods. It seems possible that these results are due to the fact that only half of the visualisation methods supported user interaction. The very positive result for the semantic zooming method is expected because this visualisation method has more interactive elements than any of the other visualisation methods.

> **Finding 1:** Novel visualisation approaches outperform traditional methods in terms of clarity and user engagement. However, in terms of scalability, they score low, as information is lost during the process of creating a mapping from the model to the visualisation.

6.2 Usability, performance and expressiveness

The second sub-question in this study sought to determine how the inclusion of interactive features impacts the usability, performance, and expressiveness of the LTS visualisation tools for expert and new users.

The results indicate that visualisation methods with interactive features were more likely to be chosen for tasks such as debugging and exploring models. It can therefore be assumed that interactive features do impact the usability and expressiveness of the LTS visualisation tools in a positive way. However, with a small sample size, caution must be applied, as the findings might not be accurate for new users. A possible explanation for this might be that the survey was mainly filled out by experts and less so by new users, which introduces a bias into the results.

Performance was not tested extensively; however, we hypothesise that some interactive features, such as animations, clickable pop-up elements, could affect the performance.

Finding 2: Visualisation methods with interactive features impact the usability and expressiveness of the visualisation in a positive way.

6.3 Answer to main questions

This thesis set out to find whether alternative visualisation types enhance the comprehension and analysis of complex LTSs for users. We found that alternative visualisation types can help the users comprehend and analyse complex LTS models because they provide an abstraction that reduces the visual clutter that complex systems often produce. This is supported by finding 1.

By developing the semantic zooming visualisation method, our work has also shown that interactive features and user-driven exploration can be incorporated into LTSs visualisations to support model checking and analysis more effectively. This is further supported by finding 2.

Although this study is based on a small sample of respondents in the survey, the findings suggest that using alternative visualisation methods together with interactive features is a feasible option to reduce visual clutter in complex visualisations and improve user engagement.

6.4 Threats to validity

To conclude this chapter, we will now examine potential threats to the validity of this thesis. We have identified two points of concern: the limited exploration of ways to adapt the D3 tree diagram, and the possible bias in the survey responses.

6.4.1 Tree diagram

In our tree diagram, we limited the number of parents per node to one, as is the standard for the D3 tree diagram. Multiple parents per node are possible when adapting the D3 source code or changing to another tree diagram generator altogether. This would change the layout of Figure 5.1, making the dashed red lines redundant and improving the readability, aesthetics, and comprehensibility.

6.4.2 Survey responses

The survey went out to a mixed population. On the one hand, there were expert users connected to Axini and developers who designed the current situation. On the other hand, there were MSc Software Engineering students who had some experience with the visualisation of Axini, but were by no means expert users. We got a total of 13 responses, of which we do not know how many were experts or novices. Without the respondents' background, these results need to be interpreted with caution, and it is important to bear in mind the possible bias in these responses.

At the time of sending out the survey, the semantic zooming method was in an early state of development. This may have influenced the appreciation of the method. Especially, the aesthetics were not refined yet. Should the respondents have been able to rate the final version, the score in Table 5.1 for aesthetics would likely have been higher, resulting in an even higher total score.

Chapter 7

Conclusion

In conclusion, this project successfully achieved the creation of multiple alternative visualisations for Labelled Transition Systems (LTS) and the creation of an interactive semantic zooming component to run on an already established dataset.

We have shown that LTSs suffer from *state space explosion* and have established six design criteria that should be used to select the alternative visualisation methods to help clean up the visual clutter.

With our use case from Axini, we developed the semantic zooming method, which combines a directed acyclic graph with a force-directed layout algorithm and interactive functionalities, such as glyphs, that make the graph more scalable, modular, clear and aesthetically pleasing.

Several key findings have been obtained. We found that novel visualisation approaches outperform traditional methods in terms of clarity and user engagement. Furthermore, visualisation methods with interactive features positively affect the usability and expressiveness of the visualisation. Taken together, these findings suggest that adding interactive features to visualisations of LTSs helps to reduce visual clutter.

7.1 Future work

As the size of network data sources is expected to gradually increase over time, visualisations of LTSs will supposedly continue to exhibit signs of visual clutter. Further research is needed to find a way to minimise this. We propose an experimental investigation into the impact of incorporating glyphs into LTS visualisations. For example, researching how glyphs together with the visualisation of LTSs may further reduce visual clutter. To develop a full picture of this, additional studies will be needed to assess how much additional information may be added to the visualisation using glyphs.

Before the semantic zoom visualisation method can be implemented in the Axini model, a better understanding of the user value needs to be collected. We suggest repeating the survey with the latest version of the model, after adding a question about the background of the respondent. Also, to increase the response to the questionnaire, we suggest arranging funding for an incentive. In this way, with maximal response from respondents with different backgrounds, the collection of a dataset of a higher quality will allow the drawing of a more widely supported conclusion regarding the esteem for and usability of the semantic zoom model.

An alternative way to validate the semantic zoom visualisation method before implementation is to encourage various users and developers to utilise the model for their respective datasets, in combination with schematic observation of their utilisation. Such an observational study is expected to provide significant insights into the practical challenges associated with implementation.

Bibliography

[1] J. Tretmans, "Model based testing with labelled transition systems," in *Formal Methods and Testing: An Outcome of the FORTEST Network, Revised Selected Papers*, R. M. Hierons, J. P. Bowen, and M. Harman, Eds., Berlin, Heidelberg: Springer, 2008, pp. 1–38, ISBN: 978-3-540-78917-8. DOI: 10.1007/978-3-540-78917-8_1. [Online]. Available: https://doi.org/10.1007/978-3-540-78917-8_1 (visited on 01/06/2025).

[2] J. Tretmans, "On the existence of practical testers," in *ModelEd, TestEd, TrustEd: Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*, J.-P. Katoen, R. Langerak, and A. Rensink, Eds., Cham: Springer International Publishing, 2017, pp. 87–106, ISBN: 978-3-319-68270-9. DOI: 10.1007/978-3-319-68270-9_5. [Online]. Available: https://doi.org/10.1007/978-3-319-68270-9_5 (visited on 01/08/2025).

[3] P. van den Bos and J. Tretmans, "Coverage-based testing with symbolic transition systems," in *Tests and Proofs*, D. Beyer and C. Keller, Eds., Cham: Springer International Publishing, 2019, pp. 64–82, ISBN: 978-3-030-31157-5. DOI: 10.1007/978-3-030-31157-5_5.

[4] D. van der Werf, "Visualizing symbolic transition systems," Master Thesis, University of Amsterdam, 2018, 44 pp.

[5] G. van Ieperen, "Visualisation of large labelled transition systems," Apr. 23, 2021.

[6] F. van Ham, H. van de Wetering, and J. van Wijk, "Interactive visualization of state transition systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 4, pp. 319–329, Oct. 2002, Number: 4 Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2002.1044518. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1044518 (visited on 01/07/2025).

[7] Y. Hu and L. Shi, "Visualizing large graphs," *WIREs Computational Statistics*, vol. 7, no. 2, pp. 115–136, 2015, Number: 2 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1343, ISSN: 1939-0068. DOI: 10.1002/wics.1343. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1343 (visited on 01/14/2025).

[8] L. Frantzen, J. Tretmans, and T. A. C. Willemse, "Test generation based on symbolic specifications," in *Formal Approaches to Software Testing*, J. Grabowski and B. Nielsen, Eds., Berlin, Heidelberg: Springer, 2005, pp. 1–15, ISBN: 978-3-540-31848-4. DOI: 10.1007/978-3-540-31848-4_1.

[9] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings 1996 IEEE Symposium on Visual Languages*, ISSN: 1049-2615, Sep. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307. [Online]. Available: https://ieeexplore.ieee.org/document/545307 (visited on 01/10/2025).

[10] Card, Stuart K, Mackinlay, Jock, and Shneiderman, Ben, *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[11] J.-D. Fekete, J. J. Van Wijk, J. T. Stasko, and C. North, "The value of information visualization," in *Information Visualization*, A. Kerren, J. T. Stasko, J.-D. Fekete, and C. North, Eds., red. by D. Hutchison *et al.*, vol. 4950, Series Editors: _:n187 Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–18, ISBN: 978-3-540-70955-8 978-3-540-70956-5. DOI: 10.1007/978-3-540-70956-5_1. [Online]. Available: http://link.springer.com/10.1007/978-3-540-70956-5_1 (visited on 01/10/2025).

[12] H. Shen *et al.*, "Information visualisation methods and techniques: State-of-the-art and future directions," *Journal of Industrial Information Integration*, vol. 16, p. 100 102, Dec. 1, 2019, ISSN: 2452-414X. DOI: 10.1016/j.jii.2019.07.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X19300573 (visited on 01/10/2025).

[13] J. van Wijk, "The value of visualization," in *VIS 05. IEEE Visualization, 2005.*, Oct. 2005, pp. 79–86. DOI: 10.1109/VISUAL.2005.1532781. [Online]. Available: https://ieeexplore.ieee.org/document/1532781/?arnumber=1532781 (visited on 01/07/2025).

[14] J. H. Larkin and H. A. Simon, "Why a diagram is (sometimes) worth ten thousand words," *Cognitive Science*, vol. 11, no. 1, pp. 65–100, Jan. 3, 1987, Number: 1 Publisher: John Wiley & Sons, Ltd, ISSN: 0364-0213. DOI: 10.1111/j.1551-6708.1987.tb00863.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1551-6708.1987.tb00863.x (visited on 01/08/2025).

[15] Bennett, Chris, Ryall, Jody, Spalteholz, Leo, and Gooch, Amy, "The aesthetics of graph visualization," in *ResearchGate*, The Eurographics Association, 2007, ISBN: 978-3-905673-43-2. DOI: 10.2312/COMPAESTH/COMPAESTH07/057-064. [Online]. Available: https://www.researchgate.net/publication/220795329_The_Aesthetics_of_Graph_Visualization (visited on 04/22/2025).

[16] J. Heer, M. Bostock, and V. Ogievetsky, "A tour through the visualization zoo," *Communications of the ACM*, vol. 53, no. 6, pp. 59–67, Jun. 2010, Number: 6, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1743546.1743567. [Online]. Available: https://dl.acm.org/doi/10.1145/1743546.1743567 (visited on 04/01/2025).

[17] M. Petre, "Why looking isn't always seeing: Readership skills and graphical programming," *Commun. ACM*, vol. 38, no. 6, pp. 33–44, Jun. 1, 1995, Number: 6, ISSN: 0001-0782. DOI: 10.1145/203241.203251. [Online]. Available: https://dl.acm.org/doi/10.1145/203241.203251 (visited on 01/10/2025).

[18] D. Moody, "The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 756–779, Nov. 2009, Number: 6 Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.2009.67. [Online]. Available: https://ieeexplore.ieee.org/document/5353439 (visited on 01/08/2025).

[19] R. Borgo *et al.*, "Glyph-based visualization: Foundations, design guidelines, techniques and applications," 2013, Publisher: The Eurographics Association, ISSN: 1017-4656. [Online]. Available: https://doi.org/10.2312/conf/EG2013/stars/039-063 (visited on 06/04/2025).

[20] J. F. Groote and F. van Ham, "Interactive visualization of large state spaces," *International Journal on Software Tools for Technology Transfer*, vol. 8, no. 1, pp. 77–91, Feb. 1, 2006, Number: 1, ISSN: 1433-2787. DOI: 10.1007/s10009-005-0198-5. [Online]. Available: https://doi.org/10.1007/s10009-005-0198-5 (visited on 01/08/2025).

[21] B. Ploeger and C. Tankink, "Improving an interactive visualization of transition systems," in *Proceedings of the 4th ACM symposium on Software visualization*, ser. ACM Digital Library, New York, NY: ACM, 2008, p. 11, ISBN: 978-1-60558-112-5.

[22] F. D. Luca, I. Hossain, K. Gray, S. Kobourov, and K. Börner, *Multi-level tree based approach for interactive graph visualization with semantic zoom*, Issue: arXiv:1906.05996, Dec. 9, 2019. DOI: 10.48550/arXiv.1906.05996. arXiv: 1906.05996[cs]. [Online]. Available: http://arxiv.org/abs/1906.05996 (visited on 01/14/2025).

[23] S. Dasgupta, "Learning polytrees," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, ser. UAI'99, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 30, 1999, pp. 134–141, ISBN: 978-1-55860-614-2. (visited on 06/02/2025).

[24] P. T. Ward, "The transformation schema: An extension of the data flow diagram to represent control and timing," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 2, pp. 198–210, Feb. 1986, Number: 2 Conference Name: IEEE Transactions on Software Engineering, ISSN: 1939-3520. DOI: 10.1109/TSE.1986.6312936. [Online]. Available: https://ieeexplore.ieee.org/document/6312936/?arnumber=6312936 (visited on 01/27/2025).

[25] G. Booch, J. Rumbaugh, and I. Jacobson, "Unified modeling language user guide, the (2nd edition) (addison-wesley object technology series)," *J. Database Manag.*, vol. 10, Jan. 1, 1999, ISSN: 0321267974.

[26] K. H. Han, S. K. Yoo, and B. Kim, “Qualitative and quantitative analysis of workflows based on the UML activity diagram and petri net,” Jul. 2009.

[27] Donald Bell. “Explore the UML sequence diagram,” Explore the UML sequence diagram. (Oct. 8, 2023), [Online]. Available: `https://developer.ibm.com/articles/the-sequence-diagram/` (visited on 10/06/2025).

Appendix A

Supplementary data

A.1 Questionnaire

1. What do you use the visualisation function of AMP for?
2. Should the visualisation be interactive? If yes, what kind of interactions would you like to have in the visualisations? (currently: zooming in/out, and clicking on nodes)
3. Are there any problems you run into when using the visualisation? If so, what kind of problems?
4. If you have used the visualisation but don't use it any more, why did you stop using it?

A.2 Requirements

Data requirements:

1. The visualisation should support a hierarchical structure (i.e. files in folders/ sub processes)
2. The visualisation should depict relationships in the data.
3. The visualisation should follow basic principles:
   (a) The edge crossings should be minimised.
   (b) The nodes should be close to neighbours.
   (c) Clusters of nodes should be far enough apart.
   (d) Symmetry should be highlighted.

User requirements:

1. The user should get an overview of the whole visualisation.
2. The user should be able to interact with the visualisation:
   (a) The user should be able to zoom in/out.
   (b) The user should be able to click on nodes to display the details of that node.
   (c) The user should be able to drag nodes around for better presentation
3. The user should be able to gather information about the flow of the visualisation.
4. The visualisation parts should scale accordingly.
5. The functions / behaviour related nodes should be collapsible.

Design requirements:

1. The visualisation should have a clear start and end point(s).
2. The edges should have labels.
   (a) The flow of the visualisation should be clear.
   (b) When clicking on a node, the flow to and from should be highlighted.
   (c) The edges connecting the nodes should be directed.
   (d) After running tests, the completed trace should be highlighted.

A.3 Survey

Which of the above visualisations has your preference for debugging models?

- original situation
- tree diagram
- data flow diagram
- activity diagram
- sequence diagram
- semantic zooming method

Which of the above visualisations has your preference for exploring/getting to know models?

- original situation
- tree diagram
- data flow diagram
- activity diagram
- sequence diagram
- semantic zooming method

Which of the above visualisations has your preference for presenting models?

- original situation
- tree diagram
- data flow diagram
- activity diagram
- sequence diagram
- semantic zooming method

Consider the original situation	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					

Consider the tree diagram	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					

Consider the data flow diagram	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					

Consider the activity diagram	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					

Consider the sequence diagram	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					

Consider the semantic zooming method	- -	-	+/-	+	+ +
This diagram has a clear flow					
This diagram has user interactions					
This diagram is scalable without becoming more complex					
This diagram looks nice					
This diagram includes all elements of the system					
This diagram reduces the complexity of the system					