

Using LLMs for obtaining workflow representations in ATWL

1 Experiments with LLMs

1.1 Creating Initial Examples Using an LLM

For obtaining initial examples, we used an existing representation of the Cluster-Calendar workflow [14], which was written manually at an earlier stage of the language development and used an obsolete version of the language syntax. With the help of an LLM (GPT 5.1), the representation was transformed to the actual syntax definition.

Then we used different, independent sessions with GPT5.1 to produce two additional example ATWL representations of the workflows described in published papers. In each session, the LLM was provided with the text of the ATWL specification, the earlier created Cluster-Calendar example, a PDF of the paper describing a workflow, and a request to generate a workflow summary in a natural language and a representation in ATWL.

After receiving a response, it was often necessary to give additional prompts for improving the result. In particular, we requested the LLM to omit excessive technical details and increase the level of abstraction in workflow representation. Another request was for simplification of representations of iteratively repeated workflow parts.

After making corrections, the workflow representations (including the Cluster-Calendar example) were reviewed by another LLM agent (Claude Sonnet 4.5R), which was provided with the ATWL specification text, the paper PDF, the natural language workflow summary, and the ATWL representation. The LLM agent was requested to check the representation for alignment with the paper content, syntactic and logical correctness, and possibilities for simplification. The agent did detect several issues and proposed corrections. The reports were checked by a human expert, and recommended corrections implemented.

1.2 Designing the Subsequent Experiment Procedure

As insufficient abstraction level turned to be a recurring problem, the LLM agent that reviewed the representations (Claude Sonnet 4.5R) was requested to formulate the guidelines for an appropriate level of abstraction in ATWL representations, which resulted in the text attached as Appendix B.

In the following experiments, we used the full specification of ATWL, the abstraction guidelines, and the examples representing the Cluster-Calendar and the EventFlow workflows to prompt GPT5.1 and Claude Sonnet 4.5R before giving them the task to extract the workflow description from a paper on dynamic network exploration [13] and represent them in ATWL. The results of Claude Sonnet 4.5R were superior to those of GPT5.1, which motivated us to use the former LLM in the subsequent experiments.

We then used a different, independent instance of Claude Sonnet 4.5R to review the ATWL representation generated. The two LLM instances can be referred to as "formalising agent" and "reviewing agent", or Formaliser and Reviewer, respectively. Reviewer detected minor occasional violations of the standard ATWL syntax (inclusion of additional fields such as 'derived-from') and inclusion of technical details such as algorithm names. The critical comments were communicated to Formaliser, which produced an improved variant of the workflow representation, which is included in Appendix C.2.

1.3 Conducting the Experiments

In the subsequent experiments, we consistently used the two-agent approach providing feedback from Reviewer to Formaliser, so that Formaliser could further learn and improve its representation skills.

In our communication with Reviewer, we sometimes provided corrective prompts. In response to Reviewer's comment "Problem: The template shows only one visualise step, but the workflow has two visualise transforms":

The workflow template does not need to include all transforms. It should be a summary, where it is quite valid to join several transforms with the same intent, especially if they can be done in parallel and independently of each other.

In response to Reviewer's comment "Critical Issue: Visualization Scoping Problem. Visualizations declared inside temporal loop are referenced outside it.":

The identified scoping issue is not problematic. ATWL is not a programming language, and any artifacts created within a loop continue to exist after exiting the loop. Therefore, it is fine to reference such artifacts in the statements following the loop end.

More complex situations also occurred. Thus, the Formaliser failed to correctly represent the logic and structure of the quite complex workflow for progressive clustering of trajectories [12]. After several steps of reviewing, passing Reviewer's feedback to Formaliser, and sending revised versions back to Reviewer, the latter finally found a suitable approach to representing this workflow. The Reviewer's output describing and presenting the approach is included as Appendix A. It demonstrates a high level of the LLM's reasoning capability.

However, the experimenters found that the proposed approach makes the ATWL representation of the workflow too specific to the study described in the paper. They provided the following feedback to Reviewer:

The resulting workflow representation appears to be very specific to the given dataset and analysis with its unique findings. It will be problematic to compare it with other workflows and to adapt to other applications. What can be done to increase its generality?

In response, Reviewer made concrete recommendations for modifications, which were passed to the Formaliser. The representation resulting from the multiple steps of iterative improvement is included in Appendix A.10.1.

Generally, the experiments on translating narrative workflow descriptions into ATWL representations with LLM’s assistance demonstrated the feasibility of this approach, which substantially reduces human efforts required for knowledge formalisation.

1.4 ATWL Representations Obtained With LLM Assistance

The ATWL representations of the workflows from papers are put together in Appendix C:

- C.1: Cluster-Calendar [14]
- C.2: Dynamic network analysis [13]
- C.3: Graph-based analysis of mass mobility flows [15]
- C.4: EventFlow [10]
- C.5: EventAction [7]
- C.6: Extracting significant places from trajectories [1]
- C.7: Progressive clustering of trajectories [12]
- C.8: Human-steered topic modelling [6]
- C.9: Progressive abstraction of multivariate temporal data [4]
- C.10: Partition-based Regression Modelling [11].
- C.12: Data engineering for movement pattern classification [3]
- C.13: Exploratory Model Analysis [5]
- C.14: Binary Classifier Diagnostics [9]
- C.15: Interactive exploration of random forest classifier [8]
- C.16: Exploring Deep Learning Models in TensorFlow [17]
- C.17: What-If Probing of ML Models [16]

1.5 Systematic Review and Language Refinement

After collecting all 16 workflow representations produced through the experiment procedure described above, a systematic review was conducted collaboratively between a human expert and an LLM agent (Claude Opus 4.6R, used as an interactive assistant rather than an autonomous agent). The LLM was provided with the full ATWL specification document, the document containing all 16 representations, and was asked to check them for correctness and consistency with respect to the ATWL syntax and semantics.

Initial audit

The LLM produced a structured audit report (see Appendix E) identifying seven cross-cutting *systematic issues* affecting multiple workflows and numerous *workflow-specific issues*. The systematic issues were:

- S1. `feature` artifacts declaring `representation form` instead of the required `value structure field` (6 workflows).
- S2. `visualisation` artifacts using a single `representation form` field instead of the required `layout, form, and encoding` fields (6 workflows).
- S3. `entities` artifacts missing the `embedding` field (6 workflows).
- S4. `generate-knowledge` transforms producing `specification` outputs, which was not explicitly sanctioned by the language definition (8 workflows).
- S5. `model` artifacts missing type parameterisation (5 workflows).
- S6. Loops lacking explicit `specification` artifacts and `assign` statements to close the feedback cycle (4 workflows).
- S7. Loops lacking an explicit exit mechanism despite using `assess` transforms (2 workflows).

The LLM also noted that issues S1–S3 affected exactly the same six workflows, suggesting that these were produced by a different LLM configuration or prompting strategy than the remaining workflows, which were largely free of these issues.

Iterative correction

The human expert and the LLM then proceeded to correct the workflow representations one by one. For each workflow, the LLM identified remaining issues and proposed specific corrections as replacement text fragments. The human expert evaluated each proposal, accepting, modifying, or rejecting it. In several cases, the expert disagreed with the LLM’s proposals, which led to productive discussions and ultimately to refinements of the ATWL definition itself.

Language refinements triggered by the review

The review process revealed ambiguities and gaps in the ATWL specification that had not been apparent during its initial design. The following refinements were made to the language definition as a direct result of the collaborative review:

- **Specification production by generate-knowledge.** Since specifications represent *control knowledge* in the workflow, the definition was updated to explicitly state that `generate-knowledge` transforms may produce `specification` artifacts when the analyst’s primary cognitive act is to formulate a methodological decision. This resolved S4 across all affected workflows.
- **Two styles of loop termination.** The definition was updated to distinguish *explicit assessment* (an `assess` transform followed by a conditional with `exit loop`) from *implicit termination* (the `until` condition is monitored informally by the analyst). The latter style was recognised as appropriate when the stopping criterion reflects a gradual, holistic change in the analyst’s understanding that cannot be naturally reduced to a single assessable artifact. The `exit loop` construct was also clarified as allowed in either branch of a conditional, not restricted to the `else` branch.
- **Pre-loop assignment.** The `assign` construct was extended to allow use before a loop to initialise an artifact identifier that will be re-assigned within the loop body, rather than being restricted exclusively to loop interiors.
- **Embedment for single entities.** A clarification was added that the `embedment` field is omitted when an `entities` artifact represents a single analytical object (e.g., a single aggregation tree or a single projection space), since embedment characterises how *multiple* entities are positioned relative to each other.
- **Unified feature value scheme.** The most substantial refinement concerned the description of features. The original specification used different vocabularies for internal features (inside `entities` declarations) and standalone `feature` artifacts. These were unified into a single two-level scheme: `value structure` describes the overall organisation (`atomic`, `list`, `vector`, `matrix`, or `relational configuration`) and `value type` describes the type of atomic components (`numeric`, `ordinal`, `categorical`, `temporal`, `spatial`, `text`, or `reference`). The previous distinction between `homogeneous vector` and `heterogeneous vector` was eliminated in favour of a single `vector` structure, since the distinction was found to have no effect on how artifacts are treated in workflows. The value structures `matrix` and `list` were added to cover pairwise similarity matrices and member enumerations, respectively.
- **Specification production by assess.** A clarification confirmed that `assess` transforms may optionally produce `specification` artifacts alongside evaluative `knowledge`, as when threshold adjustments are a direct by-product of quality evaluation.

- **Tangibility of knowledge artifacts.** The review identified several knowledge artifacts whose `representation form` described mental states (e.g., “architectural understanding”, “comprehensive model assessment”) rather than tangible outputs. Since ATWL represents explicit, recorded artifacts rather than cognitive states, these were corrected to refer to concrete representations such as “statements and diagrams” or “statements and recommendations”.

1.6 Creation of procedural instructions

The collaborative review also informed the development of procedural guidance documents for future use of ATWL with LLM assistance. Based on the patterns of errors observed across the 16 workflows and the corrective strategies that proved effective, the LLM was asked to produce two sets of instructions (included in Appendix):

1. **Instructions for workflow extraction and representation** (for the Formaliser agent), covering: how to identify the appropriate level of abstraction; how to distinguish artifact types (in particular, `pattern` vs. `model`, `knowledge` vs. `specification`, and `feature` vs. `pattern`); how to structure loops and conditionals; and how to apply the unified value structure and value type scheme. See Appendix F.
2. **Instructions for reviewing representations** (for the Reviewer agent), covering: a checklist of syntactic and semantic requirements; common error patterns to watch for (drawn from the systematic issues S1–S7); and guidance on evaluating intent–output type consistency. See Appendix G.

The LLM also recommended four representative examples from the corrected workflow collection to be provided to the agents along with the instructions, selected to collectively cover the full range of ATWL constructs while remaining diverse in analytical pattern and complexity:

- **Cluster-Calendar** (C.1): the cleanest, most canonical iterative workflow. Shows the fundamental refinement pattern in its simplest form: a single loop with explicit assessment (`Pattern A`), a `specification` artifact with iterative update via `assign`, `arrangement` with `contextualise`, and post-loop `generate-knowledge`.
- **EventFlow** (C.4): a complex single loop with a rich conditional body containing multiple transforms (`filter`, `restructure`, `assign`). Demonstrates when to omit `embedment` (the aggregation tree is a single structured entity), entities with `internal structure`: `formation`, and a given `knowledge` artifact influencing assessment.
- **Data engineering for movement pattern classification** (C.12): the best example of a two-loop workflow where the analytical goal shifts between loops (feature refinement followed by model validation). Shows `build-model` and model validation; demonstrates that `characterise` (not `define-unit`) is correct for applying an existing model to assign labels; includes a projection reference space as a single entity without embedment.

- **Progressive clustering of trajectories (C.7)**: the most structurally diverse workflow, with multi-phase progressive analysis, pre-loop `assign` for initialisation, and both loop termination patterns in a single workflow (explicit conditional exit in the destination clustering loop and implicit termination in the route analysis loop). Also illustrates cross-phase synthesis where a post-loop transform consumes artifacts from multiple earlier phases.

These four examples were selected because they collectively cover all eight artifact types, all eight transform intents, both loop termination patterns, pre-loop and in-loop assignment, single-entity and collection-entity embedment conventions, simple and complex conditional branches, and single-loop and multi-loop workflow structures. The one construct not illustrated by these examples — `assess` producing a `specification` artifact alongside evaluative `knowledge` — is addressed by a brief annotated example in the extraction instructions. The four workflows were recommended for inclusion in prompts for both the Formaliser and Reviewer agents.

1.7 Testing the use of the instructions

To validate the procedural instructions, a fresh extraction experiment was conducted using two new, independent LLM instances (Claude Opus 4.6R) that had not participated in any of the preceding experiments or the review process. The first instance (Formaliser) was provided with the full ATWL language definition and the extraction instructions, along with the four recommended representative examples. It was then given the PDF of the paper by Andrienko and Andrienko [2] on spatio-temporal analysis and modelling through time series fitting, and requested to produce a natural language workflow summary and an ATWL representation.

The second instance (Reviewer) was provided with the language definition and the reviewing instructions, and was asked to check the Formaliser’s output for alignment with the paper content, syntactic and semantic correctness, appropriate level of abstraction, and possibilities for simplification. The two-agent procedure of passing Reviewer feedback to Formaliser and returning revised versions to Reviewer was repeated over several iterations until both agents and the human expert were satisfied with the result. The final workflow representation is included in Appendix C.11.

1.7.1 Formaliser session

The Formaliser instance was initialised with the full ATWL language specification, the extraction instructions document (including abstraction guidelines) (Appendix F), and the four representative example workflows (Cluster-Calendar, EventFlow, Data Engineering for Movement Pattern Classification, and Progressive Clustering of Trajectories). It was then provided with the PDF of the paper on spatio-temporal analysis and modelling [2] and asked to produce a natural language workflow summary and an ATWL representation.

First version. The Formaliser produced an initial workflow with a correct high-level structure—an outer analysis cycle containing inner grouping and modelling refinement loops—but with several syntactic and representational issues. It used a flat, keyword-based syntax (e.g., `entities`, `feature`, `transform` without the `artifact/transform ID : type` convention), did not declare artifact types with parenthesised references, omitted `origin: given` markers, and used `representation form` on `knowledge` artifacts declared directly with the `knowledge` keyword rather than as typed `artifact` declarations. The template was absent, and control structures used a non-standard format. Overall, the representation followed the general analytical logic of the paper but did not conform to the ATWL syntax demonstrated in the provided examples.

Self-review and second version. When informed that the result was unsatisfactory and asked to review it more carefully against the instructions and examples, the Formaliser produced a substantially revised representation. This version adopted the correct `artifact ID : type` and `transform ID : declaration` syntax, introduced a well-abstracted template, added `origin: given` to exogenous artifacts, properly structured loops with `purpose`, `until`, `body`, and `end loop` keywords, and used the conditional-exit pattern (Pattern A) for all three loops. The three-level loop nesting (outer analysis cycle, inner grouping refinement, inner model refinement) correctly captured the paper’s iterative structure. Artifact typing and intent–output matching were accurate throughout.

First Reviewer feedback and third version. The Reviewer identified one critical issue and three recommended improvements. The critical issue was the use of `representation form` as a field on four `feature` artifacts, which is not a valid ATWL feature field. The recommended improvements were:

1. Adding `model type` and `representation form` fields to `model` artifacts, together with a dedicated `specification` artifact for model configuration.
2. Broadening the residual-assessment else-branch to capture both refinement paths described in the paper (re-cluster and re-model), not only the re-clustering path.
3. Correcting a compound value type: `{numeric, temporal, spatial}` and incomplete `embedment: {set, time}` on the input data artifact.

The Reviewer also offered minor suggestions regarding residual clustering as a diagnostic technique, multiple time-cycle handling, and per-group iteration. The Formaliser implemented all corrections: it removed `representation form` from all features; added `model type` and `representation form` to both `model` artifacts; introduced `S_model_config` as a `specification` artifact with a dedicated `T_configure_model` transform and assignment-based update inside the modelling loop; expanded the outer-loop else-branch to produce both updated clustering parameters and a refinement strategy `knowledge` artifact; corrected the input entity’s `embedment` to `{set, time, space}` with a simple `numeric` value type; and incorporated all minor suggestions into relevant descriptions.

Second Reviewer feedback and final version. The Reviewer rated the updated workflow as ready for use with optional minor polish and identified two remaining minor issues. First, the artifact

`K_refinement_strategy` produced in the outer loop’s else-branch was never consumed by any subsequent transform, making it a dead-end in the dependency graph. The Reviewer offered two resolution options; the human expert selected the option of adding it as input to `T_configure_model` and declaring it with `origin: given` before the outer loop to handle the first iteration when no prior refinement guidance exists. Second, the Reviewer suggested adding `V_model` as a direct input to `T_adjust_model_config` inside the modelling loop, to more accurately represent the analyst’s interactive parameter adjustment while viewing the model curve. The Formaliser implemented both changes, producing the final version of the workflow representation.

The complete session required four Formaliser responses across three feedback cycles (one self-review and two Reviewer-mediated rounds), converging on a representation that both agents and the human expert judged satisfactory. The final workflow representation is included in Appendix C.11.

1.7.2 Reviewer session

The Reviewer instance was initialised with the full ATWL language specification, the document containing example workflow representations, and the reviewing instructions document (including the common-issues checklist and the five-phase review framework) (Appendix G). It was then provided with the PDF of the spatio-temporal analysis and modelling paper [2] and the Formaliser’s second version of the workflow representation (produced after the Formaliser’s self-review), along with the natural language workflow summary.

First review. The Reviewer produced a structured report rating the workflow as Yellow–Green (sound overall, targeted fixes needed). It identified seven strengths—including correct three-level loop architecture, accurate intent–output matching, well-modelled visualisation artifacts, correct use of `origin: given`, and clean dependency graph—and flagged two critical issues and several recommended improvements. The critical issues were: (1) the use of `representation form` as a field on four `feature` artifacts, which is not a valid ATWL feature field; and (2) missing `parameterisation` on model artifacts. The recommended improvements included restructuring the template to expose nested loops, broadening the residual-assessment else-branch to capture both refinement paths, correcting a non-standard compound `value type` and incomplete `embedding` on the input data artifact, and adjusting model artifact parenthesised references. Minor suggestions covered residual clustering as a diagnostic technique, multiple time-cycle handling, and per-group iteration.

Human expert corrections. The human expert reviewed the Reviewer’s report and identified three errors in the feedback that required correction before it could be passed to the Formaliser.

First, the recommendation to introduce nested `loop(...)` notation in the template contradicted the language definition, which explicitly states that nested loops should be avoided in templates to maintain clarity and abstraction. The Reviewer acknowledged this error, withdrew the recommendation, and reclassified the original template as a strength.

Second, the corrected `embedment` for the input data artifact used the keyword `spatial`, whereas the ATWL specification defines the embedment keyword as `space`. The Reviewer also initially stated that `spatial` is not a valid value type, which was incorrect—it is a valid `value type` for features. The Reviewer corrected both errors, using `space` in the embedment and clarifying that the issue was one of appropriate field assignment (`embedment` vs. `value type`) rather than keyword validity.

Third, the claim that `parameterisation` is a required field on model artifacts was not supported by the language definition or the provided examples, which instead showed models using `model type` and `representation form` fields, with configuration details captured in separate `specification` artifacts. The Reviewer restructured this issue from a critical fix to a recommended improvement: adding `model type` and `representation form` to the model artifacts and introducing a dedicated `specification` artifact for model configuration, following the pattern demonstrated in the examples.

After each correction, the Reviewer regenerated the complete feedback document to ensure consistency.

Second review. After the Formaliser produced an updated workflow incorporating all feedback, the Reviewer was provided with the revised representation and produced a second review. It confirmed that all previously identified issues had been addressed and upgraded the rating to Green (ready for use with optional minor polish). The Reviewer identified two remaining minor issues: (1) the `K_refinement_strategy` artifact produced in the outer loop’s else-branch was never consumed as input by any subsequent transform, creating a dead-end in the dependency graph; and (2) adding `V_model` as a direct input to `T_adjust_model_config` would more accurately represent the interactive nature of the analyst’s parameter adjustment. For the first issue, the Reviewer offered two resolution options: either connecting the artifact as input to `T_configure_model` in the next iteration (with an `origin: given` initialisation for the first pass), or removing it and folding its content into the transform description. The human expert selected the first option, and the feedback was passed to the Formaliser for a final revision.

References

- [1] Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 161–170, 2011.

- [2] Natalia Andrienko and Gennady Andrienko. A visual analytics framework for spatio-temporal analysis and modelling. *Data Mining and Knowledge Discovery*, 27(1):55–83, Jul 2013.
- [3] Natalia Andrienko, Gennady Andrienko, Alexander Artikis, Periklis Mantenoglou, and Salvatore Rinzivillo. Human-in-the-loop: Visual analytics for building models recognizing behavioral patterns in time series. *IEEE Computer Graphics and Applications*, 44(3):14–29, 2024.
- [4] Natalia Andrienko, Gennady Andrienko, and Gota Shirato. Episodes and topics in multivariate temporal data. *Computer Graphics Forum*, 42(6):e14926, 2023.
- [5] Dylan Cashman, Shah Rukh Humayoun, Florian Heimerl, Kendall Park, Subhajit Das, John Thompson, Bahador Saket, Abigail Mosca, John Stasko, Alex Endert, Michael Gleicher, and Remco Chang. A user-based visual analytics workflow for exploratory model analysis. *Computer Graphics Forum*, 38(3):185–199, 2019.
- [6] Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001, 2013.
- [7] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. Eventaction: Visual analytics for temporal event sequence recommendation. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 61–70, 2016.
- [8] J. Eirich, M. Münch, D. Jäckle, M. Sedlmair, J. Bonart, and T. Schreck. Rfx: A design study for the interactive exploration of a random forest to enhance testing procedures for electrical engines. *Computer Graphics Forum*, 41(6):302–315, 2022.
- [9] Josua Krause, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs, and Enrico Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 162–172, 2017.
- [10] Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2227–2236, 2013.
- [11] Thomas Mühlbacher and Harald Piringer. A partition-based framework for building and validating regression models. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1962–1971, 2013.
- [12] Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia Andrienko, and Gennady Andrienko. Visually-driven analysis of movement data by progressive clustering. *Information Visualization*, 7:225–239, 07 2008.

- [13] Stef van den Elzen, Danny Holten, Jorik Blaas, and Jarke J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, January 2016.
- [14] Jarke J. van Wijk and Edward R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '99)*, pages 4–9, Los Alamitos, CA, USA, 1999. IEEE Computer Society.
- [15] Tatiana von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):11–20, 2016.
- [16] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda B. Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020.
- [17] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B. Viégas, and Martin Wattenberg. Visualizing dataflow graphs of deep learning models in TensorFlow. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):1–12, 2018.

A Representing Interleaved Workflows in ATWL: The Progressive Clustering Case

Note: The following text was generated by LLM Claude Sonnet 4.5R acting as a reviewing agent as a feedback on a representation of the progressive clustering workflow presented in paper [12].

A.1 The Challenge

The progressive trajectory clustering workflow described in Rinzivillo et al. (2007) presents an interesting challenge for ATWL representation. The paper’s analytical progression is not strictly linear but involves:

- Iterative parameter refinement within analytical phases
- Shifts between analyzing subsets and the complete dataset
- Different distance functions applied to different data scopes
- Hypothesis generation from subset analysis followed by verification on full dataset

The question arises: **Can ATWL elegantly represent such interleaved structure, or is a simplified representation preferable?**

A.2 The Paper’s Actual Analytical Progression

Reading Section 5 of the paper carefully reveals that the workflow answers **four successive analytical questions**:

Phase	Question	Method	Scope
1	Where do people go?	Cluster by “common destination” with varying MinNbs	ALL trajectories, progressive refinement
2	What routes to major destinations?	Cluster by “route similarity”	SUBSETS (red, blue, green clusters)
3	Are short local trips common everywhere?	Cluster by “common source and destination”	Back to ALL trajectories (verification)
4	What routes for longer trips?	Cluster by “route similarity”	REMAINING after excluding local trips

Table 1: Four analytical phases in the progressive clustering workflow

Key insight: This is a *logical progression*, not arbitrary branching. Each phase builds on findings from the previous one.

A.3 Recommended ATWL Structure: Sequential Phases

Answer: Yes, ATWL can represent the interleaved workflow elegantly, but as a **sequence of logical phases** rather than complex nested loops.

A.3.1 Simplified Structure

```
workflow progressive-trajectory-clustering
  template: define-unit (destinations: progressive density) →
            define-unit (routes: subset analysis) →
            define-unit (verification: all trajectories) →
            define-unit (routes: remaining trajectories) →
            abstract → generate-knowledge

// Phase 1: Where do people go? (Section 5.2)
loop L_destination_clustering:
  purpose: "Handle uneven density by progressively clustering
           with same distance function but varying sensitivity"
  until: "Major destination regions identified across all
         density levels"
  body:
    # Cluster by destination with current MinNbs
    # Visualize and assess
    # Exclude identified clusters, reduce MinNbs, repeat

// Phase 2: What routes to major destinations? (Section 5.3, first)
transform T_select_major_destination_clusters:
  input: destination_clusters
  output: selected_destination_clusters
  description: "Select major destination clusters (red, blue,
              green) for in-depth route analysis"

loop L_route_analysis_subsets:
  purpose: "Examine routes to major destinations"
  body:
    # Select one major destination cluster
    # Cluster by route similarity
    # Visualize and assess
    # Find short local trips within each

// Phase 3: Are short local trips common everywhere?
//           (Section 5.3, middle) - CRITICAL STEP
transform T_verify_local_trips:
  intent: define-unit
  manner: "cluster by common source and destination"
  input: trajectories # ALL trajectories!
  output: source_dest_clusters, local_trip_clusters
  description: "Verify hypothesis that short local trips are
              common throughout dataset by clustering ALL trajectories
              by source-destination proximity"
```

```

artifact local_trip_clusters : entities
  description: "Clusters where starts are close to ends
    (23% of all trajectories)"

// Phase 4: What routes for longer trips? (Section 5.3, final)
transform T_exclude_local:
  intent: define-unit
  manner: "filter by excluding clusters"
  input: trajectories, local_trip_clusters
  output: long_distance_trajectories

transform T_cluster_routes_final:
  intent: define-unit
  manner: "cluster by route similarity"
  input: long_distance_trajectories
  output: route_clusters
  description: "Identify frequent routes among longer trips"

# Visualize, assess, generate knowledge

```

A.4 Why Sequential Phases Are Superior

A.4.1 Advantages of Sequential Structure

1. **Reflects analytical logic:** Each phase answers a distinct analytical question
2. **Easier to understand:** Clear progression through the analysis
3. **Matches paper's narrative:** Section 5.2 (destinations) → Section 5.3 (routes) → Section 5.3 (verification) → Section 5.3 (final routes)
4. **Appropriate abstraction:** Captures *what* was done and *why*, not implementation details
5. **Facilitates comparison:** Other analysts can see the analytical strategy
6. **Handles scope changes naturally:** Return to full dataset is simply Phase 3

A.4.2 Problems with Complex Nesting

1. **Hard to follow:** Multiple levels of loops and conditionals obscure logic
2. **Obscures reasoning:** Why return to all trajectories? Why exclude local trips?
3. **Implementation-focused:** Details of iteration instead of analytical reasoning
4. **Not comparable:** Too specific to this exact execution
5. **Awkward scope handling:** Returning to full dataset within nested structure creates complexity

A.5 The Critical Missing Element

Both attempted ATWL versions are missing **Phase 3** (verification on ALL trajectories), which is **essential** to the paper’s findings. From Section 5.3:

*“This observation about the trajectories in the three largest clusters makes us think that it may be also true for the whole set of trajectories. **To check this, we apply clustering with the distance function ‘common source and destination’ to the whole set of trajectories.**”* (emphasis added)

This verification step:

- Tests a hypothesis generated from subset analysis
- Applies to **ALL trajectories** (not a subset)
- Leads to the discovery that $\sim 23\%$ of all trips are local
- Informs the decision to exclude local trips before final route analysis
- Is necessary before the final route clustering

Without this step, the workflow is incomplete and the analytical logic is broken.

A.6 Comparison: Complex Nesting vs. Sequential Phases

Aspect	Complex Nesting	Sequential Phases
Readability	Low (multiple loop levels)	High (clear progression)
Analytical logic	Obscured by control structures	Explicit through phase structure
Workflow comparison	Difficult (implementation-specific)	Easy (conceptual similarity)
Captures paper’s intent	Partial (misses verification)	Complete (all phases)
Abstraction level	Too detailed (iteration mechanics)	Appropriate (analytical strategy)
Handles scope changes	Awkward (subset \rightarrow all \rightarrow remaining)	Natural (phases 2 \rightarrow 3 \rightarrow 4)

Table 2: Comparison of representation approaches

A.7 Scope Changes Are Not Complexity to Hide

The fact that phases 2, 3, and 4 operate on different subsets of data is **not a complexity to hide** but **the core analytical strategy to highlight**:

- Phase 2: Selected destination clusters (subsets)

- Phase 3: Return to **ALL** trajectories (verification)
- Phase 4: Remaining trajectories after exclusion

This pattern—*explore subset* → *verify hypothesis on all* → *refine based on findings*—is a **general analytical strategy** that may appear in other workflows and should be clearly visible in the ATWL representation.

A.8 Recommendation

Use the **sequential phase structure** with explicit phases for:

1. Destination-based progressive clustering (varying MinNbs)
2. Route analysis of major destination subsets
3. **Verification of local trips hypothesis on ALL trajectories** (currently missing)
4. Exclusion of local trips and final route analysis

Rationale:

- ATWL’s strength is representing *analytical reasoning*, not *execution details*
- The sequential phase structure shows *how the analyst’s understanding evolved*
- This representation facilitates comparison with other progressive analysis workflows
- The approach is elegant, understandable, and captures the paper’s contribution faithfully

A.9 General Principle for ATWL Workflow Design

When facing complex, interleaved workflows:

Model the logical progression of analytical questions, not the execution trace.

- Identify the **analytical questions** driving the workflow
- Organize phases by **question-method-scope** combinations
- Make **scope changes explicit** (subsets vs. complete dataset)
- Represent **hypothesis generation and verification** as distinct phases
- Keep complexity proportional to **conceptual sophistication**, not implementation detail

A.10 Conclusion

ATWL **can** elegantly represent interleaved workflows like progressive clustering by structuring them as **sequences of logical phases** rather than complex nested control structures. This approach:

- Maintains appropriate abstraction level for workflow comparison
- Makes analytical reasoning transparent
- Handles scope changes (subset \leftrightarrow all dataset) naturally
- Highlights rather than obscures the core analytical strategy
- Remains understandable and comparable across different studies

The key is to remember that ATWL represents *analytical recipes* (what questions were asked, what methods answered them) rather than *computational cookbooks* (how loops were nested, what variables were assigned).

A.10.1 A workflow example with iteration and branching

The following ATWL code represents the workflow for progressive clustering of trajectories from paper [12].

```
workflow progressive-clustering-of-movement-data
  pattern: cluster → assess → refine (loop) →
    branch → re-cluster → interpret → generate knowledge

# input data
artifact D0 : data-unit
  structural form: collection of sequences
  semantics:
    reference target:
      temporality = temporal
      spatiality = aspatial          # moving objects
    attribute associations: spatial,  # coordinates
      numeric                        # speed, direction, ...
  description: "Movement trajectories represented as
    time-ordered sequences of spatial positions"

transform T1 : structure
  manner: density-based clustering using a chosen
    distance function
  input: D0
  output: D1, F1
  actor: hybrid
  description: "Cluster trajectories using a chosen distance
    function and initial clustering parameters"

artifact D1 : data-unit(D0)
  structural form: cluster
  description: "Clusters of trajectories with common properties
    the distance function deals with"

artifact F1 : feature(D0)
  structural form: categorical label
  description: "Labels denoting cluster affiliations of trajectories"

loop L1:
  purpose : obtain meaningful clusters of different density
  until : clusters are internally coherent and interpretable;
    remaining trajectories do not make significant groups
  body :
    transform T2 : visualise
      manner: lines on a geographic map (D0) + colours (F1)
      input: D0, F1
      output: V1
      actor: machine
      description: "Visualise clusters of trajectories
        on a geographic map"

    artifact V1 : visualisation(D0, F1)
```

```

    layout: map-based; form: coloured lines
    description: "Clusters displayed with distinct colours"

transform T3 : abstract
  manner: perception, interpretation, comparison
  input: V1
  output: P1, P2
  actor: human
  description: "Assess clustering results"

artifact P1 : pattern(D1)
  representation form: qualitative assessment
  description: "Judgement about variability of cluster
    properties (size, extent, internal coherence) and
    information loss (proportion of unclustered objects)"

artifact P2 : pattern(D1)
  representation form: qualitative assessment
  description: "Judgements about quality (internal coherence,
    interpretability) of individual clusters"

transform T4 : extract
  manner: interactive
  input: D1, V1, P2
  output: D2, F2
  actor: human
  description: "Select high quality clusters for further use"

artifact D2 : data-unit(D0)
  structural form: cluster
  description: "Selected clusters of good quality"

artifact F2 : feature(D0)
  structural form: categorical label
  description: "Labels denoting cluster affiliations of
    trajectories included in D2"

if P1 = too extensive or too loose clusters or
  high information loss
then:
  transform T5 : structure
    manner: adjust clustering parameters and re-cluster
    input: D0, P1
    output: D1', F1'
    actor: hybrid
    description: "Refine clustering parameters to
      compensate for uneven data density,
      keeping the distance function unchanged"

  assign:
    D1 := D1'
    F1 := F1'

```

```

        else:
            exit loop L1
end loop L1

loop L2
  purpose: knowledge enrichment
  until: all clusters in D2 studied
  body:
    transform T6: extract
      input: D2, D0, F2
      output: D3
      actor: human
      description: "Select one or more clusters for
        in-depth study"

    artifact D3: data-unit(D0)
      structural form: cluster
      description: "Members of selected clusters"

    transform T7 : structure
      manner: density-based clustering using another
        distance function
      input: D3
      output: D4, F3
      actor: hybrid
      description: "Apply clustering with a different distance
        function to reveal additional behavioural patterns"

    artifact D4 : data-unit(D0)
      structural form: cluster
      description: "Secondary clusters of trajectories with
        common properties the current and previously used
        distance functions deal with"

    artifact F3 : feature(D0)
      structural form: categorical label
      description: "Labels denoting affiliations of
        trajectories in D4"

    transform T8 : visualise
      manner: lines on a geographic map (D0) + colours (F3)
      input: D0, F3
      output: V2
      actor: machine
      description: "Visualise secondary clusters of trajectories
        on a geographic map"

    artifact V2 : visualisation(D0, F1)
      layout: map-based; form: coloured lines
      description: "Clusters displayed with distinct colours"

    transform T9 : abstract

```

```
manner: perception, interpretation, comparison
input: V2, D4
output: P3
actor: human
description: "Identify different patterns of movement
behaviour"
```

```
artifact P3 : pattern(D4)
representation form: textual annotation
description: "Conceptual patterns of movement behaviour"
```

```
end loop L2
```

```
transform T10 : generate knowledge
manner: formulate analytical conclusions
input: P3
output: K1
actor: human
description: "Generate knowledge about existing movement
patterns"
```

```
artifact K1 : knowledge-unit(P3)
representation form: statements
description: "Analytical statements about dominant movement
patterns and their variability"
```

B Guidelines for Appropriate Abstraction Level in ATWL Representations

Note: The following text was generated by Claude Sonnet 4.5R after a conversation on reviewing 3 ATWL representations of workflows extracted from published papers.

B.1 Core Principle

ATWL represents workflows at a conceptual level that facilitates comparison and pattern identification across different studies, while remaining specific enough to understand the analytical logic. Think of ATWL as capturing the “analytical recipe” rather than the “cooking instructions with exact measurements and equipment specifications.”

B.2 Focus on WHAT, Not HOW

Include: The analytical purpose and type of operation

Omit: Specific algorithms, tools, or implementation details

Too Detailed	Appropriate Level	Too Abstract
Apply k-means clustering with k=5 using scikit-learn with Euclidean distance	Cluster episodes by feature similarity (manner: “cluster-by-similarity”)	Group things
Use UMAP with n_neighbors=15, min_dist=0.1 implemented in Python	Project to 2D space preserving similarity (manner: “projection by dimensionality reduction”)	Reduce dimensions
Draw line chart using D3.js with SVG rendering	Display temporal profiles as line graphs (manner: “line-graph”)	Show data

Table 3: Examples of abstraction levels

B.3 Transform Specifications

B.3.1 Intent (Required)

Use **one of the seven generic intents** from the ATWL specification:

- `define-unit`
- `characterise`
- `contextualise`
- `visualise`

- abstract
- build-model
- generate-knowledge

B.3.2 Manner (Optional but Recommended)

Provide **conceptual refinement** without implementation details:

Good manners:

- “time-partitioning into episodes”
- “cluster-by-similarity”
- “compute synoptic statistics”
- “calendar-based arrangement”
- “perception-and-interpretation”

Too specific (avoid):

- “k-means clustering with k=5”
- “UMAP dimensionality reduction”
- “SQL GROUP BY query”
- “matplotlib bar chart”

Too vague (avoid):

- “processing”
- “calculation”
- “display”

B.3.3 Actor (Required)

Always specify: human, machine, or hybrid

B.4 Artifact Descriptions

B.4.1 DO Include:

- Analytical role and conceptual structure
- Key distinguishing characteristics
- Semantic meaning

B.4.2 DO NOT Include:

- Data formats (JSON, CSV, database schema)
- Storage details (file paths, table names)
- Size metrics (number of rows, memory usage)
- Technical implementation

B.4.3 Example: Feature Artifact

Too detailed (avoid):

```
artifact F1 : feature(D1)
  value structure: heterogeneous vector
  description: "5-dimensional numpy array containing:
    mean speed (float32), max acceleration (float32),
    trajectory length (int), sinuosity (float64),
    time duration (timedelta64), stored in pandas
    DataFrame with 10,000 rows"
```

Appropriate:

```
artifact F1 : feature(D1)
  value structure: heterogeneous vector
  description: "Synoptic features capturing movement
    characteristics: speed statistics, trajectory
    geometry, and temporal extent"
```

Too abstract (avoid):

```
artifact F1 : feature(D1)
  value structure: heterogeneous vector
  description: "Features of trajectories"
```

B.4.4 Example: Entities Artifact

Too detailed (avoid):

```
artifact D1 : entities
  internal structure: episode
  embedment: time
  features:
    - id: patient_id (UUID format)
    - id: start_time (ISO 8601 datetime)
    - id: end_time (ISO 8601 datetime)
    - id: event_count (32-bit integer)
    - id: primary_diagnosis_code (ICD-10 string)
    - id: secondary_diagnoses (array of ICD-10 codes)
    - id: medications (JSON with drug names/dosages)
```

```
description: "Electronic health record episodes
stored in PostgreSQL..."
```

Appropriate:

```
artifact D1 : entities
  internal structure: episode
  embedment: time
  features:
    - id: episode_id

    value type: categorical
    - id: temporal_extent

    value type: temporal
  description: "Patient care episodes containing
time-referenced clinical events"
```

Too abstract (avoid):

```
artifact D1 : entities
  description: "Patient data"
```

B.5 Control Structures

B.5.1 Loop Termination Conditions

Express as **qualitative analytical goals**:

Good examples:

```
until: "Clusters provide clear, interpretable
decomposition of daily patterns"
until: "Model errors are understood and minimized"
until: "Feature space adequately distinguishes
target patterns"
```

Avoid quantitative thresholds:

```
until: "Silhouette score > 0.65"
until: "Classification accuracy > 95%"
until: "Number of clusters < 8"
```

Rationale: Specific thresholds vary by application and are implementation details. The analytical goal is what matters.

B.6 Features of Entities

B.6.1 When to List Individual Features

- When they represent **distinct semantic categories** important for analysis
- When they play **different roles** in the workflow

- For **exogenous artifacts** (`origin: given`) to show what's available

B.6.2 When to Consolidate

- When features are **similar in type and role**
- For **derived artifacts** where specific features are less important
- When the feature set is **large and varied**

B.6.3 Example

Good: Distinct semantic categories

```
artifact D_trajectories : entities
  origin: given
  features:
    - id: position

      value type: spatial
    - id: speed

      value type: numeric
    - id: heading

      value type: numeric
  description: "Movement trajectories..."
```

Also good: Consolidated when less critical

```
artifact D_trajectories : entities
  origin: given
  features:
    - id: movement_attributes

      value type: heterogeneous vector
      description: "Spatial positions, speeds,
        and movement directions"
  description: "Movement trajectories..."
```

Too detailed (avoid): Listing every column

```
artifact D_trajectories : entities
  features:
    - id: longitude (decimal degrees, WGS84)
    - id: latitude (decimal degrees, WGS84)
    - id: altitude (meters above sea level)
    - id: speed_kmh (kilometers per hour)
    - id: speed_knots (nautical miles per hour)
    - id: heading_degrees (0-360, true north)
    - id: heading_radians (0-2pi)
    - id: gps_accuracy (meters)
    - id: timestamp (Unix epoch milliseconds)
```

B.7 Descriptions

Keep descriptions **concise but informative**:

- **Good length:** 1–2 sentences (10–25 words)
- **Too long:** Paragraph-length explanations with multiple clauses
- **Too short:** Single word or fragment

Too verbose (avoid):

```
description: "This transform takes the input trajectories
and applies a sophisticated time-based partitioning
algorithm that analyzes the temporal distribution of
trajectory points and identifies natural breakpoints
based on gaps in the temporal sequence, then creates
episodic segments each representing a contiguous period
of movement activity separated from other episodes by
periods of inactivity or gaps in data recording"
```

Appropriate:

```
description: "Partition trajectories into episodes
based on temporal gaps"
```

Too terse (avoid):

```
description: "Partition data"
```

B.8 Domain Specifics

Balance domain context with generalizability:

- **Do** mention domain to provide context
- **Do** use generic terms when possible
- **Don't** enumerate all domain categories

Good: Generic with domain context

```
description: "Event sequences from electronic health
records capturing patient diagnoses, medications,
and procedures"
```

Too domain-specific (avoid):

```
description: "Event sequences containing exactly 127
ICD-10 diagnosis codes, 83 prescription drug categories
(including Albuterol, Budesonide, Fluticasone,
Montelukast, ...), and 45 procedure types (including
Spirometry, Chest X-ray, ...)"
```

Too generic (avoid):

```
| description: "Event sequences"
```

B.9 Abstracting Low-Level Details

If the paper provides extensive implementation details, **abstract upward** to the conceptual level.

Paper states: “We implemented a custom k-nearest-neighbor classifier using the scikit-learn `KNeighborsClassifier` class with `k=7`, `weights='distance'`, `algorithm='ball_tree'`, `leaf_size=30`, and `p=2` (Euclidean distance). The model was trained on 80% of the labeled examples...”

ATWL representation:

```
transform T_train :
  intent: build-model
  manner: "supervised classification model training"
  input: D_labeled, F_features, F_labels
  output: M_classifier
  actor: hybrid
  description: "Train classifier to recognize behavioral
    patterns from labeled examples"

artifact M_classifier : model(F_features, F_labels)
  model type: "classifier"
  representation form: "instance-based learning model"
  description: "Model for automated pattern recognition"
```

B.10 Quick Reference Checklist

When creating ATWL representations, ask yourself:

1. Could someone understand the **analytical logic** without knowing specific tools?
2. Could this workflow be **compared** to similar workflows from other papers?
3. Are **implementation details** (algorithms, tools, parameters) moved to descriptions or omitted?
4. Are **analytical purposes** (intents) clearly specified?
5. Are descriptions **concise** (1–2 sentences) yet **informative**?
6. Do artifact types and structures reflect **conceptual organization** rather than data formats?
7. Are control structures expressed as **analytical conditions** rather than numerical thresholds?

B.11 Common Mistakes to Avoid

Mistake	Correction
Naming algorithms in intent	Use generic intent + conceptual manner
Including tool names (R, Python, Tableau)	Omit entirely or mention only in description
Listing all domain categories	Provide representative examples or generic description
Specifying exact parameters (k=5, threshold=0.8)	Describe qualitative purpose
Data format details (CSV, JSON, SQL)	Focus on semantic content
Multi-paragraph descriptions	Condense to 1–2 sentences
Using “process” or “analyze” without specificity	Use precise ATWL intents
Omitting actor types	Always specify human/machine/hybrid

Table 4: Common mistakes and their corrections

B.12 Final Guideline

When in doubt, think:

“If I’m comparing this workflow to ten other similar workflows, what level of detail helps me see patterns and differences in the analytical approach, without getting lost in implementation specifics?”

This is your target abstraction level.

C Visual Analytics Workflows Represented in ATWL

C.1 Cluster-calendar workflow

Source: Jarke J. van Wijk and Edward R. van Selow. Cluster and calendar based visualization of time series data. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis '99), pages 4–9, Los Alamitos, CA, USA, 1999. IEEE Computer Society. doi: 10.1109/INFVIS.1999.801851

Concise Workflow Summary

The cluster-calendar workflow combines hierarchical clustering with calendar-based visualization to identify and analyse recurring patterns in time series data measured at regular intervals (e.g., hourly) over extended periods. The workflow partitions continuous time series into daily episodes, characterizes each day by its temporal profile, and applies hierarchical bottom-up clustering to group days with similar patterns. Users iteratively refine the cluster structure—adjusting the number of clusters, selecting alternative distance measures, or focusing on specific time intervals—until meaningful behavioural patterns emerge. The results are presented through coordinated visualizations: a calendar view where days are colour-coded by cluster membership to reveal weekly and seasonal distributions, and line graphs showing the average temporal profile for each cluster to illustrate characteristic patterns. Through interactive exploration of these coordinated views, analysts identify both standard patterns (e.g., typical weekdays, weekends, seasonal variations) and exceptional days (e.g., holidays, special events), formulating insights about temporal regularities and anomalies that would be difficult to detect through traditional time series analysis methods.

Figure 2 provides a diagrammatic representation of the workflow.

ATWL representation

```
workflow cluster-calendar
  template: define-unit → contextualise → characterise →
    loop(define-unit(similarity-based) →
      characterise (groups) → visualise →
      abstract → assess) →
    generate-knowledge
  description: "Identify and analyze recurring daily patterns in time series
    data through interactive clustering and calendar-based visualization;
    detect standard patterns and exceptional days"

artifact D_hour : entities
  origin: given
  internal structure: elementary
  embedment: time
  features:
    - id: f_value
      value structure: atomic
```

```
    value type: numeric
    description: "Measured value"
description: "Time series measurements at regular intervals over extended period
"
```

```
artifact D_calendar : entities
```

```
  origin: given
  internal structure: elementary
  embedment: time
  features:
    - id: f_temporal_coords
      value structure: vector
      value type: {categorical, numeric}
      description: "Month, day of week, day number"
  description: "Calendar structure providing temporal context with
    month and weekday organization"
```

```
transform T_partition :
```

```
  intent: define-unit
  manner: "time-partitioning into daily episodes"
  input: D_hour
  output: D_day
  actor: machine
  description: "Organize time series into daily episodes, each containing
    measurements for one 24-hour period"
```

```
artifact D_day : entities
```

```
  internal structure: episode
  embedment: time
  features:
    - id: f_day_index
      value structure: atomic
      value type: ordinal
      description: "Sequential position of day in the year"
  description: "Daily episodes consisting of all measurements within each
    24-hour period"
```

```
transform T_arrange :
```

```
  intent: contextualise
  manner: "calendar-based"
  input: D_day, D_calendar
  output: A_calendar
  actor: machine
  description: "Arrange daily episodes in calendar context
    according to their temporal position"
```

```
artifact A_calendar : arrangement(D_day)
```

```
  context: D_calendar
  principle: "calendar date mapping to grid position"
  description: "Calendar-based arrangement where each day occupies
    its corresponding calendar cell"
```

```

transform T_profile :
  intent: characterise
  manner: "extract temporal profile"
  input: D_day
  output: F_day_profile
  actor: machine
  description: "Represent each day by its measurement sequence"

artifact F_day_profile : feature(D_day)
  value structure: vector
  value type: numeric
  description: "Daily temporal profile: sequence of measurements within each day"

artifact S_clustering : specification
  origin: given
  representation form: "parameter settings"
  description: "Initial parameters for hierarchical clustering:
    number of clusters (dendrogram cut level), distance measure
    (geometric, normalized, shift-invariant, max-based),
    time interval focus"

loop L1:
  purpose: "Iteratively explore cluster structure to identify meaningful and
    interpretable daily patterns"
  until: "Clusters provide clear, interpretable decomposition of daily patterns;
    standard patterns and exceptional days are identified"
  body:
    transform T_cluster :
      intent: define-unit
      manner: "hierarchical clustering by similarity"
      input: D_day, F_day_profile, S_clustering
      output: D_cluster, F_cluster_label
      actor: hybrid
      description: "Apply hierarchical clustering to group days with similar
        profiles; user selects cut through dendrogram to determine clusters"

    artifact D_cluster : entities
      internal structure: group/cluster
      embedment: set
      features:
        - id: cluster_size
          value structure: atomic
          value type: numeric
          description: "Number of days in cluster"
      description: "Groups of days with similar daily profiles selected from
        hierarchical clustering tree"

    artifact F_cluster_label : feature(D_day)
      value structure: atomic
      value type: categorical
      description: "Cluster membership identifier for each day"

```

```

transform T_aggregate :
  intent: characterise
  manner: "aggregate profiles per cluster"
  input: D_cluster, F_day_profile
  output: F_cluster_profile
  actor: machine
  description: "Compute average daily profile for each cluster to
    represent typical pattern"

artifact F_cluster_profile : feature(D_cluster)
  value structure: vector
  value type: numeric
  description: "Cluster-level average daily profiles representing typical
    patterns for each group"

transform T_calendar_vis :
  intent: visualise
  manner: "calendar grid with color-coded clusters"
  input: A_calendar, F_cluster_label
  output: V_calendar
  actor: machine
  description: "Display days on calendar grid, colored by cluster
membership"

artifact V_calendar : visualisation(A_calendar, F_cluster_label)
  layout: "calendar grid (months as rows, weekdays as columns)"
  form: "colored cells"
  encoding: "position from A_calendar; color from F_cluster_label"
  description: "Calendar view showing temporal distribution of cluster
    patterns across year and week"

transform T_profile_vis :
  intent: visualise
  manner: "line graphs of cluster profiles"
  input: F_cluster_profile, D_cluster
  output: V_profiles
  actor: machine
  description: "Display average daily profile for each cluster as line
graph"

artifact V_profiles : visualisation(F_cluster_profile, D_cluster)
  layout: "time axis (hour of day)"
  form: "line graphs (one per cluster)"
  encoding: "x-position: time within day; y-position: average measurement
    value; color: cluster identity matching calendar colors"
  description: "Line graphs showing characteristic temporal patterns for
    each cluster"

transform T_interpret :
  intent: abstract
  manner: "interpret cluster meanings"
  input: V_calendar, V_profiles, D_cluster, F_cluster_profile

```

```

        output: P_patterns
        actor: human
        description: "Interpret cluster patterns: identify behavioral meaning
                    of each cluster type"

    artifact P_patterns : pattern(D_cluster, F_cluster_profile)
        representation form: "textual labels and descriptions"
        description: "Interpreted meanings of daily patterns (e.g., 'typical
                    weekday', 'weekend', 'holiday', 'summer Friday', 'exceptional event
                    ')'"

    transform T_assess_clusters :
        intent: assess
        manner: "evaluate cluster quality and interpretability"
        input: V_calendar, V_profiles, P_patterns, D_cluster
        output: cluster_assessment
        actor: human
        description: "Assess whether clusters provide meaningful decomposition:
                    patterns are interpretable, clusters well-separated, standard vs.
                    exceptional days identified"

    artifact cluster_assessment : knowledge(D_cluster)
        representation form: "quality judgment"
        description: "Assessment of cluster quality: interpretability,
                    separation, coverage of pattern types, and whether refinement
                    with adjusted parameters is needed"

    if cluster_assessment indicates refinement needed:
        then:
            transform T_adjust :
                intent: generate-knowledge
                manner: "adjust clustering parameters based
                        on assessment"
                input: cluster_assessment, V_calendar,
                       V_profiles, S_clustering
                output: S_clustering'
                actor: human
                description: "Adjust clustering parameters:
                            modify number of clusters, select
                            different distance measure, or change
                            time interval focus"

                artifact S_clustering' : specification
                    representation form: "parameter settings"
                    description: "Updated clustering parameters
                                after analyst refinement"

                assign:
                    S_clustering := S_clustering'
            else:
                exit loop L1
        end loop L1

```

```

transform T_synthesize :
  intent: generate-knowledge
  manner: "formulate statements about temporal patterns"
  input: P_patterns, V_calendar, V_profiles, cluster_assessment
  output: K1
  actor: human
  description: "Synthesize findings: document discovered patterns, their temporal
    distribution, correlations with external events, and exceptional occurrences
  "

artifact K1 : knowledge(P_patterns)
  representation form: "statements and explanations"
  description: "Understanding of temporal patterns: standard daily patterns
    identified, their distribution over week and year, correlation with
    calendar events, exceptional patterns and their causes"

```

C.2 Dynamic Network Exploration workflow

Source: Stef van den Elzen, Danny Holten, Jorik Blaas, and Jarke J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration.

IEEE Transactions on Visualization and Computer Graphics, 22(1):1–10, January 2016. doi: 10.1109/TVCG.2015.2468078

Workflow summary:

The paper presents a visual analytics approach for exploring dynamic networks by treating network snapshots as points in high-dimensional space and projecting them to 2D for visualization. The workflow processes timestamped network activities through four main stages:

1. **Discretisation** creates temporal snapshots using sliding time windows with overlap to smooth temporal transitions and avoid missing patterns.
2. **Vectorisation and normalization** converts each snapshot into a feature vector, with optional normalization to emphasize specific patterns.
3. **Dimensionality reduction** projects high-dimensional vectors to 2D, preserving similarity relationships between snapshots.
4. **Visualisation and interaction** presents two coordinated views: a projection view showing snapshots as coloured points (revealing clusters, outliers, and trajectories) and a network view displaying detailed topology of selected snapshots.

Through interactive exploration, analysts identify stable states (uniform-coloured clusters), recurring states (multi-coloured clusters), outlier topologies (isolated points), and

transitions between states. The method is demonstrated on artificial networks and real-world high-school contact patterns, revealing daily rhythms, break periods, and network evolution.

Figure 3 provides a diagrammatic representation of the workflow.

ATWL representation:

```

workflow reducing-snapshots-to-points
  template: define-unit (temporal) → characterise →
           contextualise (projection-based) → visualise → abstract →
           generate-knowledge
  description: "Explore dynamic network evolution by projecting snapshots to
              2D points, revealing stable states, recurring patterns, and transitions"

# =====
# INPUT: Timestamped network activity data
# =====

artifact event_log : entities
  origin: given
  internal structure: elementary
  embedment: {set, time}
  features:
    - id: activity
      value structure: vector
      value type: reference
      description: "Source-target node pairs with timestamps"
  description: "Timestamped network activities involving node pairs"

# =====
# DISCRETIZATION: Create network snapshots
# =====

transform T_discretize :
  intent: define-unit
  manner: "time-partitioning with sliding windows"
  input: event_log
  output: network_snapshots
  actor: machine
  description: "Discretize activity stream into snapshots using overlapping
              time windows"

artifact network_snapshots : entities
  internal structure: formation
  embedment: time
  features:
    - id: snapshot_time
      value structure: atomic
      value type: temporal
      description: "Timestamp of snapshot"
    - id: network_structure

```

```

    value structure: relational configuration
    value type: {reference, numeric}
    description: "Network topology with edge weights"
description: "Temporal sequence of network snapshots, each aggregating
activities over a time window"

# =====
# VECTORIZATION: Convert snapshots to feature vectors
# =====

transform T_vectorize :
    intent: characterise
    manner: "encode network structure as vector"
    input: network_snapshots
    output: snapshot_vectors
    actor: machine
    description: "Convert snapshots to high-dimensional vectors representing
network structure"

artifact snapshot_vectors : feature(network_snapshots)
    value structure: vector
    value type: numeric
    description: "High-dimensional vectors encoding network topology"

# =====
# NORMALIZATION (optional): Apply scaling
# =====

transform T_normalize :
    intent: characterise
    manner: "scaling"
    input: snapshot_vectors
    output: normalized_vectors
    actor: machine
    description: "Apply normalization to emphasize specific patterns"

artifact normalized_vectors : feature(network_snapshots)
    value structure: vector
    value type: numeric
    description: "Scaled vectors for improved projection"

# =====
# DIMENSIONALITY REDUCTION: Project to 2D
# =====

transform T_project :
    intent: contextualise
    manner: "projection by dimensionality reduction"
    input: network_snapshots, normalized_vectors
    output: projection_space, snapshot_arrangement
    actor: machine
    description: "Reduce to 2D preserving similarity relationships"

```

```

artifact projection_space : entities
  internal structure: elementary
  features:
    - id: dimensionality
      value structure: atomic
      value type: numeric
      description: "Number of spatial dimensions (2)"
  description: "Two-dimensional projection space serving as
    reference frame for snapshot arrangement"

artifact snapshot_arrangement : arrangement(network_snapshots)
  context: projection_space
  principle: "dimensionality reduction preserving similarity"
  description: "2D positioning where proximity indicates structural similarity"

# =====
# VISUALIZATION: Create linked views
# =====

transform T_visualize_projection :
  intent: visualise
  manner: "scatterplot with temporal encoding"
  input: snapshot_arrangement
  output: projection_view
  actor: machine
  description: "Visualize snapshots as colored points with
    connecting lines"

artifact projection_view : visualisation(snapshot_arrangement)
  layout: "2D scatterplot"
  form: "colored points with connecting lines"
  encoding: "position from snapshot_arrangement; color by time;
    lines connect consecutive snapshots"
  description: "Interactive scatterplot revealing clusters, outliers, and
    temporal trajectories"

transform T_visualize_network :
  intent: visualise
  manner: "node-link diagram"
  input: network_snapshots
  output: network_view
  actor: machine
  description: "Visualize detailed network structure for selected snapshot"

artifact network_view : visualisation(network_snapshots)
  layout: "graph layout"
  form: "node-link diagram"
  encoding: "nodes as circles; edges as lines; node color by attributes"
  description: "Detailed topology view linked to projection selection"

# =====

```

```

# EXPLORATION: Identify patterns through interaction
# =====

transform T_explore :
  intent: abstract
  manner: "perception-and-interpretation with linked exploration"
  input: projection_view, network_view
  output: identified_patterns
  actor: hybrid
  description: "Identify network states through interactive exploration of
    coordinated views"

artifact identified_patterns : pattern(network_snapshots, snapshot_arrangement)
  representation form: "labeled clusters and trajectories"
  description: "Network states: stable states, recurring states, outliers,
    and transitions"

# =====
# KNOWLEDGE GENERATION: Synthesize insights
# =====

transform T_generate_knowledge :
  intent: generate-knowledge
  manner: "formulate-statements"
  input: identified_patterns
  output: evolution_insights
  actor: human
  description: "Synthesize understanding of network evolution from identified
    states and transitions"

artifact evolution_insights : knowledge(identified_patterns)
  representation form: "statements and explanations"
  description: "Understanding of network evolution: stable states, recurring
    patterns, outliers, and temporal dynamics"

```

C.3 Visual Analysis of Mass Mobility Dynamics (MobilityGraphs)

Source: Tatiana von Landesberger, Felix Brodkorb, Philipp Roskosch, Natalia Andrienko, Gennady Andrienko, and Andreas Kerren. MobilityGraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):11–20, 2016. doi: 10.1109/TVCG.2015.2468111
doi: 10.1109/TVCG.2015.2468111

Workflow summary:

The MobilityGraphs workflow addresses visual analysis of mass mobility dynamics through combined spatial and temporal simplifications. Starting from time-varying flow data showing people’s presence in places and movements between them, the approach employs graph-based clustering to aggregate spatially close places with strong flows into regions.

This spatial simplification reduces visual clutter while preserving large-scale movement patterns. Temporal clustering then groups time steps with similar flow patterns, reducing the number of distinct situations requiring analysis. Results are presented through coordinated views: a calendar showing temporal cluster distribution, cluster thumbnails displaying simplified flow graphs, difference views for comparing situations, and geographic maps providing spatial context. Interactive parameter adjustment enables exploration at various abstraction levels. The approach revealed routine mobility patterns in Greater London (mono-centric structure with morning centre-directed flows and evening outward movements) and Abidjan (polycentric structure with three activity centres), showing correspondence between flow topology and transportation networks.

Figure 4 provides a diagrammatic representation of the workflow.

ATWL representation

```

workflow mobility-graphs
  template: loop(define-unit (spatial aggregation) → characterise (quality) →
    visualise → assess) →
    characterise (vectors) →
    loop(define-unit (temporal clustering) → characterise (representatives
  ) →
    visualise → assess) →
    visualise (additional) → abstract → generate-knowledge
  description: "Analyze mass mobility dynamics through combined spatial and
  temporal
  simplifications, revealing movement patterns occluded in traditional flow
  maps"

# =====
# INPUT: Time-varying flow data
# =====

artifact spatial_situations : entities
  origin: given
  internal structure: formation
  embedment: time
  features:
  - id: presence_and_flows
    value structure: vector
    value type: {reference, numeric}
    description: "Presence counts in places and directed flows with magnitudes"
  description: "Spatial situations at successive time steps: distribution of
  people
  across places and flows between them, represented as directed weighted
  graphs"

# =====
# SPECIFICATION: Spatial aggregation parameters
# =====

```

```

artifact S_spatial_params : specification
  origin: given
  representation form: "parameter settings"
  description: "Initial spatial aggregation parameters: proximity
    threshold, flow strength threshold"

# =====
# SPATIAL SIMPLIFICATION: Aggregate places and flows
# =====

loop L_spatial_refinement:
  purpose: "Iteratively adjust spatial aggregation parameters to find suitable
    abstraction level"
  until: "Spatial aggregates balance detail preservation with clutter reduction
    and reveal interpretable large-scale patterns"
  body:
    transform T_spatial_aggregate :
      intent: define-unit
      manner: "graph-based spatial clustering by proximity and flow strength"
      input: spatial_situations, S_spatial_params
      output: aggregated_situations, region_labels
      actor: machine
      description: "Aggregate spatially close places with strong flows into
        regions; aggregate flows between places into flows between regions"

    artifact aggregated_situations : entities
      internal structure: formation
      embedment: time
      features:
        - id: aggregated_flow_structure
          value structure: vector
          value type: {reference, numeric}
          description: "Regions with aggregated presence counts and flows"
      description: "Simplified spatial situations where places are grouped

into
    regions and flows are aggregated accordingly"

    artifact region_labels : feature(spatial_situations)
      value structure: atomic
      value type: categorical
      description: "Region membership for each place"

    transform T_compute_spatial_quality :
      intent: characterise
      manner: "compute clustering quality metrics"
      input: aggregated_situations, region_labels
      output: spatial_quality
      actor: machine
      description: "Compute spatial aggregation quality measures"

    artifact spatial_quality : feature(aggregated_situations)

```

```

value structure: list
value type: numeric
description: "Quality metrics: within-region flows, number of regions,
            average inter-region distance"

transform T_visualize_quality :
  intent: visualise
  manner: "heatmap of parameter combinations"
  input: spatial_quality
  output: quality_heatmap
  actor: machine
  description: "Display quality metrics for various parameter combinations"
"

artifact quality_heatmap : visualisation(spatial_quality)
  layout: "grid with distance and connectivity thresholds as axes"
  form: "colored cells"
  encoding: "position by parameter values; color by quality metrics"
  description: "Heatmap showing quality of spatial aggregation for
different
            parameter settings"

transform T_assess_spatial :
  intent: assess
  manner: "evaluate aggregation quality and interpretability"
  input: quality_heatmap, aggregated_situations, S_spatial_params
  output: spatial_assessment
  actor: human
  description: "Assess whether spatial aggregation provides
            suitable abstraction level and interpretable patterns;
            determine if parameter adjustment needed"

artifact spatial_assessment : knowledge(aggregated_situations)
  representation form: "quality judgment"
  description: "Assessment of spatial aggregation quality: balance between
            detail and simplicity, interpretability of regions, and whether
            parameter refinement is needed"

if spatial_assessment indicates parameter adjustment needed:
  then:
    transform T_adjust_spatial :
      intent: generate-knowledge
      manner: "select improved spatial parameters from
            quality landscape"
      input: spatial_assessment, quality_heatmap, S_spatial_params
      output: S_spatial_params'
      actor: human
      description: "Adjust proximity and flow strength thresholds
            based on quality heatmap and assessment"

    artifact S_spatial_params' : specification
      representation form: "parameter settings"

```

```

        description: "Updated spatial aggregation parameters"

    assign:
        S_spatial_params := S_spatial_params'

    else:
        exit loop L_spatial_refinement
end loop L_spatial_refinement

# =====
# FEATURE EXTRACTION: Prepare for temporal clustering
# =====

transform T_extract_vectors :
    intent: characterise
    manner: "extract flow magnitude vectors"
    input: aggregated_situations
    output: situation_vectors
    actor: machine
    description: "Represent each aggregated situation by vector of flow magnitudes
        between regions"

artifact situation_vectors : feature(aggregated_situations)
    value structure: vector
    value type: numeric
    description: "Feature vectors characterizing spatial situations by aggregated
        flow magnitudes"

# =====
# SPECIFICATION: Temporal clustering parameters
# =====

artifact S_temporal_params : specification
    origin: given
    representation form: "parameter settings"
    description: "Initial temporal clustering parameters: number
        of clusters"

# =====
# TEMPORAL SIMPLIFICATION: Cluster time steps
# =====

loop L_temporal_refinement:
    purpose: "Iteratively adjust temporal clustering to identify meaningful pattern
        groups"
    until: "Temporal clusters reveal interpretable mobility patterns across time
        cycle"
    body:
        transform T_temporal_cluster :
            intent: define-unit
            manner: "cluster by similarity of spatial situations"
            input: aggregated_situations, situation_vectors, S_temporal_params

```

```

output: temporal_clusters, time_labels
actor: hybrid
description: "Group time steps with similar spatial situations into
temporal clusters"

artifact temporal_clusters : entities
internal structure: group/cluster
embedment: set
features:
- id: cluster_size
value structure: atomic
value type: numeric
description: "Number of time steps in cluster"
description: "Groups of time steps with similar flow patterns"

artifact time_labels : feature(aggregated_situations)
value structure: atomic
value type: categorical
description: "Temporal cluster membership for each time step"

transform T_compute_representatives :
intent: characterise
manner: "aggregate situations within clusters"
input: temporal_clusters, aggregated_situations
output: representative_situations
actor: machine
description: "Compute representative spatial situation for each temporal
cluster by averaging flows"

artifact representative_situations : feature(temporal_clusters)
value structure: vector
value type: {reference, numeric}
description: "Average flow patterns representing each temporal cluster"

transform T_visualize_calendar :
intent: visualise
manner: "calendar grid with color-coded clusters"
input: time_labels, aggregated_situations
output: calendar_view
actor: machine
description: "Display temporal cluster distribution in calendar layout"

artifact calendar_view : visualisation(time_labels)
layout: "calendar grid (days as rows, hours as columns)"
form: "colored cells"
encoding: "position by time coordinates; color by temporal cluster
membership"
description: "Calendar view showing temporal distribution of clusters"

transform T_visualize_thumbnails :
intent: visualise
manner: "small multiples of flow graphs"

```

```

    input: representative_situations, temporal_clusters
    output: thumbnail_view
    actor: machine
    description: "Display representative spatial situation for each temporal
        cluster as flow graph thumbnail"

    artifact thumbnail_view : visualisation(representative_situations,
        temporal_clusters)
        layout: "sequential or similarity-based positioning"
        form: "node-link diagrams with geographic positioning"
        encoding: "node position by geography; node size by presence count;
            edge width by flow magnitude; color by temporal cluster"
        description: "Overview of flow patterns for all temporal clusters"

    transform T_assess_temporal :
        intent: assess
        manner: "evaluate clustering quality and pattern interpretability"
        input: calendar_view, thumbnail_view, S_temporal_params
        output: temporal_assessment
        actor: human
        description: "Assess whether temporal clustering reveals
            interpretable mobility patterns and differentiation across
            time cycle; determine if parameter adjustment needed"

    artifact temporal_assessment : knowledge(temporal_clusters)
        representation form: "quality judgment"
        description: "Assessment of temporal clustering quality:
interpretability
            of patterns, temporal differentiation, and whether parameter
            refinement is needed"

    if temporal_assessment indicates refinement needed:
        then:
            transform T_adjust_temporal :
                intent: generate-knowledge
                manner: "adjust number of temporal clusters"
                input: temporal_assessment, calendar_view,
                    thumbnail_view, S_temporal_params
                output: S_temporal_params'
                actor: human
                description: "Adjust number of temporal clusters based on
                    assessment of pattern quality"

                artifact S_temporal_params' : specification
                    representation form: "parameter settings"
                    description: "Updated temporal clustering parameters"

                assign:
                    S_temporal_params := S_temporal_params'
            else:
                exit loop L_temporal_refinement
        end loop L_temporal_refinement

```

```

# =====
# ADDITIONAL VISUALIZATION: Difference and map views
# =====

transform T_visualize_differences :
  intent: visualise
  manner: "difference graphs with change encoding"
  input: representative_situations, temporal_clusters
  output: difference_view
  actor: machine
  description: "Display differences between selected spatial situations with
    color-coded changes"

artifact difference_view : visualisation(representative_situations)
  layout: "graph layout with geographic positioning"
  form: "node-link diagram with change encoding"
  encoding: "color by relative change; node size by initial presence; edge width
    by initial flow magnitude"
  description: "Detailed comparison showing flow and presence changes between
    two selected situations"

transform T_visualize_maps :
  intent: visualise
  manner: "geographic overlay"
  input: region_labels, aggregated_situations
  output: map_view
  actor: machine
  description: "Display spatial aggregates in geographic context"

artifact map_view : visualisation(aggregated_situations, region_labels)
  layout: "geographic map"
  form: "colored regions"
  encoding: "spatial position by geography; color by region identity"
  description: "Map showing geographic extent of spatial aggregates"

# =====
# EXPLORATION: Identify mobility patterns
# =====

transform T_explore_patterns :
  intent: abstract
  manner: "identify mobility patterns through coordinated views"
  input: calendar_view, thumbnail_view, difference_view, map_view
  output: mobility_patterns
  actor: hybrid
  description: "Identify mobility patterns through interactive exploration:
    daily/weekly cycles, center-directed flows, directional asymmetries,
    correspondence to geography"

artifact mobility_patterns : pattern(temporal_clusters, representative_situations)
  representation form: "labeled patterns with descriptions"

```

```

    description: "Identified mobility patterns: morning center-directed flows,
    evening outward flows, weekly cycles, quiet periods, city structure
    (mono/polycentric), correspondence to transportation networks"

# =====
# KNOWLEDGE GENERATION: Synthesize insights
# =====

transform T_generate_insights :
    intent: generate-knowledge
    manner: "formulate-statements"
    input: mobility_patterns, temporal_clusters, representative_situations
    output: mobility_knowledge
    actor: human
    description: "Synthesize understanding of mobility dynamics including routine
    behaviors, city structure, and temporal rhythms"

artifact mobility_knowledge : knowledge(mobility_patterns)
    representation form: "statements and explanations"
    description: "Understanding of mass mobility: routine movement patterns, spatial
    organization (activity centers and residential areas), temporal cycles,
    correspondence between flow topology and geographic features"

```

C.4 EventFlow workflow

Source: Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. Temporal event sequence simplification. *IEEE Transactions on Visualization and Computer Graphics*, 19:2227–2236, 2013 doi: 10.1109/TVCG.2013.200

Workflow summary:

The workflow iteratively simplifies large temporal event sequence datasets (such as EHRs or play-by-play logs) in EventFlow to balance visual comprehensibility with analytic fidelity. Starting from given event records, analysts specify sentinel events and time windows that define a relative-time context; EventFlow then aligns records, aggregates similar sequences into a tree, and visualises an overview plus individual timelines. Visual complexity metrics (e.g., number of visual elements and average aggregation level) together with human judgment are used to assess whether the display remains too cluttered. If further simplification is needed, analysts use interactive filters (by record, category, time, and attributes) and transformation-based operations (interval merging, category merging, marker insertion, and pattern-based Find & Replace) to rewrite the sequences into more compact, semantically meaningful representations. After each simplification, the system realigns and reaggregates the modified records, and this loop continues until the display is both interpretable and sufficient to answer the study questions, at which point analysts interpret domain patterns and formulate explicit findings and recommendations.

Figure 5 provides a diagrammatic representation of the workflow.

ATWL representation:

```
workflow event-sequence-simplification
  template: loop(contextualise (align) → define-unit (aggregate) →
                visualise → characterise (complexity) → assess →
                define-unit (simplify, conditional)) →
                abstract → generate-knowledge
  description: "Iterative simplification of temporal event sequence datasets
  using EventFlow: aligning, aggregating, and repeatedly simplifying
  event sequences until the display is compact yet retains information
  needed to answer study questions"

artifact D_records : entities
  origin: given
  internal structure: sequence
  embedment: set
  features:
    - id: event_data
      value structure: vector
      value type: {categorical, temporal, numeric}
      description: "Event categories, timestamps, and attributes"
  description: "Temporal event sequences, one per record, from EHR or
  other temporal logs"

artifact K_questions : knowledge(D_records)
  origin: given
  representation form: "study questions and analytic goals"
  description: "Analyst questions guiding simplification and determining
  adequacy of simplified representation"

artifact alignment_spec : specification
  origin: given
  representation form: "sentinel events and time window parameters"
  description: "Analyst-specified sentinel event(s) and relative time
  window for alignment"

loop L_simplify:
  purpose: "Iteratively align, aggregate, visualise, assess complexity, and
  simplify event sequences until visualization is interpretable and
  adequate for hypothesis generation and testing"
  until: "Aggregated display is compact and interpretable,
  and remaining events and records sufficiently support
  analytical questions"
  body:
    transform T_align :
      intent: contextualise
      manner: "relative-time-alignment with computed context"
      input: D_records, alignment_spec
      output: D_rel_time_axis, A_aligned
      actor: hybrid
      description: "Create relative-time context centered at sentinel events"
```

and arrange sequences by transforming absolute to relative times"

```
artifact D_rel_time_axis : entities
  internal structure: sequence
  embedment: time
  features:
    - id: relative_time
      value structure: atomic
      value type: temporal
      description: "Time coordinate relative to alignment event"
  description: "Computed relative time axis serving as context for
    aligned event sequences"

artifact A_aligned : arrangement(D_records)
  context: D_rel_time_axis
  principle: "relative-time alignment at sentinel events"
  description: "Arrangement of records and events on relative time axis
    centered at key events"

transform T_aggregate :
  intent: define-unit
  manner: "hierarchical aggregation by event sequence similarity"
  input: D_records, A_aligned
  output: D_agg_tree
  actor: machine
  description: "Group records with identical or similar aligned
    event sequences into branches of an aggregation tree
    positioned along the relative time axis"

artifact D_agg_tree : entities
  internal structure: formation
  features:
    - id: branch_size
      value structure: atomic
      value type: numeric
      description: "Number of records aggregated at each branch"
    - id: branch_time_span
      value structure: atomic
      value type: temporal
      description: "Temporal extent of each branch on the
        relative time axis, inherited from alignment"
  description: "Tree structure capturing how records share
    event subsequences; branches positioned along the
    relative time axis from alignment"

transform T_visualise_overview :
  intent: visualise
  manner: "aggregated and individual timeline views"
  input: D_agg_tree, A_aligned
  output: V_overview, V_records
  actor: machine
  description: "Render aggregated tree display and synchronized
```

```

individual record display"

artifact V_overview : visualisation(D_agg_tree)
  layout: "branched aggregation tree aligned to relative
    time axis"
  form: "vertical coloured bars"
  encoding: "branch position from D_agg_tree; bar height:
    number of records; color: event categories"
  description: "Population-level overview of event sequence
    patterns and their frequencies"

artifact V_records : visualisation(D_records, A_aligned)
  layout: "stacked aligned timelines"
  form: "horizontal event glyphs and intervals"
  encoding: "horizontal position from A_aligned; shape and color: event
    categories and durations; one row per record"
  description: "Detail view of individual event sequences for
    intra-record analysis"

transform T_compute_complexity :
  intent: characterise
  manner: "compute structural complexity metrics"
  input: D_agg_tree
  output: F_complexity
  actor: machine
  description: "Compute structural complexity metrics of the
    aggregation tree that determine visual complexity
    of the display"

artifact F_complexity : feature(D_agg_tree)
  value structure: list
  value type: numeric
  description: "Structural complexity metrics: number of
    branches, distinct event categories, average
    aggregation depth"

transform T_assess_simplification :
  intent: assess
  manner: "evaluate display complexity and adequacy"
  input: V_overview, V_records, F_complexity, K_questions
  output: simplification_assessment
  actor: human
  description: "Assess whether complexity is acceptable and display
    contains sufficient information to answer study questions"

artifact simplification_assessment : knowledge(V_overview, F_complexity)
  representation form: "quality judgment"
  description: "Assessment of visualization adequacy: balance between
    clutter and information, sufficiency for analytical goals, need
    for further simplification"

if simplification_assessment indicates further simplification needed:

```

```

then:

  transform T_filter :
    intent: define-unit
    manner: "filter by record selection and event criteria"
    input: D_records, K_questions, V_overview, V_records
    output: D_records_f
    actor: hybrid
    description: "Remove unneeded records and events through
      interactive filters: record selection, category filters,
      time windows, attribute-based queries"

  artifact D_records_f : entities
    internal structure: sequence
    embedment: set
    description: "Subset of records and events after filter-based
      simplification"

  transform T_restructure :
    intent: define-unit
    manner: "transform sequences through merging and substitution"
    input: D_records_f, K_questions
    output: D_records'
    actor: hybrid
    description: "Transform event sequences: merge intervals and
      categories, insert markers, apply pattern-based replacements"
"

  artifact D_records' : entities
    internal structure: sequence
    embedment: set
    description: "Simplified event sequences after
transformation-based
      simplification"

  assign:
    D_records := D_records'

  else:
    exit loop L_simplify
end loop L_simplify

transform T_identify_patterns :
  intent: abstract
  manner: "interpret domain-specific temporal patterns"
  input: V_overview, V_records, D_records, K_questions
  output: P_domain_patterns
  actor: human
  description: "Interpret patterns in final simplified dataset relevant to
    study questions"

artifact P_domain_patterns : pattern(D_records)

```

```

representation form: "named temporal pattern types and descriptions"
description: "Named temporal patterns and behaviors identified in simplified
event sequences"

transform T_generate_knowledge :
  intent: generate-knowledge
  manner: "formulate-statements"
  input: P_domain_patterns, V_overview, V_records, K_questions
  output: K_findings
  actor: human
  description: "Formulate study conclusions, recommendations, and narratives
based on identified temporal patterns"

artifact K_findings : knowledge(P_domain_patterns)
  representation form: "statements, explanations, illustrative figures"
  description: "Documented insights about temporal behaviors: evidence of
practices, validation of classifications, factors influencing outcomes"

```

C.5 EventAction: temporal event sequence recommendation

Source:

F. Du, C. Plaisant, N. Spring and B. Shneiderman,
 "EventAction: Visual analytics for temporal event sequence recommendation," 2016
 IEEE Conference on Visual Analytics Science and Technology (VAST), Baltimore, MD,
 USA, 2016, pp. 61-70,
 doi: 10.1109/VAST.2016.7883512.

Workflow summary

This paper presents EventAction, a visual analytics system for prescriptive analytics on temporal event sequences. The workflow helps analysts recommend actions to improve outcomes by leveraging historical records.

Review Current Record. The analyst loads the current subject's temporal event sequence and reviews it as a timeline table showing event categories over time periods.

Find Similar Archived Records. The system computes similarity scores between the current record and each archived record based on event sequence feature distances. The analyst examines the similarity distribution and interactively selects a cohort of similar records by adjusting a similarity threshold, guided by indicators including cohort size, proportion with the desired outcome, and average similarity.

Explore Outcomes and Recommendations. The system computes outcome distributions relative to baseline, event-outcome correlations per category, and aggregated temporal activity patterns for the cohort and the desired-outcome subgroup. Coordinated views display outcome probability bars, per-category correlation charts with trend encoding, and an activity summary integrated with the current record's timeline — filterable by overall cohort, desired-outcome subgroup, and distinguishing activities. The analyst identifies which event categories are most associated with the desired outcome and when

they typically occur.

Plan Specification and Iterative Tuning. The analyst creates an initial action plan by specifying planned events by category and time period, guided by identified recommendation patterns. The system recomputes similarity with the extended record and updates outcome probability estimates. The analyst assesses whether the plan achieves sufficient probability improvement; if not, they refine the plan based on outcome feedback and recommendation patterns. This cycle continues until the analyst is satisfied with the plan's estimated impact.

Finalize Guidance. The analyst integrates the data-driven action plan with domain expertise to formulate actionable guidance with associated outcome probability estimation.

Figure 6 provides a diagrammatic representation of the workflow.

ATWL representation

```
workflow EventActionPrescriptiveAnalytics
  template: visualise →
    characterise → visualise → define-unit →
    characterise → visualise → abstract →
    generate-knowledge (specify) →
    loop(characterise → visualise → assess →
      generate-knowledge (refine)) →
    generate-knowledge

  description: "Visual analytics workflow for prescriptive temporal event
    sequence recommendation: finding similar archived records, exploring
    potential outcomes, reviewing recommended actions, and iteratively
    tuning a personalized action plan with outcome feedback"

  # =====
  # GIVEN ARTIFACTS
  # =====

  artifact current_record : entities
    origin: given
    internal structure: sequence
    description: "Current subject's temporal event sequence with
      events categorized by type and positioned in time periods"

  artifact archived_records : entities
    origin: given
    internal structure: elementary
    embedment: set
    description: "Collection of historical records, each containing
      a temporal event sequence with known outcome"

  artifact desired_outcome : specification
    origin: given
    representation form: "outcome category identifier"
```

```

    description: "User's target outcome to optimize for in the
        action plan"

# =====
# PHASE 1: REVIEW CURRENT RECORD
# =====

transform T_visualize_current :
    intent: visualise
    manner: "render timeline table with event categories as rows
        and time periods as columns, encoding event frequency by
        square size"
    input: current_record
    output: current_viz
    actor: machine
    description: "Display the current record's event sequence for
        initial review and orientation"

artifact current_viz : visualisation(current_record)
    layout: "matrix grid (event categories as rows, time periods
        as columns)"
    form: "sized squares"
    encoding: "row: event category; column: time period; square
        size: event frequency per cell"
    description: "Timeline of current record with event categories
        over time periods and square-size encoding for event
        frequency per cell"

# =====
# PHASE 2: FIND SIMILAR ARCHIVED RECORDS
# =====

transform T_compute_similarity :
    intent: characterise
    manner: "extract feature vectors from event sequences within
        the comparison window and compute pairwise distance-based
        similarity scores"
    input: current_record, archived_records
    output: similarity_scores
    actor: machine
    description: "Compute similarity between the current record
        and each archived record based on event sequence patterns"

artifact similarity_scores : feature(archived_records)
    value structure: atomic
    value type: numeric
    description: "Distance-based similarity scores between the
        archived records and current record"

transform T_visualize_similarity :
    intent: visualise
    manner: "show similarity distribution with range selection

```

```

        widget and summary indicators including cohort size,
        proportion with desired outcome, and average similarity"
input: archived_records, similarity_scores, desired_outcome
output: similarity_viz
actor: machine
description: "Display similarity score distribution for
interactive cohort selection"

artifact similarity_viz : visualisation(archived_records,
similarity_scores)
layout: "horizontal axis (similarity score)"
form: "histogram bars with range selector"
encoding: "x-position: similarity score; bar height: record
count; interactive range selector for cohort threshold;
summary indicators for cohort size, outcome proportion,
average similarity"
description: "Distribution of similarity scores with
interactive range selector and cohort summary indicators"

transform T_select_cohort :
intent: define-unit
manner: "interactively set similarity range based on
distribution shape, cohort size, and desired-outcome
proportion"
input: archived_records, similarity_scores, similarity_viz
output: similar_cohort
actor: human
description: "Select the cohort of similar archived records by
adjusting the similarity threshold"

artifact similar_cohort : entities
internal structure: elementary
embedment: set
description: "Subset of archived records selected as similar
to the current record based on user-specified similarity
range"

# =====
# PHASE 3: EXPLORE OUTCOMES AND RECOMMENDATIONS
# =====

transform T_analyze_cohort :
intent: characterise
manner: "calculate outcome probabilities relative to baseline;
compute correlation between event category frequency and
outcome probability; aggregate event prevalence and
typical frequency per time period for overall cohort and
desired-outcome subgroup"
input: similar_cohort, archived_records, desired_outcome
output: cohort_analysis
actor: machine
description: "Compute outcome distributions, event-outcome

```

correlations, and temporal activity patterns for the cohort"

```
artifact cohort_analysis : feature(similar_cohort)
  value structure: list
  value type: numeric
  description: "Outcome probability distributions compared to
    baseline, per-category event-outcome correlations, and
    aggregated temporal activity patterns for the cohort and
    desired-outcome subgroup"

transform T_visualize_recommendations :
  intent: visualise
  manner: "coordinated views showing outcome distributions vs.
    baseline, per-category correlation charts with trend
    encoding, and activity summary integrated with current
    record timeline, filterable by subgroup"
  input: current_record, similar_cohort, cohort_analysis,
    desired_outcome
  output: recommendation_viz
  actor: machine
  description: "Display outcome estimation, event-outcome
    correlations, and temporal activity patterns for
    recommendation exploration"

artifact recommendation_viz : visualisation(similar_cohort,
  cohort_analysis, current_record)
  layout: "coordinated panels: outcome distribution,
    per-category correlation charts, activity timeline"
  form: "bars, trend-encoded charts, and timeline cells"
  encoding: "outcome bars: height by probability with baseline
    comparison; correlation charts: bar height by strength,
    trend arrows; activity timeline: cell color and size by
    event prevalence; filterable by subgroup"
  description: "Outcome distribution comparing cohort to
    baseline; correlation charts showing event-outcome
    relationships with encoded strength; activity summary
    integrated with current record timeline, filterable by
    subgroup and distinguishing activities"

transform T_identify_recommendations :
  intent: abstract
  manner: "identify positively correlated event categories from
    correlation charts; recognise temporal activity patterns
    from desired-outcome subgroup; identify distinguishing
    activities between desired-outcome subgroup and overall
    cohort"
  input: recommendation_viz, cohort_analysis
  output: recommendation_patterns
  actor: human
  description: "Identify which event categories are most
    associated with the desired outcome and their recommended
```

```

        timing"

artifact recommendation_patterns : pattern(similar_cohort)
    representation form: "identified action recommendations with
        timing"
    description: "Key event categories correlated with the desired
        outcome and their recommended timing based on temporal
        activity patterns of successful archived records"

# =====
# PHASE 4: PLAN SPECIFICATION AND ITERATIVE TUNING
# =====

transform T_specify_initial_plan :
    intent: generate-knowledge
    manner: "specify planned events by category and time period,
        referencing activity patterns and correlation insights
        from the desired-outcome subgroup"
    input: recommendation_patterns, recommendation_viz,
        current_record
    output: action_plan
    actor: human
    description: "Create initial action plan guided by identified
        recommendation patterns"

artifact action_plan : specification
    representation form: "planned events per category and time
        period"
    description: "User-specified action plan defining planned
        events with their categories and time periods"

loop L_plan_tuning:
    purpose: "Refine the action plan based on immediate feedback
        on estimated outcome probability"
    until: "the user is satisfied with the plan's estimated impact
        on the probability of achieving the desired outcome"
    body:

        transform T_recompute_estimation :
            intent: characterise
            manner: "extend current record with planned events;
                recompute similarity with extended comparison
                window; update outcome probability distribution"
            input: current_record, archived_records, action_plan
            output: plan_estimation
            actor: machine
            description: "Recompute outcome estimation
                incorporating the current action plan"

        artifact plan_estimation : feature(current_record)
            value structure: vector
            value type: numeric

```

```

description: "Revised outcome probability distribution
reflecting the action plan's impact on similarity
matching and outcome estimation"

transform T_visualize_estimation :
  intent: visualise
  manner: "overlay updated outcome probabilities on
original distribution as distinct markers"
  input: plan_estimation, cohort_analysis
  output: estimation_viz
  actor: machine
  description: "Display updated outcome estimation
showing the plan's impact"

artifact estimation_viz :
  visualisation(plan_estimation, cohort_analysis)
  layout: "horizontal axis (outcome categories)"
  form: "bars with overlay markers"
  encoding: "bar height: original outcome probability; overlay
markers: plan-adjusted probability"
  description: "Outcome distribution showing original
and plan-adjusted probabilities side by side"

transform T_assess_plan :
  intent: assess
  manner: "compare estimated likelihood with and without
plan; judge whether further refinement is needed"
  input: estimation_viz, action_plan, plan_estimation
  output: plan_assessment
  actor: human
  description: "Evaluate whether the plan achieves
sufficient probability improvement for the
desired outcome"

artifact plan_assessment : knowledge(action_plan)
  representation form: "quality judgment"
  description: "Assessment of whether the action plan
provides satisfactory probability improvement
for the desired outcome"

if plan_assessment indicates further refinement needed:
  then:
    transform T_refine_plan :
      intent: generate-knowledge
      manner: "adjust planned events by adding,
removing, or retiming events, guided by
identified recommendation patterns and
outcome probability feedback"
      input: action_plan, plan_assessment,
recommendation_patterns, estimation_viz
      output: updated_action_plan
      actor: human

```

```

        description: "Refine the action plan based on
            outcome feedback and recommendation
            patterns"

    artifact updated_action_plan : specification
        representation form: "planned events per
            category and time period"
        description: "Refined action plan with
            adjusted event timing and composition"

    assign: action_plan := updated_action_plan
else:
    exit loop L_plan_tuning

end loop L_plan_tuning

# =====
# PHASE 5: FINALIZE GUIDANCE
# =====

transform T_finalize :
    intent: generate-knowledge
    manner: "integrate recommended action sequences and timing
        with outcome probability estimation and domain knowledge"
    input: action_plan, plan_assessment, recommendation_patterns
    output: final_guidance
    actor: human
    description: "Finalize action plan and synthesize actionable
        guidance combining data-driven recommendations with
        domain expertise"

artifact final_guidance : knowledge(current_record)
    representation form: "actionable guidance with supporting
        evidence"
    description: "Finalized action plan with recommended event
        sequences and timing, associated outcome probability
        estimation, and domain-informed rationale"

```

C.6 Extracting significant places from trajectories

Source: Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), pages 161–170, 2011 doi: 10.1109/VAST.2011.6102454

Workflow summary

The paper presents a visual analytics procedure for analyzing movement data to determine significant places based on recurring events. The procedure addresses problems where

relevant places have arbitrary shapes and sizes and must be delineated by processing movement data rather than selected from predefined areas.

The workflow consists of four main steps applied iteratively:

Step 1: Event extraction - Relevant movement events (m-events) are identified from trajectories using dynamic attributes representing movement characteristics (speed, direction, acceleration) and relations to spatio-temporal context. Interactive visual query tools enable specification of event-defining conditions.

Step 2: Place determination through clustering - Density-based clustering identifies places where events occur repeatedly. Two-stage clustering may be applied: first, spatio-temporal clustering filters occasional events by grouping events concentrated in both space and time; second, spatial clustering unites spatio-temporal clusters sharing spatial positions. A custom distance function accounts for spatial distance, temporal distance, and thematic attributes (particularly movement direction). Interactive parameter adjustment and visual assessment guide refinement until spatially coherent, interpretable clusters emerge. Spatial buffers around clusters define relevant places.

Step 3: Spatio-temporal aggregation - Events and trajectories are aggregated by places and time intervals, producing time series of counts and statistics (visit counts, visitor counts, durations, speeds, directions). For trajectory aggregation, flows between place pairs are also computed.

Step 4: Analysis - Aggregated data are explored through coordinated interactive visualizations to identify temporal patterns of event occurrences and movements between places.

The procedure was demonstrated on traffic congestion analysis (Milan car trajectories) and air traffic analysis (French flight data), revealing temporal patterns and spatial organization of movements.

Figure 7 provides a diagrammatic representation of the workflow.

ATWL representation

```
workflow events-to-places
  template: characterise → define-unit (extract events) →
           loop(define-unit(two-stage clustering) →
                visualise → assess) →
           define-unit (delineate places) →
           characterise (aggregate) → visualise → abstract →
           generate-knowledge
  description: "Extract significant places from movement data by identifying
              recurring events, clustering them spatially and temporally, and analyzing
              spatio-temporal patterns"

# =====
# INPUT: Movement trajectories
# =====

artifact trajectories : entities
  origin: given
```

```

    internal structure: sequence
    embedment: {set, time}
    features:
      - id: movement_data
        value structure: vector
        value type: {reference, temporal, spatial}
        description: "Time-stamped positions with object identifiers"
    description: "Movement trajectories of tracked entities recorded over time"

# =====
# STEP 1: Event extraction from trajectories
# =====

transform T_compute_dynamic_attributes :
  intent: characterise
  manner: "compute movement characteristics and context relations"
  input: trajectories
  output: trajectory_attributes
  actor: machine
  description: "Compute dynamic attributes: instant and interval movement
    characteristics, cumulative measures, and relations to spatio-temporal
    context elements"

artifact trajectory_attributes : feature(trajectories)
  value structure: vector
  value type: numeric
  description: "Dynamic attributes including speed, direction, acceleration,
    distances to context elements, temporal and spatial neighborhoods"

transform T_extract_events :
  intent: define-unit
  manner: "interactive visual querying by attribute values"
  input: trajectories, trajectory_attributes
  output: movement_events
  actor: hybrid
  description: "Define and extract trajectory points or segments satisfying
    user-specified attribute value constraints through interactive visual
    query composition"

artifact movement_events : entities
  internal structure: elementary
  embedment: {set, space, time}
  features:
    - id: spatial_position
      value structure: atomic
      value type: spatial
      description: "Location in space"
    - id: temporal_position
      value structure: atomic
      value type: temporal
      description: "Time instant or interval"
    - id: thematic_attributes

```

```

    value structure: vector
    value type: numeric
    description: "Duration, extent, speed, direction, and attribute aggregates"
description: "Spatial events extracted from trajectories where specific
movement characteristics or context relations occur"

# =====
# STEP 2: Two-stage clustering to determine places
# =====

artifact S_clustering_params : specification
    origin: given
    representation form: "parameter settings"
    description: "Initial clustering parameters: spatial distance
        threshold, temporal distance threshold, density threshold,
        direction weight, and minimum cluster size"

loop L_cluster_refinement:
    purpose: "Iteratively apply two-stage clustering with parameter adjustment
        to identify spatially coherent, interpretable concentrations"
    until: "Clusters represent meaningful places where events occur repeatedly;
        occasional events filtered as noise"
    body:
        # STAGE 1: Spatio-temporal clustering
        transform T_cluster_spatiotemporal :
            intent: define-unit
            manner: "density-based clustering by spatial, temporal, and
                thematic proximity"
            input: movement_events, S_clustering_params
            output: st_clusters, st_labels
            actor: machine
            description: "Group events concentrated in space, time, and
                thematic attributes using density-based clustering"

        artifact st_clusters : entities
            internal structure: group/cluster
            embedment: set
            description: "Groups of events concentrated in space, time, and
                thematic attributes; filters out occasional scattered events"

        artifact st_labels : feature(movement_events)
            value structure: atomic
            value type: categorical
            description: "Spatio-temporal cluster membership or noise designation"

        transform T_filter_noise :
            intent: define-unit
            manner: "exclude noise events"
            input: movement_events, st_clusters, st_labels
            output: clustered_events
            actor: machine
            description: "Retain only events belonging to spatio-temporal

```

```

        clusters, excluding occasional scattered events"

artifact clustered_events : entities
    internal structure: elementary
    embedment: {set, space, time}
    description: "Events occurring as members of spatio-temporal
        concentrations"

# STAGE 2: Spatial clustering
transform T_cluster_spatial :
    intent: define-unit
    manner: "density-based clustering by spatial and thematic proximity"
    input: clustered_events, S_clustering_params
    output: spatial_clusters, s_labels
    actor: machine
    description: "Unite spatio-temporal clusters sharing spatial
        positions by clustering according to location and thematic
attributes"

artifact spatial_clusters : entities
    internal structure: group/cluster
    embedment: set
    description: "Groups of events concentrated spatially, potentially
        spanning multiple time periods"

artifact s_labels : feature(clustered_events)
    value structure: atomic
    value type: categorical
    description: "Spatial cluster membership or noise designation"

transform T_visualize_clusters :
    intent: visualise
    manner: "events on map and space-time cube colored by cluster"
    input: clustered_events, s_labels
    output: cluster_views
    actor: machine
    description: "Display events colored by cluster membership on map
        and in space-time cube"

artifact cluster_views : visualisation(clustered_events, s_labels)
    layout: "geographic map and space-time cube"
    form: "colored points"
    encoding: "position from event coordinates; color by cluster membership"
    description: "Spatial and spatio-temporal distribution of event clusters
"

transform T_assess_clusters :
    intent: assess
    manner: "evaluate cluster quality and coherence"
    input: cluster_views, spatial_clusters, S_clustering_params
    output: cluster_assessment
    actor: human

```

```

        description: "Assess whether clusters are spatially coherent,
            interpretable, and appropriately sized; determine if parameter
            adjustment needed"

    artifact cluster_assessment : knowledge(spatial_clusters)
        representation form: "quality judgment"
        description: "Assessment of clustering quality: spatial coherence,
            interpretability, appropriate size and extent, and whether
            parameter adjustment is needed"

    if cluster_assessment indicates parameter adjustment needed:
        then:
            transform T_adjust_params :
                intent: generate-knowledge
                manner: "adjust clustering parameters based on
                    visual assessment"
                input: cluster_assessment, cluster_views,
                    S_clustering_params
                output: S_clustering_params'
                actor: human
                description: "Adjust distance thresholds, density
                    parameters, and direction weights based on
                    cluster quality assessment"

                artifact S_clustering_params' : specification
                    representation form: "parameter settings"
                    description: "Updated clustering parameters after
                        analyst refinement"

                assign:
                    S_clustering_params := S_clustering_params'
            else:
                exit loop L_cluster_refinement
        end loop L_cluster_refinement

    transform T_delineate_places :
        intent: define-unit
        manner: "build spatial buffers around clusters"
        input: spatial_clusters, clustered_events
        output: places
        actor: machine
        description: "Construct spatial buffers or convex hulls around event
            clusters to delineate place boundaries"

    artifact places : entities
        internal structure: region
        embedment: space
        features:
            - id: spatial_extent
              value structure: atomic
              value type: spatial
              description: "Area boundary"

```

```

- id: characteristic_attributes
  value structure: vector
  value type: numeric
  description: "Prevailing movement directions and other characteristics"
description: "Significant places defined by spatial buffers around event
clusters, representing areas where events occur repeatedly"

# =====
# STEP 3: Spatio-temporal aggregation
# =====

transform T_aggregate_events :
  intent: characterise
  manner: "spatio-temporal aggregation by places and time intervals"
  input: movement_events, places
  output: place_event_aggregates
  actor: machine
  description: "Group events by places and time intervals; compute counts
and statistics of event attributes"

artifact place_event_aggregates : feature(places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Time series of event counts, object counts, and attribute
statistics for each place"

transform T_aggregate_trajectories :
  intent: characterise
  manner: "spatio-temporal aggregation by places and time intervals"
  input: trajectories, places
  output: place_visit_aggregates, place_flow_aggregates
  actor: machine
  description: "Group trajectory segments by places and time intervals;
compute visit statistics and flows between places"

artifact place_visit_aggregates : feature(places)
  value structure: vector
  value type: numeric
  representation form: "time series per place"
  description: "Time series of visit counts, visitor counts, and movement
statistics for each place"

artifact place_flow_aggregates : feature(places)
  value structure: relational configuration
  value type: {reference, numeric}
  representation form: "directed flows with time series"
  description: "Aggregate movements between pairs of places with time series
of counts, durations, speeds, and path lengths"

# =====
# STEP 4: Analysis of aggregated data

```

```

# =====
transform T_visualize_place_dynamics :
  intent: visualise
  manner: "temporal diagrams on map, time graphs, flow maps"
  input: places, place_event_aggregates, place_visit_aggregates,
        place_flow_aggregates
  output: dynamics_views
  actor: machine
  description: "Display time series as temporal diagrams positioned on map,
              time graphs with interactive selection, and flow maps"

artifact dynamics_views : visualisation(places, place_event_aggregates,
                                       place_visit_aggregates,
                                       place_flow_aggregates)
  layout: "geographic map with temporal diagrams, time graphs, flow maps, tables"
  form: "temporal bar diagrams, line graphs, directed arrows, colored cells"
  encoding: "diagram position from place location; bar heights and line
            positions from time series values; arrow widths from flow magnitudes;
            colors from movement directions or place identities"
  description: "Coordinated views showing spatial distribution of places,
              temporal dynamics of events and movements, and flows between places"

transform T_explore_patterns :
  intent: abstract
  manner: "identify temporal and spatial patterns"
  input: dynamics_views
  output: spatio_temporal_patterns
  actor: hybrid
  description: "Identify temporal patterns of event occurrences and movements
              through interactive exploration: peak times, recurring patterns, spatial
              variations, flow connectivity"

artifact spatio_temporal_patterns : pattern(places, place_event_aggregates,
                                           place_visit_aggregates,
                                           place_flow_aggregates)
  representation form: "categorized patterns with descriptions"
  description: "Identified patterns: temporal profiles of place usage,
              recurring patterns across time cycles, directional preferences,
              connectivity structures between places"

# =====
# KNOWLEDGE GENERATION: Synthesize insights
# =====

transform T_generate_knowledge :
  intent: generate-knowledge
  manner: "formulate-statements"
  input: spatio_temporal_patterns, places
  output: movement_knowledge
  actor: human
  description: "Synthesize understanding of movement dynamics and place

```

```
significance from identified patterns"
```

```
artifact movement_knowledge : knowledge(spatio_temporal_patterns)
  representation form: "statements and explanations"
  description: "Understanding of movement dynamics: significant place locations
    and characteristics, temporal patterns of event occurrences and movements,
    connectivity patterns between places"
```

C.7 Progressive clustering of trajectories

Source: Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia Andrienko, and Gennady Andrienko. Visually-driven analysis of movement data by progressive clustering. *Information Visualization*, 7:225–239, 07 2008. doi: 10.1057/PALGRAVE.IVS.9500183

Workflow summary

The paper presents progressive clustering as a method for visually-driven analysis of large trajectory datasets. Progressive clustering proceeds through iterative steps where simple, interpretable distance functions are applied sequentially, with visual assessment guiding refinement decisions. The workflow addresses four successive analytical questions through progressive refinement:

Phase 1: **Destination clustering with density adaptation** - Progressive clustering by trip destinations with iteratively decreasing sensitivity handles uneven spatial data density, identifying major destination regions across varying densities.

Phase 2: **Route analysis of selected subsets** - Examining major destination clusters by route similarity reveals patterns within destinations, such as prevalence of low-displacement trips.

Phase 3: **Pattern verification on complete dataset** - Observations from subset analysis lead to hypothesis testing: clustering ALL trajectories by start-end proximity verifies pattern prevalence across the complete dataset.

Phase 4: **Route analysis of refined subset** - After excluding verified patterns, clustering remaining trajectories by route similarity identifies route types characterized by spatial structure and frequency, revealing route diversity and repetition.

The progressive approach enables sophisticated analysis while maintaining interpretability, showing how the analyst's understanding evolves through exploration. The method was demonstrated on Milan car trajectory data, revealing movement patterns and route organization.

The workflow is schematically represented in Fig. 8.

ATWL representation

```
workflow progressive-trajectory-clustering
  template: loop(define-unit (destinations) → visualise → assess) →
```

```

        define-unit (select major) →
        loop(define-unit (routes: subset) → visualise → assess) →
        define-unit (verification) → visualise → assess →
        define-unit (exclude) → define-unit (routes: final) → visualise →
        abstract (route types) → abstract (synthesize) → generate-knowledge
description: "Iteratively cluster trajectories through sequence of analytical
questions, progressively refining understanding of movement patterns"

# =====
# INPUT: Complete movement trajectory dataset
# =====

artifact all_trajectories : entities
  origin: given
  internal structure: sequence
  embedment: {set, time}
  features:
    - id: movement_data
      value structure: vector
      value type: {reference, temporal, spatial}
      description: "Time-stamped positions with object identifiers,
speeds and directions"
  description: "Movement trajectories of tracked entities over time"

artifact analysed_trajectories : entities
  internal structure: sequence
  embedment: {set, time}
  features:
    - id: movement_data
      value structure: vector
      value type: {reference, temporal, spatial}
      description: "Time-stamped positions with object identifiers,
speeds and directions"
  description: "Subset of trajectories currently under analysis, initially
equal to all_trajectories"

# =====
# PHASE 1: Destination clustering with density adaptation
# =====

assign:
  analysed_trajectories := all_trajectories

artifact S_dest_sensitivity : specification
  origin: given
  representation form: "parameter settings"
  description: "Initial sensitivity parameter settings for destination
clustering"

loop L_destination_clustering:
  purpose: "Handle uneven spatial density by progressively clustering with
same distance function but decreasing sensitivity"

```

```

until: "Major destination regions identified across all density levels;
remaining trajectories are sparse"
body:
  transform T_cluster_destinations :
    intent: define-unit
    manner: "density-based clustering by destination similarity"
    input: analysed_trajectories, S_dest_sensitivity
    output: destination_clusters, cluster_labels
    actor: machine
    description: "Group trajectories with similar destinations using
current sensitivity parameter"

  artifact destination_clusters : entities
    internal structure: group/cluster
    embedment: set
    features:
      - id: cluster_size
        value structure: atomic
        value type: numeric
        description: "Number of trajectories in cluster"
    description: "Groups of trajectories with similar trip destinations"

  artifact cluster_labels : feature(analysed_trajectories)
    value structure: atomic
    value type: categorical
    description: "Cluster membership or noise designation"

  transform T_visualize_destinations :
    intent: visualise
    manner: "trajectories on map colored by cluster"
    input: analysed_trajectories, cluster_labels
    output: destination_map
    actor: machine
    description: "Display trajectories colored by cluster membership
with adjustable transparency"

  artifact destination_map : visualisation(analysed_trajectories,
cluster_labels)

    layout: "geographic map"
    form: "colored trajectory lines"
    encoding: "position from coordinates; color by cluster;
transparency adjustable"
    description: "Map showing spatial distribution and coherence of
destination clusters"

  transform T_assess_destinations :
    intent: assess
    manner: "evaluate cluster quality and coverage"
    input: destination_map, destination_clusters
    output: destination_assessment
    actor: human
    description: "Assess whether major destination regions are captured

```

with spatial coherence and interpretability"

```
artifact destination_assessment : knowledge(destination_clusters)
  representation form: "quality judgment"
  description: "Assessment of clustering quality: spatial coherence,
  interpretability, coverage of dense regions, and whether
  significant clusters identified for exclusion"

if destination_assessment indicates significant clusters identified:
  then:
    transform T_select_major_destinations :
      intent: define-unit
      manner: "interactive selection of interpretable clusters"
      input: destination_clusters, cluster_labels, destination_map
      output: selected_destination_clusters
      actor: human
      description: "Select sufficiently large spatially coherent
      clusters for exclusion from further density analysis"

    artifact selected_destination_clusters : entities
      internal structure: group/cluster
      embedment: set
      description: "Major destination clusters to exclude from
      further density-based analysis"

    transform T_exclude_major :
      intent: define-unit
      manner: "filter by excluding selected clusters"
      input: analysed_trajectories, selected_destination_clusters,
      cluster_labels
      output: remaining_trajectories
      actor: machine
      description: "Remove trajectories in identified clusters to
      focus on remaining unanalyzed data"

    artifact remaining_trajectories : entities
      internal structure: sequence
      embedment: {set, time}
      description: "Trajectories not yet assigned to interpretable
      destination clusters"

    transform T_decrease_sensitivity :
      intent: generate-knowledge
      manner: "decrease clustering sensitivity for next density level"
      input: destination_assessment, S_dest_sensitivity
      output: S_dest_sensitivity'
      actor: human
      description: "Reduce sensitivity to capture clusters
      in sparser density regions"

    artifact S_dest_sensitivity' : specification
      representation form: "parameter settings"
```

```

        description: "Decreased sensitivity parameter
                    for next clustering iteration"

    assign:
        analysed_trajectories := remaining_trajectories
        S_dest_sensitivity := S_dest_sensitivity'

    else:
        exit loop L_destination_clustering
end loop L_destination_clustering

# =====
# PHASE 2: Route analysis of selected subsets
# =====

transform T_select_for_route_analysis :
    intent: define-unit
    manner: "interactive selection from destination clusters"
    input: selected_destination_clusters
    output: major_destination_clusters
    actor: human
    description: "Select largest destination clusters for in-depth route analysis"

artifact major_destination_clusters : entities
    internal structure: group/cluster
    embedment: set
    description: "Largest destination clusters selected for detailed route
                similarity analysis"

loop L_route_analysis_subsets:
    purpose: "Examine routes within major destinations by analyzing each cluster
            separately"
    until: "All major destination clusters of interest have been examined
          for route patterns"
    body:
        transform T_select_one_destination :
            intent: define-unit
            manner: "interactive selection of single cluster"
            input: major_destination_clusters
            output: selected_subset
            actor: human
            description: "Select one destination cluster for route similarity
                        analysis"

        artifact selected_subset : entities
            internal structure: sequence
            embedment: {set, time}
            description: "Trajectories from one destination cluster for route
                        analysis"

        transform T_cluster_routes_subset :
            intent: define-unit

```

```

    manner: "density-based clustering by route similarity"
    input: selected_subset
    output: route_clusters_subset, route_labels_subset
    actor: machine
    description: "Group trajectories with similar spatial routes"

artifact route_clusters_subset : entities
    internal structure: group/cluster
    embedment: set
    description: "Groups of trajectories following similar paths within
        destination cluster"

artifact route_labels_subset : feature(selected_subset)
    value structure: atomic
    value type: categorical
    description: "Route cluster membership or noise designation"

transform T_visualize_routes_subset :
    intent: visualise
    manner: "trajectories on map colored by route cluster"
    input: selected_subset, route_labels_subset
    output: route_map_subset
    actor: machine
    description: "Display trajectories colored by route cluster"

artifact route_map_subset : visualisation(selected_subset,
    route_labels_subset)

    layout: "geographic map"
    form: "colored trajectory lines"
    encoding: "position from coordinates; color by route cluster"
    description: "Map showing route similarity patterns within destination
        cluster"

transform T_assess_routes_subset :
    intent: assess
    manner: "evaluate route patterns and low-displacement prevalence"
    input: route_map_subset, route_clusters_subset
    output: route_assessment_subset
    actor: human
    description: "Assess route patterns within destination cluster:
        prevalence of low-displacement trips, route similarity, pattern
        coherence, and whether more clusters need examination"

artifact route_assessment_subset : knowledge(route_clusters_subset)
    representation form: "quality judgment with observations"
    description: "Assessment of route patterns within destination cluster:
        prevalence of low-displacement trips, route coherence, and whether
        additional destination clusters require analysis"

end loop L_route_analysis_subsets

# =====

```

```

# PHASE 3: Pattern verification on complete dataset
# =====

transform T_verify_pattern_hypothesis :
  intent: define-unit
  manner: "density-based clustering by source and destination proximity"
  input: all_trajectories
  output: verification_clusters, verification_labels
  actor: machine
  description: "Test hypothesis generated from subset analysis by clustering
    ALL trajectories to assess pattern prevalence across complete dataset"

artifact verification_clusters : entities
  internal structure: group/cluster
  embedment: set
  description: "Clusters from verification analysis on complete dataset"

artifact verification_labels : feature(all_trajectories)
  value structure: atomic
  value type: categorical
  description: "Classification of all trajectories according to verified pattern"

transform T_visualize_verification :
  intent: visualise
  manner: "trajectories on map colored by pattern type"
  input: all_trajectories, verification_labels
  output: verification_map
  actor: machine
  description: "Display complete trajectory set colored by pattern classification"

artifact verification_map : visualisation(all_trajectories, verification_labels)
  layout: "geographic map"
  form: "colored trajectory lines"
  encoding: "position from coordinates; color by pattern classification"
  description: "Map showing spatial distribution of verified pattern across
    complete dataset"

transform T_assess_verification :
  intent: assess
  manner: "evaluate pattern prevalence and hypothesis confirmation"
  input: verification_map, verification_clusters
  output: pattern_verification
  actor: human
  description: "Assess pattern prevalence, spatial distribution, and
    confirmation or refutation of hypothesis"

artifact pattern_verification : knowledge(verification_clusters)
  representation form: "quantitative and qualitative assessment"
  description: "Assessment of pattern prevalence and distribution across
    complete dataset, confirming or refuting hypothesis from subset analysis"

# =====

```

```

# PHASE 4: Route analysis of refined subset
# =====

transform T_exclude_verified_pattern :
  intent: define-unit
  manner: "filter by excluding clusters"
  input: all_trajectories, verification_clusters, verification_labels
  output: refined_trajectories
  actor: machine
  description: "Exclude trajectories matching verified pattern to focus on
    complementary subset for further analysis"

artifact refined_trajectories : entities
  internal structure: sequence
  embedment: {set, time}
  description: "Trajectories from complementary subset after pattern exclusion"

transform T_cluster_routes_final :
  intent: define-unit
  manner: "density-based clustering by route similarity"
  input: refined_trajectories
  output: route_clusters_final, route_labels_final
  actor: machine
  description: "Group trajectories with similar spatial routes in refined subset"

artifact route_clusters_final : entities
  internal structure: group/cluster
  embedment: set
  description: "Groups of trajectories following similar paths in refined subset"

artifact route_labels_final : feature(refined_trajectories)
  value structure: atomic
  value type: categorical
  description: "Route cluster membership or noise designation"

transform T_visualize_routes_final :
  intent: visualise
  manner: "trajectories on map colored by route cluster"
  input: refined_trajectories, route_labels_final
  output: route_map_final
  actor: machine
  description: "Display trajectories colored by route cluster"

artifact route_map_final : visualisation(refined_trajectories, route_labels_final)
  layout: "geographic map"
  form: "colored trajectory lines"
  encoding: "position from coordinates; color by route cluster"
  description: "Map showing route patterns in refined subset"

transform T_identify_route_types :
  intent: abstract
  manner: "identify and characterize route types"

```

```

input: route_map_final, route_clusters_final
output: route_patterns
actor: human
description: "Identify route types and assess frequency of route repetition"

artifact route_patterns : pattern(route_clusters_final)
  representation form: "categorized route types with characteristics"
  description: "Identified route types characterized by spatial structure
    and frequency; assessment of route diversity vs. repetition"

# =====
# PATTERN SYNTHESIS: Integrate findings across all phases
# =====

transform T_synthesize_findings :
  intent: abstract
  manner: "synthesize findings across progressive clustering phases"
  input: destination_assessment, route_assessment_subset, pattern_verification,
    route_patterns
  output: movement_patterns
  actor: human
  description: "Integrate patterns identified across four analytical phases"

artifact movement_patterns : pattern(all_trajectories)
  representation form: "comprehensive findings with descriptions"
  description: "Integrated movement patterns: destination distribution and
    popularity, prevalence of low-displacement trips, route diversity and
    repetition, spatial organization of routes"

# =====
# KNOWLEDGE GENERATION: Formulate insights
# =====

transform T_generate_knowledge :
  intent: generate-knowledge
  manner: "formulate-statements"
  input: movement_patterns
  output: mobility_knowledge
  actor: human
  description: "Synthesize comprehensive understanding of movement dynamics
    from progressive clustering results"

artifact mobility_knowledge : knowledge(movement_patterns)
  representation form: "statements and explanations"
  description: "Comprehensive understanding of movement dynamics: spatial
    distribution of destinations, prevalence of different trip types by
    displacement, route diversity and common patterns, spatial organization
    of movement flows"

```

C.8 Human-Steered Topic Modelling

Source: J. Choo, C. Lee, C. K. Reddy and H. Park, "UTOPIAN: User-Driven Topic Modeling Based on Interactive Nonnegative Matrix Factorization," in IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 1992-2001, Dec. 2013, doi: 10.1109/TVCG.2013.212.

Workflow summary

The paper presents UTOPIAN, a visual analytics system for interactive topic modelling using Nonnegative Matrix Factorization (NMF). The workflow begins by applying NMF to decompose a term-document matrix into two representations: topics as weighted combinations of keywords, and documents as weighted combinations of topics. A modified t-SNE algorithm creates a 2D layout visualizing documents as points coloured by their topic assignments, with enhanced cluster separation for clarity.

The core of UTOPIAN is an iterative refinement loop where users assess topic quality and interact with the model through multiple mechanisms: keyword refinement adjusts weights of terms within topics; topic merging combines similar topics; topic splitting divides topics based on semantic differences; document-induced topic creation builds topics from exemplar documents; and keyword-induced topic creation generates topics from selected terms. Each interaction produces reference specifications (V and G matrices with supervision weights) that guide subsequent topic modelling through semi-supervised NMF, which regularizes the solution toward user specifications while maintaining fit to the data.

Through iterative exploration and refinement, analysts progressively improve topic quality and interpretability, discovering meaningful semantic structures in document collections. The approach was demonstrated on research papers, product reviews, and news-group data, revealing topic relationships and document organization.

A schematic representation of the workflow is provided in Fig. 9.

ATWL representation

```
workflow interactive-topic-modeling
  template: characterise (matrix representation) →
    define-unit (initial topics) →
    characterise (assign topics) →
    contextualise (projection) → visualise →
  loop(assess →
    generate-knowledge (specify refinement) →
    define-unit (semi-supervised) →
    characterise (assign topics) →
    contextualise → visualise) →
  abstract → generate-knowledge
  description: "Iteratively refine topic modeling through user interactions that
  guide semi-supervised matrix decomposition, progressively improving topic
  quality and interpretability"
```

```

# =====
# INPUT: Document corpus
# =====

artifact documents : entities
  origin: given
  internal structure: elementary
  embedment: set
  features:
    - id: text_content
      value structure: atomic
      value type: text
      description: "Document text"
  description: "Collection of text documents for topic analysis"

# =====
# PREPROCESSING: Matrix representation
# =====

transform T_create_matrix :
  intent: characterise
  manner: "bag-of-words encoding with weighting"
  input: documents
  output: term_document_matrix
  actor: machine
  description: "Represent documents as weighted term vectors with tf-idf
    and normalization"

artifact term_document_matrix : feature(documents)
  value structure: matrix
  value type: numeric
  description: "Term frequency representation of documents with weighting
    and normalization"

# =====
# INITIAL TOPIC MODELING: Unsupervised matrix decomposition
# =====

transform T_initial_topic_modeling :
  intent: define-unit
  manner: "matrix decomposition with non-negativity constraints"
  input: term_document_matrix
  output: topics, topic_keyword_weights, document_topic_weights
  actor: machine
  description: "Decompose term-document matrix into topics and their
    relationships to keywords and documents"

artifact topics : entities
  internal structure: elementary
  embedment: set
  features:
    - id: topic_id

```

```

    value structure: atomic
    value type: categorical
    description: "Topic identifier"
description: "Discovered topics representing semantic themes in corpus"

artifact topic_keyword_weights : feature(topics)
    value structure: vector
    value type: numeric
    description: "Representation of each topic as a vector of keyword weights"

artifact document_topic_weights : feature(documents)
    value structure: vector
    value type: numeric
    description: "Representation of each document as a vector of topic weights"

transform T_assign_topics :
    intent: characterise
    manner: "assign primary topic based on maximum weight"
    input: documents, document_topic_weights
    output: topic_assignments
    actor: machine
    description: "Assign each document to its dominant topic for visualization"

artifact topic_assignments : feature(documents)
    value structure: atomic
    value type: categorical
    description: "Primary topic assignment for each document"

# =====
# VISUALIZATION: Layout and display
# =====

transform T_compute_projection :
    intent: contextualise
    manner: "dimensionality reduction with cluster enhancement"
    input: documents, term_document_matrix, topic_assignments
    output: projection_space, document_layout
    actor: machine
    description: "Project documents to 2D preserving similarities while enhancing
        topic cluster separation"

artifact projection_space : entities
    internal structure: elementary
    features:
        - id: dimensionality
            value structure: atomic
            value type: numeric
            description: "Number of spatial dimensions (2)"
    description: "Two-dimensional projection space serving as
        reference frame for document layout"

artifact document_layout : arrangement(documents)

```

```

context: projection_space
principle: "dimensionality reduction preserving similarities with cluster
separation enhancement"
description: "2D positioning where proximity indicates similarity and topic
clusters are spatially separated"

transform T_visualize_topics :
  intent: visualise
  manner: "scatterplot with topic coloring and keyword summaries"
  input: document_layout, topic_assignments, topic_keyword_weights
  output: topic_view
  actor: machine
  description: "Display documents as colored points with topic keyword summaries
and interactive capabilities"

artifact topic_view : visualisation(document_layout, topic_assignments,
topic_keyword_weights)

  layout: "2D scatterplot"
  form: "colored points with topic labels"
  encoding: "position from document_layout; color by topic_assignments;
annotations show top keywords per topic"
  description: "Interactive view showing document distribution and topic structure
with keyword summaries"

# =====
# ITERATIVE REFINEMENT: User-guided improvement
# =====

loop L_refinement:
  purpose: "Iteratively refine topic model through user interactions that express
desired topic characteristics or structure"
  until: "Topics are interpretable, semantically coherent, and appropriately
structured for analytical goals"
  body:
    transform T_assess_topics :
      intent: assess
      manner: "evaluate semantic coherence and interpretability"
      input: topic_view, topic_keyword_weights, document_topic_weights
      output: topic_assessment
      actor: human
      description: "Assess topic quality: semantic coherence, interpretability
,
appropriate granularity, and coverage"

    artifact topic_assessment : knowledge(topics)
      representation form: "quality judgment"
      description: "Assessment of topic quality identifying needed
improvements:
unclear topics, overly similar topics, missing themes, inappropriate
keyword weights"

    if topic_assessment indicates refinement needed:

```

```

then:
  transform T_specify_refinement :
    intent: generate-knowledge
    manner: "formulate desired topic characteristics"
    input: topics, topic_keyword_weights, document_topic_weights,
           topic_assessment
    output: user_guidance
    actor: human
    description: "Specify desired changes through interactions:
                 keyword weight adjustments, topic combinations or divisions,
                 creation of new topics from exemplar documents or keywords"

  artifact user_guidance : specification
    representation form: "reference matrices (V, G) with
                         supervision weights (MW, MH)"
    description: "User-specified constraints defining
                 desired topic-keyword relationships (V),
                 document-topic relationships (G),
                 and supervision strengths (MW, MH) for
                 semi-supervised refinement"

  transform T_refine_topics :
    intent: define-unit
    manner: "semi-supervised matrix decomposition guided by user
            specifications"
    input: term_document_matrix, user_guidance
    output: topics', topic_keyword_weights', document_topic_weights'
    actor: machine
    description: "Recompute topic model incorporating user guidance
                 as soft constraints while maintaining fit to data"

  artifact topics' : entities
    internal structure: elementary
    embedment: set
    description: "Refined topics incorporating user guidance"

  artifact topic_keyword_weights' : feature(topics')
    value structure: vector
    value type: numeric
    description: "Updated keyword weights"

  artifact document_topic_weights' : feature(documents)
    value structure: vector
    value type: numeric
    description: "Updated topic weights"

  transform T_update_assignments :
    intent: characterise
    manner: "assign primary topic based on updated weights"
    input: documents, document_topic_weights'
    output: topic_assignments'
    actor: machine

```

```

        description: "Update document topic assignments based on
            refined model"

    artifact topic_assignments' : feature(documents)
        value structure: atomic
        value type: categorical
        description: "Updated primary topic assignments"

    transform T_update_projection :
        intent: contextualise
        manner: "incremental dimensionality reduction"
        input: documents, term_document_matrix, topic_assignments',
            document_layout
        output: document_layout'
        actor: machine
        description: "Update 2D layout to reflect topic changes while
            preserving spatial stability"

    artifact document_layout' : arrangement(documents)
        context: projection_space
        principle: "updated projection incorporating topic changes with
            smooth transitions"
        description: "Updated 2D positioning reflecting refined topic
structure"

    transform T_update_visualization :
        intent: visualise
        manner: "animated transition showing topic changes"
        input: document_layout', topic_assignments',
topic_keyword_weights'
        output: topic_view'
        actor: machine
        description: "Update visualization with smooth animations
showing
            topic membership changes"

    artifact topic_view' : visualisation(document_layout',
topic_assignments',
            topic_keyword_weights')
        layout: "2D scatterplot with animated transitions"
        form: "colored points with change indicators"
        encoding: "position from updated layout; color by new
assignments;
            split coloring shows changes"
        description: "Updated view showing refined topic structure with
            explicit change indicators"

    assign:
        topics := topics'
        topic_keyword_weights := topic_keyword_weights'
        document_topic_weights := document_topic_weights'
        topic_assignments := topic_assignments'

```

```

        document_layout := document_layout'
        topic_view := topic_view'
    else:
        exit loop L_refinement
end loop L_refinement

# =====
# PATTERN IDENTIFICATION: Characterize final topics
# =====

transform T_characterize_topics :
    intent: abstract
    manner: "identify semantic themes and relationships"
    input: topics, topic_keyword_weights, document_topic_weights, topic_view
    output: topic_patterns
    actor: human
    description: "Identify and characterize semantic themes, topic relationships,
        and document organization"

artifact topic_patterns : pattern(topics, documents)
    representation form: "categorized themes with descriptions and relationships"
    description: "Identified semantic themes with their characteristics,
        representative keywords, document distributions, and inter-topic
        relationships"

# =====
# KNOWLEDGE GENERATION: Synthesize insights
# =====

transform T_generate_knowledge :
    intent: generate-knowledge
    manner: "formulate-statements"
    input: topic_patterns, topics, topic_keyword_weights
    output: corpus_knowledge
    actor: human
    description: "Synthesize understanding of corpus organization, dominant themes,
        and semantic structure"

artifact corpus_knowledge : knowledge(topic_patterns)
    representation form: "statements and explanations"
    description: "Understanding of document corpus: major themes and their
        characteristics, document organization by topics, semantic relationships
        between themes, coverage of subject matter"

```

C.9 Progressive Abstraction Analysis of Multivariate Temporal Data

Source: Andrienko, N., Andrienko, G. and Shirato, G. (2023), Episodes and Topics in Multivariate Temporal Data. Computer Graphics Forum, 42: e14926.

doi: 10.1111/cgf.14926

Workflow summary:

This visual analytics workflow analyses multivariate temporal data partitioned into episodes to understand complex dynamic phenomena through progressive abstraction. The approach has three main analytical phases:

1. **Single-attribute pattern extraction:** Transform temporal sequences of attribute values within episodes into symbolic representations (using methods like SAX encoding) that capture variation patterns. Analysts iteratively refine encoding parameters (segment counts, discretization breaks) to achieve interpretable patterns.
2. **Multi-attribute pattern discovery:** Apply topic modeling to collections of symbolic patterns to discover "topics" - combinations of single-attribute patterns that frequently co-occur across episodes. Analysts experiment with parameters (number of topics), assess topic interpretability, and may merge semantically similar topics.
3. **Distribution pattern analysis:** Visualize and explore how multi-attribute patterns (topics) are distributed across episodes in relevant contexts (time, space, external conditions). Analysts identify higher-level distribution patterns and formulate domain insights.

A schematic representation of the workflow is shown in Fig. 10.

ATWL representation:

```
workflow Episode_Based_Temporal_Analysis
  template: define-unit →
    loop(characterise → visualise → assess → generate-knowledge) →
    loop(abstract → visualise → assess → generate-knowledge) →
    loop(contextualise → visualise → abstract → generate-knowledge →
      assess → define-unit (focused exploration))

  description: "Progressive abstraction workflow for episode-based analysis
    of multivariate temporal data through three main phases:
    single-attribute pattern extraction, multi-attribute pattern
    discovery via topic modeling, and distribution pattern analysis"

  # =====
  # PHASE 0: Episode Preparation
  # =====

  artifact raw_timeseries : entities
    origin: given
    internal structure: elementary
    embedment: {set, time}
    features:
```

```

- id: entity_id
  value structure: atomic
  value type: categorical
  description: "Entity identifier (e.g., country)"
- id: attr_values
  value structure: vector
  value type: numeric
  description: "Multiple attribute values at each time point"
description: "Multivariate time series from multiple entities; each time point
              contains vector of attribute values describing dynamic phenomena"

```

```

transform T_define_episodes :
  intent: define-unit
  manner: "temporal segmentation into episodes"
  input: raw_timeseries
  output: episodes
  actor: hybrid
  description: "Segment continuous time series into meaningful temporal intervals
              based on domain-specific criteria"

```

```

artifact episodes : entities
  internal structure: episode
  embedment: time
  features:
    - id: time_span
      value structure: atomic
      value type: temporal
      description: "Episode temporal extent"
    - id: attribute_series
      value structure: vector
      value type: numeric
      description: "Multivariate time series within episode"
  description: "Temporal intervals each containing multivariate
              time series representing distinct periods in dynamic
              process"

```

```

# =====
# PHASE 1: Single-Attribute Pattern Extraction
# =====

```

```

artifact encoding_params : specification
  origin: given
  representation form: "parameter set"
  description: "Parameters controlling symbolic encoding: number of temporal
              segments, alphabet size, value discretization breaks"

```

```

loop L_encoding_refinement:
  purpose: "Iteratively refine encoding parameters until symbolic patterns
           are interpretable and align with domain knowledge"
  until: "Symbolic patterns are interpretable and discretization breaks
         reflect domain-meaningful thresholds"
  body:

```

```

transform T_encode_patterns :
  intent: characterise
  manner: "symbolic encoding of temporal variation"
  input: episodes, encoding_params
  output: symbolic_patterns
  actor: machine
  description: "Transform each attribute's time series within
    episodes into symbolic codes capturing variation patterns"

artifact symbolic_patterns : feature(episodes)
  value structure: vector
  value type: text
  description: "Symbolic codes representing single-attribute
    temporal variation patterns for each episode; one code
    per attribute encoding discretized value levels over
    temporal segments"

transform T_visualize_patterns :
  intent: visualise
  manner: "tabular display with color coding"
  input: symbolic_patterns, episodes
  output: pattern_table_vis
  actor: machine
  description: "Display symbolic patterns for all attributes
    and episodes in a color-coded table"

artifact pattern_table_vis : visualisation(symbolic_patterns, episodes)
  layout: "table with episodes as rows, attributes as columns"
  form: "color-coded symbols"
  encoding: "table position from episodes structure; symbol colors
    represent value levels; symbols show variation patterns"
  description: "Table showing symbolic patterns for all attributes and
    episodes with color coding for efficient perception"

transform T_assess_encoding :
  intent: assess
  manner: "evaluate pattern interpretability"
  input: pattern_table_vis, symbolic_patterns
  output: encoding_assessment
  actor: human
  description: "Assess whether symbolic patterns are
    interpretable and whether discretization aligns
    with domain knowledge"

artifact encoding_assessment : knowledge(symbolic_patterns)
  representation form: "quality judgment"
  description: "Assessment of whether symbolic patterns are interpretable
    and whether discretization aligns with domain knowledge"

if encoding_assessment indicates refinement needed:
  then:
    transform T_adjust_encoding :

```

```

    intent: generate-knowledge
    manner: "decide parameter adjustments"
    input: encoding_assessment
    output: encoding_params_updated
    actor: human
    description: "Decide how to adjust encoding parameters
                 based on assessment of pattern interpretability
                 and domain alignment"

artifact encoding_params_updated : specification
    representation form: "parameter set"
    description: "Updated encoding parameters with adjusted segment counts,
                 alphabet size, or discretization breaks"

    assign:
        encoding_params := encoding_params_updated
    else:
        exit loop L_encoding_refinement

end loop L_encoding_refinement

# =====
# PHASE 2: Multi-Attribute Pattern Discovery
# =====

artifact topic_params : specification
    origin: given
    representation form: "parameter set"
    description: "Parameters for topic modeling: target number of topics,
                 convergence criteria"

loop L_topic_refinement:
    purpose: "Iteratively refine topics until they represent distinct,
             interpretable multi-attribute behaviors"
    until: "Topics are interpretable, distinct, and semantically coherent"
    body:
        transform T_discover_topics :
            intent: abstract
            manner: "topic modeling of pattern co-occurrences"
            input: symbolic_patterns, topic_params
            output: topics
            actor: machine
            description: "Apply topic modeling to discover combinations
                         of single-attribute patterns that frequently co-occur
                         across episodes"

        artifact topics : pattern(episodes)
            representation form: "weighted combinations of patterns"
            description: "Multi-attribute combination patterns
                         representing integrated temporal behaviors; each topic
                         defined by weights of constituent single-attribute
                         patterns"

```

```

transform T_compute_topic_weights :
  intent: characterise
  manner: "compute episode-topic associations"
  input: topics, episodes
  output: topic_weights
  actor: machine
  description: "Compute how strongly each topic relates to
    each episode"

artifact topic_weights : feature(episodes)
  value structure: vector
  value type: numeric
  description: "Weights indicating relevance of each topic to each episode"

transform T_visualize_topics :
  intent: visualise
  manner: "tabular display of topic compositions"
  input: topics
  output: topic_table_vis
  actor: machine
  description: "Display which single-attribute patterns
    contribute to each topic with their weights"

artifact topic_table_vis : visualisation(topics)
  layout: "table with topics as rows, attributes as columns"
  form: "weighted pattern symbols with color coding"
  encoding: "table structure from topics; symbol weights shown by size or
    intensity; colors match pattern encoding"
  description: "Table showing which single-attribute patterns contribute
    to each topic with their weights"

transform T_assess_topics :
  intent: assess
  manner: "evaluate topic quality"
  input: topic_table_vis, topics
  output: topic_assessment
  actor: human
  description: "Assess whether topics represent distinct,
    interpretable behaviors; identify semantically
    similar topics or inconsistent mixtures"

artifact topic_assessment : knowledge(topics)
  representation form: "quality judgment"
  description: "Assessment of whether topics represent distinct,
    interpretable behaviors; identification of semantically
    similar topics or inconsistent mixtures"

if topic_assessment indicates refinement needed:
then:
  transform T_decide_refinement :
    intent: generate-knowledge

```

```

manner: "formulate refinement strategy"
input: topic_assessment
output: topic_refinement_strategy
actor: human
description: "Decide on merge operations, definition edits,
             or topic count adjustment"

artifact topic_refinement_strategy : specification
representation form: "strategic decisions"
description: "Decisions on merging similar topics, editing definitions,
             or adjusting topic count parameter"

transform T_merge_topics :
intent: abstract
manner: "consolidate semantically similar topics"
input: topics, topic_weights, topic_refinement_strategy
output: topics_updated, topic_weights_updated,
        topic_params_updated
actor: hybrid
description: "Combine semantically similar topics by
             summing weights; edit individual topic definitions;
             produce updated effective parameters"

artifact topics_updated : pattern(episodes)
representation form: "weighted combinations of patterns"
description: "Consolidated multi-attribute patterns
             after merging semantically similar topics"

artifact topic_weights_updated : feature(episodes)
value structure: vector
value type: numeric
description: "Updated topic weights after refinement"

artifact topic_params_updated : specification
representation form: "parameter set"
description: "Updated topic modeling parameters reflecting
             effective number of topics and adjustments after
             merge and editing operations"

assign:
  topics := topics_updated
  topic_weights := topic_weights_updated
  topic_params := topic_params_updated
else:
  exit loop L_topic_refinement

end loop L_topic_refinement

# =====
# Topic Interpretation
# =====

```

```

transform T_interpret_topics :
  intent: generate-knowledge
  manner: "formulate semantic interpretations"
  input: topics, topic_table_vis
  output: topic_interpretations
  actor: human
  description: "Formulate human-understandable interpretations using domain
knowledge"

artifact topic_interpretations : knowledge(topics)
  representation form: "semantic descriptions"
  description: "Human-understandable interpretations of what each topic
represents in domain terms"

# =====
# PHASE 3: Distribution Pattern Analysis
# =====

transform T_characterise_context :
  intent: characterise
  manner: "extract contextual attributes from episode metadata"
  input: episodes, raw_timeseries
  output: context_info
  actor: machine
  description: "Derive contextual attributes for each episode from
underlying time series metadata: temporal positions, spatial
locations, entity identifiers, external conditions"

artifact context_info : feature(episodes)
  value structure: vector
  value type: {temporal, spatial, reference, categorical}
  description: "Contextual attributes: temporal positions, spatial
locations, involved entities, external conditions"

transform T_create_context :
  intent: define-unit
  manner: "define context structure from contextual dimensions"
  input: context_info
  output: context_structure
  actor: machine
  description: "Create reference structure from contextual
dimensions for arranging episodes"

artifact context_structure : entities
  internal structure: formation
  features:
    - id: context_dims
      value structure: relational configuration
      value type: {temporal, spatial, reference, numeric}
      description: "Contextual dimensions and their organisation"
  description: "Context structure serving as reference frame for
arranging episodes"

```

```

loop L_distribution_analysis:
  purpose: "Explore topic distributions in context space to
    identify higher-level patterns and formulate insights
    through iterative focused exploration"
  until: "Distribution patterns are sufficiently understood and
    domain insights formulated"
  body:
    transform T_arrange_in_context :
      intent: contextualise
      manner: "spatial or temporal positioning"
      input: episodes, context_info, context_structure
      output: episode_arrangement
      actor: machine
      description: "Position episodes according to relevant
        contextual dimensions"

    artifact episode_arrangement : arrangement(episodes)
      context: context_structure
      principle: "positioning based on contextual attributes"
      description: "Episodes arranged according to relevant contextual
        dimensions"

    transform T_visualize_distribution :
      intent: visualise
      manner: "context-aware spatial display"
      input: topic_weights, episode_arrangement, topics
      output: distribution_vis
      actor: machine
      description: "Display topic weight distributions across
        episodes in context space"

    artifact distribution_vis : visualisation(topic_weights, episode_arrangement)
      layout: "spatial layout from episode_arrangement"
      form: "colored marks, pie charts, or proportional symbols"
      encoding: "position from episode_arrangement; colors from topics;
        sizes or proportions from topic_weights"
      description: "Visual representation of topic distributions across
        episodes in relevant context space"

    transform T_visualize_aggregated :
      intent: visualise
      manner: "aggregated characteristic displays"
      input: symbolic_patterns, topics, topic_weights
      output: aggregated_pattern_vis
      actor: machine
      description: "Display aggregated pattern characteristics
        grouped by topic for verification"

    artifact aggregated_pattern_vis : visualisation(symbolic_patterns, topics)
      layout: "grouped by topic"
      form: "segmented bar charts or distribution plots"

```

```

encoding: "grouping by topic; segments show pattern characteristics;
        sizes show frequencies"
description: "Aggregated view of pattern characteristics grouped by
        topics for verification"

transform T_identify_distribution_patterns :
  intent: abstract
  manner: "discover distributional regularities"
  input: distribution_vis, aggregated_pattern_vis, context_info
  output: distribution_patterns
  actor: human
  description: "Identify higher-level patterns in topic
        distributions across contexts"

artifact distribution_patterns : pattern(topics, context_info)
  representation form: "described regularities"
  description: "Higher-level patterns in topic distributions: temporal
        sequences, spatial associations, relationships to
        external conditions, complementary distributions"

transform T_formulate_insights :
  intent: generate-knowledge
  manner: "synthesize understanding"
  input: distribution_patterns, topic_interpretations, context_info
  output: domain_insights
  actor: human
  description: "Synthesize domain understanding from
        distribution patterns"

artifact domain_insights : knowledge(distribution_patterns)
  representation form: "statements and explanations"
  description: "Documented high-level findings about the
        dynamic phenomenon: behavioral phases, response
        patterns to conditions, entity-specific
        characteristics, temporal evolution"

# Assessment of understanding completeness
transform T_assess_understanding :
  intent: assess
  manner: "evaluate completeness of pattern understanding"
  input: distribution_vis, distribution_patterns, domain_insights
  output: understanding_assessment
  actor: human
  description: "Assess whether distribution patterns are sufficiently
        understood or require focused exploration of specific
        episode subsets"

artifact understanding_assessment : knowledge(distribution_patterns)
  representation form: "quality judgment"
  description: "Assessment of whether distribution patterns are
        sufficiently understood; identification of areas
        requiring focused exploration"

```

```

# Conditional focused exploration
if understanding_assessment indicates focused exploration needed:
then:
  transform T_formulate_query :
    intent: generate-knowledge
    manner: "formulate selection criteria"
    input: distribution_vis, understanding_assessment
    output: user_query
    actor: human
    description: "Formulate query criteria for examining specific episode
                 subsets based on identified gaps in understanding"

  artifact user_query : specification
    representation form: "query criteria"
    description: "User-specified selection criteria for episode subsets:
                 topic dominance, context properties, pattern characteristics"
"

  transform T_filter_episodes :
    intent: define-unit
    manner: "subset selection by criteria"
    input: episodes, topic_weights, context_info, user_query
    output: filtered_episodes
    actor: machine
    description: "Select episodes satisfying query criteria
                 for detailed examination"

  artifact filtered_episodes : entities
    internal structure: episode
    embedment: time
    features:
      - id: episode_data
        value structure: vector
        value type: numeric
        description: "Episode characteristics"
    description: "Selected episodes satisfying query criteria for
                 detailed examination"

  transform T_examine_details :
    intent: visualise
    manner: "detailed episode display"
    input: filtered_episodes, symbolic_patterns, topic_weights
    output: detail_vis
    actor: machine
    description: "Display detailed patterns and topic
                 compositions for selected episodes"

  artifact detail_vis : visualisation(filtered_episodes, symbolic_patterns)
    layout: "detailed tabular or timeline layout"
    form: "detailed patterns and topic compositions"
    encoding: "episode structure from filtered_episodes; patterns from

```

```

        symbolic_patterns; topic weights from topic_weights"
    description: "Detailed view of patterns and topic compositions for
        selected episodes supporting verification and refinement"
else:
    exit loop L_distribution_analysis
end loop L_distribution_analysis

```

C.10 Partition-based Regression Modelling

Source: T. Mühlbacher and H. Piringer "A Partition-Based Framework for Building and Validating Regression Models" IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 1962-1971, Dec. 2013 doi: 10.1109/TVCG.2013.125

Workflow summary

This paper presents a partition-based framework for building and validating regression models through interactive, iterative feature subset selection. The workflow combines quantitative relevance ranking with visual exploration of conditional target distributions.

- **Data Preparation.** The dataset is split into training and validation subsets for model fitting and unbiased evaluation.
- **Initial Model Identification.** Feature domains are partitioned into disjoint regions, and goodness-of-fit relevance measures are computed for individual features and feature pairs. Ranked small-multiple visualisations of conditional target distributions allow the analyst to identify the most relevant features and characterise relationship structures (non-linearity, local patterns, interactions). The analyst then specifies an initial model (choosing features, transformations, and model type) informed by discovered relationships and domain knowledge.
- **Iterative Model Refinement.** The analytical target switches to model residuals. Relevance rankings and conditional-distribution visualisations are recomputed with respect to residuals, exposing effects not yet captured by the model: previously included features rank lower while new relevant features and pair-wise interactions emerge. The analyst discovers additional explanatory variables, transformations, and interactions, and builds a refined model. This cycle of assessment, residual-based exploration, pattern discovery, and model refinement repeats until accuracy is satisfactory.
- **Final Assessment.** Model variants are compared through visualisations of local accuracy differences, and the analyst synthesises domain insights about feature–target relationships, interaction effects, and model adequacy.

Figure 11 represents the workflow in a flow diagram.

ATWL representation

```
workflow PartitionBasedRegressionModeling
  description: "Interactive framework for regression model building through
    iterative feature subset selection, combining partition-based relevance
    ranking with visual exploration of conditional target distributions"

  template: define-unit →
    characterise → visualise → abstract → generate-knowledge (specify) →
    build-model →
    loop(assess → characterise → visualise → abstract →
    generate-knowledge (specify) → build-model) →
    assess → generate-knowledge

# =====
# GIVEN ARTIFACTS
# =====

artifact measurement_data : entities
  origin: given
  internal structure: elementary
  embedment: set
  description: "Records containing a quantitative target variable and
    candidate continuous or categorical independent variables"

# =====
# PHASE 1: DATA PREPARATION
# =====

transform T_split_data :
  intent: define-unit
  manner: "partition dataset into disjoint subsets"
  input: measurement_data
  output: training_data, validation_data
  actor: human
  description: "Separate records into training and validation subsets
    for unbiased model fitting and evaluation"

artifact training_data : entities
  internal structure: elementary
  embedment: set
  description: "Subset of records for fitting regression models"

artifact validation_data : entities
  internal structure: elementary
  embedment: set
  description: "Subset of records for evaluating model performance
    and computing feature rankings"

# =====
# PHASE 2: INITIAL MODEL IDENTIFICATION (target = predicted variable Y)
# =====
```

```

transform T_rank_features :
  intent: characterise
  manner: "partition feature domains and compute multi-resolution
          relevance measures for conditional target distributions"
  input: validation_data
  output: feature_relevance
  actor: machine
  description: "Rank features by relevance to the target variable
               using partition-based approximation of conditional distributions"

artifact feature_relevance : feature(validation_data)
  value structure: vector
  value type: numeric
  description: "Relevance measures for individual features
               and feature pairs at increasing partition complexities,
               with local distributional statistics per partition
               region"

transform T_visualize_relationships :
  intent: visualise
  manner: "display ranked small-multiple partition-based plots with
          selectable domain- or frequency-preserving layouts"
  input: validation_data, feature_relevance
  output: relationship_viz
  actor: machine
  description: "Present conditional target distributions over ranked
               features and feature pairs for visual relationship exploration"

artifact relationship_viz :
  visualisation(validation_data, feature_relevance)
  layout: "ranked small-multiple grid ordered by relevance"
  form: "1D percentile plots and 2D colour-coded cells"
  encoding: "grid position: relevance rank; 1D plots:
            x-axis from feature partition, y-axis from target
            percentiles; 2D plots: colour from conditional
            statistics; layout toggle: domain- or frequency-
            preserving partitions"
  description: "Ranked 1D percentile plots over partitioned
               features and 2D colour-coded conditional statistics
               over feature pairs, with domain- or frequency-
               preserving layout options"

transform T_identify_patterns :
  intent: abstract
  manner: "identify dominant features and characterise relationship
          shapes from visual and quantitative evidence"
  input: relationship_viz, feature_relevance
  output: relationship_patterns
  actor: human
  description: "Discover which features are most relevant and what
               relationship structures they exhibit with the target"

```

```

artifact relationship_patterns : pattern(validation_data)
  representation form: "characterised relationship structures"
  description: "Discovered feature-target relationships including
    dominant features, relationship shapes, and interaction effects
    between feature pairs"

transform T_specify_model :
  intent: generate-knowledge
  manner: "decide model structure integrating observed patterns with
    domain expertise"
  input: relationship_patterns
  output: model_spec, modeling_rationale
  actor: human
  description: "Formulate initial model configuration by selecting
    features, transformations, and model type based on discovered
    relationships and domain knowledge"

artifact model_spec : specification
  representation form: "model configuration"
  description: "Regression model type, selected features, and
    transformations such as polynomial expansions, categorisations,
    and interaction terms"

artifact modeling_rationale : knowledge(relationship_patterns)
  representation form: "domain-informed reasoning"
  description: "Analyst's rationale for feature selection and model
    structure choices, integrating domain expertise with observed
    relationships"

transform T_build_model :
  intent: build-model
  manner: "fit regression model to training data using specified
    configuration"
  input: training_data, model_spec
  output: regression_model
  actor: machine
  description: "Create initial regression model from specified
    features and transformations"

artifact regression_model : model(training_data, model_spec)
  description: "Fitted regression model predicting the target
    variable from selected features and transformations"

# =====
# PHASE 3: ITERATIVE MODEL REFINEMENT (target = residuals)
# =====

loop L_refinement:
  purpose: "Improve the regression model by incorporating features,
    transformations, and interactions that explain residual variance"
  until: "prediction accuracy is satisfactory and no further relevant

```

```

features are discoverable in residual distributions"
body:

transform T_assess_model :
  intent: assess
  manner: "compute prediction accuracy on validation data and
         track error metrics across model variants"
  input: regression_model, validation_data
  output: model_assessment
  actor: hybrid
  description: "Evaluate whether the current model captures
              sufficient variance and whether refinement is warranted"

artifact model_assessment : knowledge(regression_model)
  representation form: "quality judgment with error metrics"
  description: "Assessment of prediction accuracy on
              validation data with comparison across model variants"

if model_assessment indicates refinement needed:
  then:
    transform T_rank_on_residuals :
      intent: characterise
      manner: "compute residuals as analytical target
             and re-rank features and feature pairs by
             residual-based relevance"
      input: regression_model, validation_data
      output: updated_feature_relevance
      actor: machine
      description: "Re-rank features by relevance to
                  model residuals, exposing effects not yet
                  captured"

    artifact updated_feature_relevance :
      feature(validation_data)
      value structure: vector
      value type: numeric
      description: "Residual-based relevance measures;
                  features already captured by the model rank
                  lower while new relevant features and
                  interactions emerge"

    assign: feature_relevance := updated_feature_relevance

    transform T_visualize_residuals :
      intent: visualise
      manner: "display ranked partition-based plots of
             conditional residual distributions over
             features and feature pairs"
      input: validation_data, feature_relevance
      output: updated_relationship_viz
      actor: machine
      description: "Present conditional residual

```

```

distributions for visual exploration of
unexplained effects"

artifact updated_relationship_viz :
  visualisation(validation_data, feature_relevance)
  layout: "ranked small-multiple grid ordered by residual
  relevance"
  form: "1D percentile plots and 2D colour-coded cells"
  encoding: "grid position: residual relevance rank; 1D
  plots: x-axis from feature partition, y-axis from
  residual percentiles; 2D plots: colour from
  conditional residual statistics"
  description: "Ranked partition-based plots showing
  conditional residual distributions, revealing local
  prediction bias and unexplained effects"

assign: relationship_viz := updated_relationship_viz

transform T_discover_refinement :
  intent: abstract
  manner: "identify residual-based patterns from
  visual and quantitative evidence"
  input: relationship_viz, feature_relevance
  output: updated_relationship_patterns
  actor: human
  description: "Discover additional features,
  interactions, and local bias structures in
  residual distributions"

artifact updated_relationship_patterns :
  pattern(validation_data)
  representation form: "characterised residual
  structures"
  description: "Discovered residual-based patterns
  including additional relevant features,
  interaction effects, and local bias structures"

assign:
  relationship_patterns := updated_relationship_patterns

transform T_decide_refinement :
  intent: generate-knowledge
  manner: "formulate refinement strategy integrating
  residual analysis with domain expertise"
  input: relationship_patterns, model_assessment
  output: updated_model_spec,
  updated_modeling_rationale
  actor: human
  description: "Decide which additional features,
  transformations, and interactions to
  incorporate based on residual patterns and
  domain knowledge"

```

```

    artifact updated_model_spec : specification
      representation form: "model configuration"
      description: "Refined model specification
        incorporating additional features,
        transformations, and interactions"

    artifact updated_modeling_rationale :
      knowledge(relationship_patterns)
      representation form: "domain-informed reasoning"
      description: "Analyst's rationale for refinement
        choices, integrating residual analysis with
        domain expertise"

    assign: model_spec := updated_model_spec
    assign: modeling_rationale := updated_modeling_rationale

    transform T_build_refined :
      intent: build-model
      manner: "fit regression model to training data
        using updated configuration"
      input: training_data, model_spec
      output: updated_regression_model
      actor: machine
      description: "Create refined regression model with
        expanded feature set and transformations"

    artifact updated_regression_model :
      model(training_data, model_spec)
      description: "Refined regression model with expanded
        feature set and transformations"

    assign: regression_model := updated_regression_model

  else:
    exit loop L_refinement

end loop L_refinement

# =====
# PHASE 4: FINAL ASSESSMENT AND KNOWLEDGE SYNTHESIS
# =====

transform T_compare_models :
  intent: assess
  manner: "visualise point-wise differences of residual magnitudes
    between model variants across feature subspaces"
  input: regression_model, validation_data
  output: final_assessment
  actor: hybrid
  description: "Compare model variants from the refinement history
    by examining local differences in prediction accuracy"

```

```

artifact final_assessment : knowledge(regression_model)
  representation form: "comparative quality judgment"
  description: "Comparative assessment of model variants identifying
    regions of relative superiority and overall prediction adequacy"

transform T_synthesize :
  intent: generate-knowledge
  manner: "synthesise insights from model characteristics, assessment
    results, and accumulated modeling rationale"
  input: regression_model, final_assessment, modeling_rationale
  output: domain_knowledge
  actor: human
  description: "Formulate domain understanding of feature-target
    relationships, interaction effects, and model adequacy"

artifact domain_knowledge : knowledge(regression_model)
  representation form: "comprehensive domain insights"
  description: "Understanding of feature-target relationships,
    interaction effects, and model capabilities for prediction
    and domain decisions"

```

C.11 Spatio-temporal analysis and modelling

Source:

N. Andrienko and G. Andrienko

"A visual analytics framework for spatio-temporal analysis and modelling"

Data Mining and Knowledge Discovery, vol.27, no. 1, 2013, pp. 55-83,

doi: 10.1007/s10618-012-0285-7

Workflow summary

The workflow supports analysis and modelling of spatio-temporal data in the form of spatially referenced numeric time series. Raw data are first transformed into spatial time series through spatio-temporal aggregation. The time series are then clustered by temporal similarity, and the resulting grouping is iteratively refined through visual assessment of within-group homogeneity on time graphs and spatial pattern interpretability on maps.

For each group of similar time series, the analyst visually identifies temporal variation characteristics—such as periodicity, cycle lengths, and trends—from the time graph. A representative time series is derived from the group, and a suitable modelling method is selected and configured. A statistical model is fitted and iteratively refined by adjusting parameters while comparing the model curve against the data until it captures the group's characteristic temporal variation.

Model quality is evaluated by examining distributions of residuals over time and space: randomly distributed residuals indicate that the model captures the essential spatio-temporal variation. If systematic residual patterns emerge, the analyst decides whether to subdivide clusters, adjust the modelling approach, or both, and repeats the analysis

cycle. Once all models pass the residual check, their descriptions—including method, parameters, group membership, and distribution statistics—are stored externally for reuse, communication, and prediction.

The same workflow structure applies to modelling dependencies between two spatio-temporal variables, using dependency series and regression models in place of temporal models.

A flow diagram of the workflow is shown in Fig. 12.

ATWL representation:

```

workflow ST_analysis_modelling

template: define-unit (prepare) →
  loop(define-unit (group) → visualise (grouping) → assess (grouping) →
    abstract (temporal) → characterise (representative) →
    generate-knowledge (configure) → build-model →
    visualise (model) → assess (fit) →
    characterise (residuals) → visualise (residuals) → assess (residuals) →
    generate-knowledge (refinement)) →
  generate-knowledge (externalize)

description: "Visual analytics framework for analysis and modelling of
  spatio-temporal data: combines interactive clustering with statistical
  time series modelling to build externally storable, reusable
  spatio-temporal models"

# =====
# INPUT
# =====

artifact D_raw : entities
  origin: given
  internal structure: elementary
  embedment: {set, time, space}
  features:
    - id: measurements
      value structure: vector
      value type: numeric
      description: "Attribute values (e.g., call duration) with
        associated spatial coordinates and time stamps"
  description: "Raw spatio-temporal data records to be aggregated into
  spatial time series"

# =====
# STEP 0: DATA PREPARATION
# =====

```

```

transform T_prepare :
  intent: define-unit
  manner: "spatio-temporal aggregation"
  input: D_raw
  output: D_locations, F_TS
  actor: hybrid
  description: "Transform raw spatio-temporal data into spatial time
    series by dividing territory into spatial compartments and
    aggregating attribute values by location and time interval"

artifact D_locations : entities
  internal structure: elementary
  embedment: space
  description: "Spatial locations or objects, each associated with a
    numeric time series"

artifact F_TS : feature(D_locations)
  value structure: vector
  value type: numeric
  description: "Numeric time series of attribute values, one per
    spatial location"

# =====
# SPECIFICATIONS AND INITIAL LOOP ARTIFACTS
# =====

artifact S_clustering : specification
  origin: given
  representation form: "parameter settings"
  description: "Initial clustering parameters: number of clusters and
    distance measure"

artifact K_refinement_strategy : knowledge
  origin: given
  representation form: "refinement strategy"
  description: "Refinement guidance from prior residual analysis;
    initially neutral (no prior guidance available)"

# =====
# OUTER LOOP: ANALYSIS CYCLE (Steps 1-3)
# =====

loop L_analysis:
  purpose: "Iteratively group time series, build statistical models,
    and evaluate model quality until residual distributions indicate
    satisfactory spatio-temporal models"
  until: "Model residuals are randomly distributed over time and space
    for all groups"
  body:

    # =====

```

```

# STEP 1: GROUPING (inner loop for cluster refinement)
# =====

loop L_grouping:
  purpose: "Iteratively cluster and refine grouping until groups
    are homogeneous and form meaningful spatial patterns"
  until: "Groups exhibit sufficient within-group homogeneity and
    interpretable spatial patterns"
  body:

    transform T_cluster :
      intent: define-unit
      manner: "clustering by temporal similarity"
      input: D_locations, F_TS, S_clustering
      output: D_groups, F_group_label
      actor: hybrid
      description: "Group time series by similarity of temporal
        variation; includes progressive clustering of
        heterogeneous groups"

    artifact D_groups : entities
      internal structure: group/cluster
      embedment: set
      description: "Groups of spatial locations with similar
        temporal variation"

    artifact F_group_label : feature(D_locations)
      value structure: atomic
      value type: categorical
      description: "Cluster membership identifier for each
        location"

    transform T_vis_timegraph :
      intent: visualise
      input: F_TS, F_group_label
      output: V_timegraph
      actor: machine
      description: "Display time series colored by cluster
        membership in detailed or summarized form"

    artifact V_timegraph : visualisation(F_TS, F_group_label)
      layout: "temporal axis"
      form: "colored lines and summary bands"
      encoding: "color from cluster membership; vertical position
        from value; quintile stripes for summarized view"
      description: "Time graph showing temporal variation of time
        series grouped by cluster"

    transform T_vis_map :
      intent: visualise
      input: D_locations, F_group_label
      output: V_map

```

```

    actor: machine
    description: "Display spatial distribution of clusters on a
        map with similarity-based coloring"

artifact V_map : visualisation(D_locations, F_group_label)
    layout: "geographic map"
    form: "colored areas"
    encoding: "color from cluster membership; similar colors
        reflect similar cluster centres"
    description: "Map showing spatial distribution of clusters"

transform T_assess_grouping :
    intent: assess
    input: V_timegraph, V_map, D_groups
    output: K_grouping_quality
    actor: human
    description: "Assess within-group homogeneity on the time
        graph and spatial pattern interpretability on the map"

artifact K_grouping_quality : knowledge(D_groups)
    representation form: "quality judgment"
    description: "Assessment of grouping quality: homogeneity,
        spatial interpretability, and whether refinement is
        needed"

if K_grouping_quality indicates satisfactory grouping:
    then:
        exit loop L_grouping
    else:
        transform T_adjust_clustering :
            intent: generate-knowledge
            manner: "adjust clustering approach"
            input: K_grouping_quality, V_timegraph, V_map,
                S_clustering
            output: S_clustering_g
            actor: human
            description: "Adjust clustering parameters: modify
                number of clusters, apply progressive clustering
                to heterogeneous groups, or change distance
                measure"

            artifact S_clustering_g : specification
                representation form: "parameter settings"
                description: "Updated clustering parameters for
                    grouping refinement"

            assign:
                S_clustering := S_clustering_g

        end loop L_grouping

# =====

```

```

# STEP 2: ANALYSIS AND MODELLING
# =====

# Step 2a: Identify temporal characteristics

transform T_identify_characteristics :
  intent: abstract
  input: V_timegraph, D_groups
  output: P_temporal_chars
  actor: human
  description: "Identify characteristics of temporal variation by
    visual inspection: periodicity, cycle lengths, and trends"

artifact P_temporal_chars : pattern(D_groups)
  representation form: "identified temporal characteristics"
  description: "Characteristics of temporal variation within each
    group: periodicity, cycle lengths, trends"

# Step 2b: Derive representative time series

transform T_derive_representative :
  intent: characterise
  manner: "aggregate within-group statistics"
  input: D_groups, F_TS
  output: F_representative
  actor: hybrid
  description: "Derive representative time series for each group
    from mean, median, or percentile values at each time step"

artifact F_representative : feature(D_groups)
  value structure: vector
  value type: numeric
  description: "Representative time series capturing typical
    temporal variation for each group, derived from mean,
    median, or percentile values"

# Step 2c: Configure and build model

transform T_configure_model :
  intent: generate-knowledge
  input: P_temporal_chars, K_refinement_strategy
  output: S_model_config
  actor: human
  description: "For each group, select a suitable modelling method
    and configure initial parameters based on identified
    temporal characteristics and, when available, refinement
    guidance from prior residual analysis"

artifact S_model_config : specification
  representation form: "model configuration"
  description: "Model type selection, smoothing coefficients,
    cycle length, trend parameters, and initialisation values"

```

```

transform T_build_model :
  intent: build-model
  input: F_representative, S_model_config
  output: M_temporal
  actor: machine
  description: "For each group, fit the selected statistical model
    to the representative time series with automatic parameter
    optimization; may involve per-cycle models when multiple
    temporal cycles are present"

artifact M_temporal : model(F_representative, P_temporal_chars)
  model type: "statistical time series model"
  representation form: "fitted parametric model"
  description: "Statistical model of characteristic temporal
    variation for each group, fitted to representative time
    series; may be a combination of per-cycle models when
    multiple temporal cycles are present"

# Step 2d: Model refinement (inner loop)

loop L_modelling:
  purpose: "Iteratively adjust model parameters until the model
    curve captures the characteristic temporal variation"
  until: "Model curve adequately represents the general temporal
    variation pattern"
  body:

    transform T_predict :
      intent: characterise
      input: M_temporal, D_groups
      output: F_model_curve
      actor: machine
      description: "Generate model-predicted values for original
        and future time steps"

    artifact F_model_curve : feature(D_groups)
      value structure: vector
      value type: numeric
      description: "Model-predicted values representing
        characteristic temporal variation for each group"

    transform T_vis_model :
      intent: visualise
      input: F_model_curve, F_representative, F_TS
      output: V_model
      actor: machine
      description: "Display model curve on time graph alongside
        representative and individual time series"

    artifact V_model : visualisation(F_model_curve,
      F_representative, F_TS)

```

```

layout: "temporal axis"
form: "lines"
encoding: "color distinguishing model curve from
representative and individual series; extended range
for future predictions"
description: "Time graph showing model prediction overlaid
on data for visual comparison"

transform T_assess_fit :
  intent: assess
  input: V_model
  output: K_model_fit
  actor: human
  description: "Assess whether model curve captures the
general temporal variation pattern"

artifact K_model_fit : knowledge(M_temporal)
  representation form: "quality judgment"
  description: "Assessment of model fit and whether parameter
refinement is needed"

if K_model_fit indicates satisfactory fit:
  then:
    exit loop L_modelling
  else:
    transform T_adjust_model_config :
      intent: generate-knowledge
      input: K_model_fit, V_model, S_model_config
      output: S_model_config_m
      actor: human
      description: "Adjust model parameters or method
based on fit assessment and visual inspection
of the model curve"

    artifact S_model_config_m : specification
      representation form: "model configuration"
      description: "Updated model configuration with
adjusted parameters"

    transform T_refit_model :
      intent: build-model
      manner: "re-fitting with adjusted parameters"
      input: F_representative, S_model_config_m
      output: M_temporal_r
      actor: machine
      description: "Re-fit model with updated
configuration"

    artifact M_temporal_r : model(F_representative,
P_temporal_chars)
      model type: "statistical time series model"
      representation form: "fitted parametric model"

```

```

        description: "Refined temporal model with adjusted
            parameters"

    assign:
        S_model_config := S_model_config_m
        M_temporal := M_temporal_r

end loop L_modelling

# =====
# STEP 3: MODEL EVALUATION
# =====

transform T_compute_residuals :
    intent: characterise
    input: D_locations, F_TS, M_temporal, D_groups
    output: F_residuals
    actor: machine
    description: "Compute per-location residuals as differences
        between actual values and individually adjusted model
        predictions"

artifact F_residuals : feature(D_locations)
    value structure: vector
    value type: numeric
    description: "Model residual time series for each location,
        computed as differences between actual values and
        individually adjusted model predictions"

transform T_vis_residuals_temporal :
    intent: visualise
    input: F_residuals, D_groups
    output: V_residuals_temporal
    actor: machine
    description: "Display temporal distribution of residuals as
        quintile summaries and segmented temporal histograms"

artifact V_residuals_temporal : visualisation(F_residuals, D_groups)
    layout: "temporal axis"
    form: "summary bands and histogram bars"
    encoding: "quintile stripes for distribution shape; color-coded
        segments for residual value intervals"
    description: "Time graph summarizing residual distributions
        over time"

transform T_vis_residuals_spatial :
    intent: visualise
    input: F_residuals, D_locations
    output: V_residuals_spatial
    actor: machine
    description: "Display spatial distribution of residuals on a
        map"

```

```

artifact V_residuals_spatial : visualisation(F_residuals,
  D_locations)
  layout: "geographic map"
  form: "colored areas"
  encoding: "color for residual magnitude"
  description: "Map showing spatial distribution of model
  residuals"

transform T_assess_residuals :
  intent: assess
  input: V_residuals_temporal, V_residuals_spatial, D_groups
  output: K_residual_quality
  actor: human
  description: "Assess whether residuals are randomly distributed
  over time and space; may involve clustering residual time
  series to isolate subgroups with systematic non-random
  patterns"

artifact K_residual_quality : knowledge(M_temporal, D_groups)
  representation form: "quality judgment"
  description: "Assessment of residual distributions: presence or
  absence of systematic temporal or spatial patterns"

if K_residual_quality indicates random residual distribution:
  then:
    exit loop L_analysis
  else:
    transform T_decide_refinement :
      intent: generate-knowledge
      input: K_residual_quality, V_residuals_temporal,
        V_residuals_spatial, S_clustering
      output: S_clustering_a, K_refinement_strategy_a
      actor: human
      description: "Determine refinement approach from residual
      patterns: subdivide groups by updating clustering
      parameters, adjust the modelling approach, or both"

    artifact S_clustering_a : specification
      representation form: "parameter settings"
      description: "Potentially updated clustering parameters;
      unchanged if only modelling adjustments are needed"

    artifact K_refinement_strategy_a : knowledge(M_temporal,
      D_groups)
      representation form: "refinement strategy"
      description: "Decision on whether to subdivide groups,
      adjust modelling method or parameters, or both"

  assign:
    S_clustering := S_clustering_a
    K_refinement_strategy := K_refinement_strategy_a

```

```

end loop L_analysis

# =====

# STEP 4: MODEL EXTERNALIZATION

# =====

transform T_externalize :
  intent: generate-knowledge
  input: M_temporal, D_groups
  output: K_externalized
  actor: machine
  description: "Store model descriptions externally in reusable format
    including modelling method, parameters, group membership, and
    distribution statistics for each location"

artifact K_externalized : knowledge(M_temporal, D_groups)
  representation form: "formal descriptions and data files"
  description: "Externally stored model descriptions enabling
    re-creation, review, communication, prediction, and further
    analysis"

```

C.12 Feature engineering for behaviour pattern recognition

Source: N. Andrienko, G. Andrienko, A. Artikis, P. Mantenoglou and S. Rinzivillo, "Human-in-the-Loop: Visual Analytics for Building Models Recognizing Behavioral Patterns in Time Series," in IEEE Computer Graphics and Applications, vol. 44, no. 3, pp. 14-29, May-June 2024, doi: 10.1109/MCG.2024.3379851

Concise Workflow Summary

The human-in-the-loop workflow for recognizing behavioural patterns in vessel trajectories combines feature engineering with interactive visual analytics to build classification models that are both flexible and tolerant to data noise. The workflow partitions continuous vessel trajectories into overlapping episodes of appropriate duration (3-hour windows with 1-hour shifts), then derives interval-based synoptic features capturing behavioural aspects such as speed levels, movement curvature, and spatial context (e.g., distance to ports) that distinguish trawling activities from other movements.

Through iterative visual exploration, domain experts refine the feature space by evaluating whether episodes clustered by feature similarity exhibit coherent behavioural patterns, adjusting or adding features (such as applying logarithmic transformations to handle skewed distributions) until groups are interpretable and well-separated. A dimensionality reduction projection (UMAP) enables simultaneous visual inspection of feature similarity across all episodes, while coordinated views showing trajectory shapes on geographic maps and feature distributions in histograms allow experts to efficiently examine and label

cluster cores as representative examples of different pattern types.

This iterative refinement process—cycling between feature engineering, clustering, visualization, pattern interpretation, and quality assessment—continues until labelled examples enable reliable automated classification, as validated through k-nearest neighbour testing. The approach not only successfully identified expected trawling patterns but also revealed three distinct trawling subtypes (wide curves, tight loops, and straight movements with 180-degree turns) that would have remained hidden without the flexibility and discovery capabilities of the human-in-the-loop methodology.

A flow diagram representing the workflow is shown in Fig. 13.

ATWL representation:

```
workflow behaviour-pattern-recognition-in-time-series
  template: define-unit → characterise →
    loop(define-unit (cluster) → contextualise → visualise →
      abstract (interpret) → assess → characterise (refine features)) →
    define-unit (label) → build-model →
    loop(characterise (predict) → visualise → assess →
      build-model (refine)) →
    generate-knowledge
  description: "Human-in-the-loop visual analytics workflow for building
    classification models that recognize behavioral patterns in movement
    trajectories through iterative feature engineering and example selection"

artifact D0 : entities
  origin: given
  internal structure: sequence
  embedment: {set, time}
  features:
    - id: movement_data
      value structure: vector
      value type: {numeric, spatial}
      description: "Geographic positions, speeds, headings"
  description: "Multivariate movement trajectories of multiple
    vessels over extended time period"

transform T1 :
  intent: define-unit
  manner: "time-partitioning with overlapping windows"
  input: D0
  output: D1
  actor: hybrid
  description: "Partition continuous trajectories into overlapping episodes of
    appropriate duration to capture behavioral patterns"

artifact D1 : entities
  internal structure: episode
  embedment: {set, time}
  features:
    - id: entity_id
```

```

    value structure: atomic
    value type: reference
    description: "Entity identifier"
  - id: temporal_extent
    value structure: atomic
    value type: temporal
    description: "Episode time span"
description: "Time-bounded trajectory segments potentially
  containing behavioral patterns of interest"

```

transform T2 :

```

intent: characterise
manner: "compute interval-based synoptic features"
input: D1
output: F1
actor: machine
description: "Abstract elementary positional data into summary statistics
  representing behavioral aspects such as speed level, trajectory curvature,
  and spatial context"

```

artifact F1 : feature(D1)

```

value structure: vector
value type: numeric
description: "Initial set of synoptic features capturing speed
  characteristics, movement curvature, and spatial
  relationships"

```

loop L1:

```

purpose: "Iteratively refine feature space to align with interpretable
  behavioral
  patterns"
until: "Features adequately distinguish target patterns from other behaviors;
  episode groups show clear behavioral coherence"
body:

```

transform T3 :

```

  intent: define-unit
  manner: "cluster-by-similarity"
  input: D1, F1
  output: D2, F2
  actor: machine
  description: "Group episodes with similar feature profiles to reveal
    natural behavioral clusters"

```

artifact D2 : entities

```

  internal structure: group/cluster
  embedment: set
  description: "Groups of episodes exhibiting similar behavioral
    characteristics"

```

artifact F2 : feature(D1)

```

  value structure: atomic
  value type: categorical

```

```

    description: "Cluster membership indicators"

transform T4 :
    intent: contextualise
    manner: "projection by dimensionality reduction"
    input: D1, F1
    output: D_proj_space, A1
    actor: machine
    description: "Create low-dimensional projection space from feature
similarity
        and arrange episodes within it to expose behavioral groupings"

artifact D_proj_space : entities
    internal structure: elementary
    features:
        - id: dimensions
          value structure: atomic
          value type: numeric
          description: "Number of spatial dimensions (2)"
    description: "Computed 2D projection space where proximity
        reflects feature similarity"

artifact A1 : arrangement(D1)
    context: D_proj_space
    principle: "similarity-preserving projection from feature space"
    description: "Spatial arrangement where proximity indicates behavioral
        similarity"

transform T5 :
    intent: visualise
    manner: "coordinated multiple views"
    input: A1, F2, D1
    output: V1
    actor: machine
    description: "Display episodes in projection space with cluster coloring
,
        linked to geographic trajectory views and feature distribution
displays"

artifact V1 : visualisation(A1, F2, D1)
    layout: "coordinated views: projection plot, geographic map, feature
        distributions"
    form: "colored points, trajectory lines, distribution charts"
    encoding: "projection position from arrangement; color from cluster
        membership; trajectory shape and feature values from episode data"
    description: "Interactive visualization enabling assessment of cluster
        quality and behavioral coherence"

transform T6 :
    intent: abstract
    manner: "interpret behavioral patterns from clusters"
    input: V1, D2, F1

```

```

    output: P2
    actor: human
    description: "Interpret clusters as behavioral patterns; assign
        preliminary semantic labels to discovered groupings"

    artifact P2 : pattern(D1, D2)
        representation form: "tentative behavioral labels with descriptions"
        description: "Conceptual interpretations of discovered behavioral
patterns
        with preliminary semantic labels"

    transform T7 :
        intent: assess
        manner: "evaluate feature adequacy and cluster coherence"
        input: V1, D2, F1, P2
        output: feature_assessment
        actor: human
        description: "Assess whether episodes within clusters exhibit coherent
        behavioral patterns and evaluate cluster separation; determine if
        feature refinement needed"

    artifact feature_assessment : knowledge(D2, F1)
        representation form: "quality judgment"
        description: "Expert judgment of feature adequacy: whether current
features
        achieve meaningful behavioral discrimination and cluster coherence,
        and whether refinement is needed"

    if feature_assessment indicates refinement needed:
        then:
            transform T8 :
                intent: characterise
                manner: "feature space engineering"
                input: D1, F1, feature_assessment
                output: F1'
                actor: hybrid
                description: "Refine features based on assessment: add new
features,
                apply transformations, adjust feature selection to better
                capture behavioral distinctions"

            artifact F1' : feature(D1)
                value structure: vector
                value type: numeric
                description: "Updated synoptic features with refined
                or expanded feature set"

            assign:
                F1 := F1'
        else:
            exit loop L1
    end loop L1

```

```

transform T9 :
  intent: define-unit
  manner: "interactive selection and labeling of representative exemplars"
  input: D1, D2, A1, V1, P2
  output: D3, F3
  actor: human
  description: "Select cluster cores or high-quality representatives through
visual
  inspection; label episodes as instances of identified behavioral patterns"

artifact D3 : entities
  internal structure: group/cluster
  embedment: set
  features:
    - id: cluster_size
      value structure: atomic
      value type: numeric
      description: "Number of labeled episodes per pattern type"
  description: "Labeled training set of behavioral pattern
  exemplars representing diverse pattern types discovered
  in the data"

artifact F3 : feature(D3)
  value structure: atomic
  value type: categorical
  description: "Expert-assigned behavioral pattern labels for
  supervised learning"

transform T10 :
  intent: build-model
  manner: "supervised classification model training"
  input: D3, F1, F3
  output: M1
  actor: hybrid
  description: "Train machine learning classifier to map synoptic features to
  behavioral pattern labels using labeled exemplars"

artifact M1 : model(F1, F3)
  model type: "classifier"
  representation form: "supervised learning model"
  description: "Computational model for automated recognition of behavioral
  patterns
  from trajectory episodes"

loop L2:
  purpose: "Validate and refine model to ensure reliable pattern recognition"
  until: "Model errors are understood and minimized; classification accuracy is
  acceptable for intended application"
  body:
    transform T11 :
      intent: characterise

```

```

    manner: "apply model for prediction"
    input: D1, F1, M1
    output: F4
    actor: machine
    description: "Apply trained model to classify episodes based on their
        feature profiles"

artifact F4 : feature(D1)
    value structure: atomic
    value type: categorical
    description: "Model-predicted behavioral pattern labels"

transform T12 :
    intent: visualise
    manner: "comparative error analysis display"
    input: D1, D3, F3, F4, A1
    output: V2
    actor: machine
    description: "Visualize classification results showing agreement/
disagreement
        between predictions and human labels; highlight uncertain
classifications"

artifact V2 : visualisation(D1, F3, F4, A1)
    layout: "coordinated views with classification results"
    form: "color-coded trajectories and confidence indicators"
    encoding: "position from arrangement; color from predicted class;
opacity
        or highlighting indicates classification confidence or error"
    description: "Visual display of model performance showing accuracy,
        systematic errors, and uncertain classifications"

transform T13 :
    intent: assess
    manner: "evaluate model performance and diagnose errors"
    input: V2, D1, M1, F3, F4
    output: model_assessment
    actor: human
    description: "Analyze classification errors to determine causes:
ambiguous
        features, borderline cases, insufficient training coverage, labeling
inconsistencies; decide if refinement needed"

artifact model_assessment : knowledge(M1, D1)
    representation form: "quality judgment with diagnostic insights"
    description: "Understanding of model performance: identification of
error
        types, error patterns, and potential causes; assessment of whether
model refinement is needed"

if model_assessment indicates refinement needed:
    then:

```

```

        transform T14 :
            intent: build-model
            manner: "model or training set adjustment"
            input: D3, F1, F3, model_assessment, M1
            output: M1'
            actor: hybrid
            description: "Refine model based on error analysis: adjust
parameters,
                    modify training examples, or revise feature selection"

        artifact M1' : model(F1, F3)
            model type: "classifier"
            representation form: "refined supervised learning model"
            description: "Updated behavioral pattern recognition model with
                    improved performance"

            assign:
                M1 := M1'
            else:
                exit loop L2
    end loop L2

transform T15 :
    intent: generate-knowledge
    manner: "formulate-statements"
    input: M1, P2, model_assessment, feature_assessment
    output: K1
    actor: human
    description: "Synthesize insights from pattern discovery and model development
        process"

artifact K1 : knowledge(M1, P2)
    representation form: "statements and insights"
    description: "Discovered behavioral pattern subtypes and their characteristics;
        understanding of pattern manifestations in data; insights into data-driven
        pattern recognition compared to specification-based approaches"

```

C.13 Exploratory Model Analysis

Source:

Cashman, D., Humayoun, S.R., Heimerl, F., Park, K., Das, S., Thompson, J., Saket, B., Mosca, A., Stasko, J., Endert, A., Gleicher, M. and Chang, R. (2019), A User-based Visual Analytics Workflow for Exploratory Model Analysis. Computer Graphics Forum, 38: 185-199. doi:10.1111/cgf.13681

Workflow summary

This paper presents a visual analytics workflow for Exploratory Model Analysis (EMA) — discovering and selecting predictive models on a dataset when neither the modeling

task nor the model type is predetermined.

Data Exploration. The analyst examines dataset attributes through interactive, cross-linked visualizations appropriate to the data types present (histograms for tabular data, node-link diagrams for graphs, line charts for time-series), identifying feature distributions, inter-attribute relationships, and potentially predictive variables. From this exploration, the analyst forms initial modelling interests — which variables appear predictive and what types of models might be suitable.

Problem Exploration. The system automatically enumerates valid modelling problems by pairing each variable as a potential prediction target with compatible model types (classification, regression, clustering, forecasting, etc.) and evaluation metrics. The analyst browses these candidate problems in context of their data understanding.

Iterative Model Discovery. The analyst specifies a modelling problem by selecting a target variable, model type, evaluation metric, and predictor features — choosing from system-generated candidates or defining a custom specification, informed by their analytical direction from data exploration and diagnostic reasoning from any prior modelling attempts. An automated system trains a diverse set of candidate models, which are visualized through type-appropriate prediction displays (confusion matrices for classification, residual bar charts for regression) cross-linked with data exploration views. The analyst identifies comparative performance patterns across models and assesses their suitability for deployment. If models are unsatisfactory — exhibiting poor performance or systematic bias in certain subgroups — the analyst diagnoses deficiencies and returns to specify a different modelling problem. This cycle continues until suitable models are found.

Model Selection. When satisfactory models have been identified, the analyst selects preferred models based on prediction quality, error distribution, and deployment requirements, and exports them for use on unseen data.

Figure 14 shows the workflow represented as a flow diagram.

ATWL representation

```
workflow ExploratoryModelAnalysis
  description: "Visual analytics workflow for exploratory model analysis:
    discovering and selecting predictive models through iterative
    problem specification, automated model training, and interactive
    model comparison on holdout data"

  template: characterise → visualise → abstract →
    characterise → visualise →
    loop(generate-knowledge (specify) → build-model →
      visualise → abstract → assess →
      generate-knowledge (diagnose)) →
    generate-knowledge (select)

  # =====
  # GIVEN ARTIFACTS
  # =====
```

```

artifact dataset : entities
  origin: given
  internal structure: elementary
  embedment: set
  description: "Multi-type dataset with tabular, graph, temporal,
    text, image, or other data types for model discovery"

# =====
# PHASE 1: DATA EXPLORATION (Step 1)
# =====

transform T_compute_data_stats :
  intent: characterise
  manner: "calculate per-attribute distributional statistics and
    data type properties"
  input: dataset
  output: data_statistics
  actor: machine
  description: "Compute summary statistics and distributions for
    all dataset attributes"

artifact data_statistics : feature(dataset)
  value structure: vector
  description: "Summary statistics and distributional
    properties for all dataset attributes"

transform T_visualize_data :
  intent: visualise
  manner: "render cross-linked type-appropriate visualizations:
    histograms for tabular data, node-link diagrams for graphs,
    line charts for time-series, sortable collections for text
    and media"
  input: dataset, data_statistics
  output: data_viz
  actor: machine
  description: "Display interactive data overview for exploration
    of attribute distributions and relationships"

artifact data_viz : visualisation(dataset, data_statistics)
  layout: "cross-linked multi-panel dashboard"
  form: "histograms, node-link diagrams, line charts,
    sortable collections"
  encoding: "view type matched to data type; cross-linking
    by shared instances for coordinated exploration"
  description: "Interactive visualizations of dataset
    attributes with cross-linking between views for
    coordinated exploration"

transform T_explore_data :
  intent: abstract
  manner: "explore feature distributions and inter-attribute
    relationships to assess predictive potential and form

```

```

        analytical direction"
input: data_viz, data_statistics
output: data_characteristics, analytical_direction
actor: human
description: "Identify data characteristics, potentially
    predictive relationships, and initial modelling interests"

artifact data_characteristics : pattern(dataset)
    representation form: "identified data properties"
    description: "Notable data characteristics including feature
        distributions, inter-attribute relationships, and
        potentially predictive variables"

artifact analytical_direction : knowledge(dataset)
    representation form: "recorded modelling interests and goals"
    description: "Analyst's initial assessment of which variables
        and model types appear promising based on data exploration"

# =====
# PHASE 2: PROBLEM EXPLORATION (Step 2)
# =====

transform T_enumerate_problems :
    intent: characterise
    manner: "enumerate each variable as potential prediction target;
        determine compatible model types and evaluation metrics per
        target variable type"
    input: dataset
    output: modeling_possibilities
    actor: machine
    description: "Generate the set of valid modeling problems based
        on dataset structure and variable types"

artifact modeling_possibilities : feature(dataset)
    value structure: vector
    description: "Valid modeling problems pairing target
        variables with compatible model types and evaluation
        metrics"

transform T_present_problems :
    intent: visualise
    manner: "show list of auto-generated problem specifications
        organized by target variable, model type, and metric"
    input: modeling_possibilities
    output: problems_viz
    actor: machine
    description: "Display candidate modeling problems for user
        exploration and selection"

artifact problems_viz : visualisation(modeling_possibilities)
    layout: "organized list grouped by target variable"
    form: "interactive list entries"

```

```

encoding: "grouping by target variable; entries show
model type and evaluation metric"
description: "Interactive list of candidate modeling
problems with target variables, model types, and
metrics"

# =====
# PHASE 3: ITERATIVE MODEL DISCOVERY (Steps 3-7)
# =====

loop L_model_discovery:
  purpose: "Discover suitable predictive models by iteratively
specifying modeling problems, training candidate models,
and evaluating their predictions"
  until: "satisfactory models have been found and selected for
export"
  body:

    transform T_specify_problem :
      intent: generate-knowledge
      manner: "choose target variable, model type, evaluation
metric, and predictor features informed by
analytical direction and data characteristics"
      input: modeling_possibilities, data_characteristics,
problems_viz, analytical_direction
      output: problem_spec
      actor: human
      description: "Select and refine a modeling problem
specification defining the prediction task"

    artifact problem_spec : specification
      representation form: "modeling problem configuration"
      description: "Target variable, model type, evaluation
metric, and selected predictor features defining
the modeling task"

    transform T_train_models :
      intent: build-model
      manner: "automated search across compatible algorithms
and hyperparameter configurations"
      input: dataset, problem_spec
      output: candidate_models
      actor: machine
      description: "Train diverse candidate models for the
specified modeling problem"

    artifact candidate_models :
      model(dataset, problem_spec)
      description: "Diverse set of trained candidate models
ranked by the specified evaluation metric"

    transform T_visualize_models :

```

```

intent: visualise
manner: "render ranked, problem-type-appropriate
prediction visualizations with cross-linking
between models and to data exploration views"
input: candidate_models, dataset, data_viz
output: model_viz
actor: machine
description: "Display model predictions on holdout data
for exploration and comparison"

artifact model_viz :
    visualisation(candidate_models, dataset)
layout: "ranked model comparison grid"
form: "confusion matrices, residual charts, or other
type-appropriate prediction displays"
encoding: "grid position by evaluation rank;
prediction displays matched to problem type;
cross-linking to data exploration views"
description: "Ranked model prediction visualizations on
holdout data with cross-linking to data views for
instance-level exploration"

transform T_identify_performance :
    intent: abstract
manner: "compare prediction patterns, error
distributions, and per-class or per-subset
performance across models"
input: model_viz, candidate_models
output: performance_patterns
actor: human
description: "Identify performance characteristics and
differences across candidate models"

artifact performance_patterns : pattern(candidate_models)
representation form: "identified performance
characteristics"
description: "Comparative performance patterns across
candidate models including error distributions,
per-class accuracy, and relative strengths"

transform T_assess_models :
    intent: assess
manner: "judge model quality, robustness, and
suitability based on performance patterns and
task requirements"
input: performance_patterns, model_viz
output: model_assessment
actor: human
description: "Evaluate whether candidate models are
suitable for the analyst's deployment needs"

artifact model_assessment : knowledge(candidate_models)

```

```

    representation form: "quality judgment"
    description: "Assessment of whether candidate models
        meet task requirements, identifying deficiencies
        or confirming suitability"

if model_assessment indicates further problem exploration needed:
    then:
        transform T_revise_approach :
            intent: generate-knowledge
            manner: "diagnose why current models are
                unsatisfactory and decide which
                alternative problem to try"
            input: model_assessment, data_characteristics
            output: updated_analytical_direction
            actor: human
            description: "Determine what alternative
                modeling problem to explore based on
                observed deficiencies"

        artifact updated_analytical_direction :
            knowledge(candidate_models)
            representation form: "diagnostic reasoning"
            description: "Understanding of why current
                models are inadequate and rationale for
                exploring an alternative modeling problem"

        assign:
            analytical_direction := updated_analytical_direction
    else:
        exit loop L_model_discovery

end loop L_model_discovery

# =====
# PHASE 4: MODEL SELECTION (Steps 6-7)
# =====

transform T_select_models :
    intent: generate-knowledge
    manner: "choose one or more models based on prediction
        quality, error distribution, and deployment requirements"
    input: model_assessment, performance_patterns, model_viz
    output: selection_decision
    actor: human
    description: "Select preferred models for export based on
        comparative evaluation"

artifact selection_decision : knowledge(candidate_models)
    representation form: "model preference ranking"
    description: "Selected models with preference ranking and
        rationale for deployment on unseen data"

```

C.14 Diagnosing binary classifiers

Source:

J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs and E. Bertini,
"A Workflow for Visual Diagnostics of Binary Classifiers using Instance-Level Explanations,"
2017 IEEE Conference on Visual Analytics Science and Technology (VAST), Phoenix, AZ, USA, 2017, pp. 162-172,
doi: 10.1109/VAST.2017.8585720.

Workflow summary

This paper presents a visual analytics workflow for diagnosing binary classifiers using instance-level explanations. The workflow enables data scientists and domain experts to understand model decisions, identify weaknesses, and generate actionable improvement hypotheses through a structured multi-level exploration.

Data Preparation and Model Training. The analyst prepares a feature set based on domain requirements, splits data into training and test subsets, and trains a binary classifier.

Explanation Computation and Grouping. Prediction scores are computed for all test instances. Instance-level explanations are then generated for each data item by determining the minimal set of features whose removal changes the predicted label, treating the classifier as a black box. Instances sharing identical explanation feature sets are grouped into subsets representing distinct model decisions, and per-group statistics (prediction distributions, odds ratios with confidence intervals, discriminative feature rankings) are computed.

Initial Overview. Three linked diagnostic views are rendered: prediction score histograms, confusion matrix, and ROC curve at the outcome level; a sortable explanation list with prediction distributions and statistical significance at the feature level; and an item-feature matrix with discriminative ordering at the instance level. The analyst reads the statistical summary to assess overall accuracy, error distribution, and generalization.

Multi-Level Exploration. The analyst iteratively explores across the three levels — re-sorting explanations by frequency, significance, or uncertainty; filtering by prediction score range or feature search; and drilling down to instance-level feature matrices for specific explanation groups. Each cycle reveals decision patterns, significant or ambiguous predictors, misclassification sources, and feature associations that explain prediction errors. Exploration continues until sufficient patterns and root causes have been identified.

Diagnostic Synthesis and Improvement. The analyst synthesizes diagnostic insights about model behavior, data limitations, and improvement opportunities. If the insights indicate that data engineering changes, additional features, or model adjustments could improve predictions, the analyst specifies those improvements and the cycle repeats with updated data and a retrained model. When the model is adequate or fundamental limitations are understood, the analyst formulates a final assessment of the classifier's practical utility.

A flow diagram representing this workflow is shown in Fig. 15.

ATWL representation

```
workflow BinaryClassifierDiagnostics
  description: "Visual analytics workflow for diagnosing binary
    classifiers using instance-level explanations: computing
    model-agnostic explanations, grouping instances by shared
    decision logic, and iteratively exploring decisions at
    outcome, explanation, and instance levels to produce
    diagnostic insights for model and data improvement"

  # Outer loop: model improvement cycle
  # Inner exploration loop retained in body only
  template: loop(define-unit → build-model →
    characterise → characterise →
    define-unit → characterise →
    visualise → abstract →
    visualise (explore) → abstract → assess →
    generate-knowledge (diagnose) →
    generate-knowledge (specify) →
    generate-knowledge (finalize)

  # =====
  # GIVEN ARTIFACTS
  # =====

  artifact raw_data : entities
    origin: given
    internal structure: elementary
    embedment: set
    description: "Records with binary feature vectors and ground
      truth labels for binary classification"

  artifact data_model_spec : specification
    origin: given
    representation form: "feature selection and model
      configuration"
    description: "Initial specification of feature selection
      and model type based on domain requirements"

  # =====
  # OUTER LOOP: MODEL IMPROVEMENT CYCLE
  # =====

  loop L_model_improvement:
    purpose: "Iteratively build, diagnose, and improve binary
      classifiers to understand model behavior and identify
      data and model improvement opportunities"
    until: "model behavior is understood and either adequate for
      deployment or fundamental limitations are identified"
    body:
```

```

# -----
# Phase 1: Data Preparation and Model Training
# -----

transform T_prepare_data :
  intent: define-unit
  manner: "select and engineer features per
          specification; split into training and test
          subsets"
  input: raw_data, data_model_spec
  output: training_data, test_data
  actor: machine
  description: "Prepare training and test datasets
               according to current data and feature
               specifications"

artifact training_data : entities
  internal structure: elementary
  embedment: set
  description: "Subset of records for model training"

artifact test_data : entities
  internal structure: elementary
  embedment: set
  description: "Subset of records for model evaluation
               and explanation computation"

transform T_train_model :
  intent: build-model
  manner: "evaluate candidate model types and fit
          classifier using best-performing type and
          parameters"
  input: training_data, data_model_spec
  output: trained_classifier
  actor: machine
  description: "Train binary classifier on the
               training data"

artifact trained_classifier : model(training_data, data_model_spec)
  description: "Trained binary classifier predicting
               class labels from feature vectors"

# -----
# Phase 2: Prediction and Explanation Computation
# -----

transform T_compute_predictions :
  intent: characterise
  manner: "apply classifier to compute prediction
          scores and derive labels using optimal threshold"
  input: test_data, trained_classifier

```

```

    output: prediction_scores
    actor: machine
    description: "Compute prediction scores for all test
        instances"

artifact prediction_scores : feature(test_data)
    value structure: vector
    description: "Classifier prediction scores and
        threshold-derived labels for all test instances"

transform T_compute_explanations :
    intent: characterise
    manner: "for each instance, determine minimal set of
        features whose removal changes the predicted
        label, treating the model as a black box"
    input: test_data, trained_classifier
    output: instance_explanations
    actor: machine
    description: "Compute instance-level explanations
        describing the classifier's decision logic for
        each data item"

artifact instance_explanations : feature(test_data)
    value structure: vector
    description: "For each instance, the minimal set of
        features whose removal changes the classifier's
        prediction, describing the local decision logic"

# -----
# Phase 3: Explanation Grouping and Statistics
# -----

transform T_group_explanations :
    intent: define-unit
    manner: "aggregate instances by identical explanation
        feature sets"
    input: test_data, instance_explanations
    output: explanation_groups
    actor: machine
    description: "Group instances sharing the same
        explanation into subsets representing distinct
        model decisions"

artifact explanation_groups : entities
    internal structure: group/cluster
    embedment: set
    description: "Subsets of instances grouped by shared
        explanation, each representing a distinct
        decision pattern of the classifier"

transform T_compute_group_stats :
    intent: characterise

```

```

manner: "calculate prediction score distributions,
  label distributions, odds ratios with confidence
  intervals, and relative feature importance per
  group"
input: explanation_groups, prediction_scores
output: group_statistics
actor: machine
description: "Compute prediction accuracy and
  statistical significance for each explanation
  group"

artifact group_statistics : feature(explanation_groups)
  value structure: vector
  description: "Prediction distributions, odds ratios
  with confidence intervals, and discriminative
  feature rankings per explanation group"

# -----
# Phase 4: Initial Visualization and Overview
# -----

transform T_visualize_diagnostic :
  intent: visualise
  manner: "display prediction score histograms,
  confusion matrix, and ROC curve at outcome
  level; sortable explanation list with prediction
  distributions, group sizes, and odds ratios at
  feature level; item-feature matrix with
  discriminative feature ordering at instance
  level"
  input: test_data, prediction_scores,
  explanation_groups, group_statistics,
  instance_explanations
  output: diagnostic_viz
  actor: machine
  description: "Render linked multi-level diagnostic
  views for exploring classifier decisions"

artifact diagnostic_viz : visualisation(test_data,
  explanation_groups, group_statistics)
  layout: "three linked vertical panels: outcome level,
  feature level, instance level"
  form: "histograms and confusion matrix; sortable list
  with embedded distribution bars; item-feature
  binary matrix"
  encoding: "outcome panel: score histograms with
  class coloring, confusion matrix cell counts,
  ROC curve; feature panel: rows sorted by
  frequency, significance, or uncertainty, with
  embedded prediction distribution bars and odds
  ratio indicators; instance panel: binary matrix
  with discriminative feature ordering"

```

```

description: "Outcome-level statistical summary,
feature-level explanation explorer with sortable
group statistics, and instance-level feature
matrix with discriminative ordering"

transform T_read_overview :
  intent: abstract
  manner: "recognise prediction score distribution
characteristics, confusion matrix balance, ROC
performance, and training-test generalization
patterns"
  input: diagnostic_viz, group_statistics
  output: diagnostic_patterns
  actor: human
  description: "Identify overall accuracy
characteristics and error distribution from
the statistical summary"

artifact diagnostic_patterns :
  pattern(trained_classifier)
  representation form: "identified model behavior
characteristics"
  description: "Overall accuracy patterns, error
clustering, and generalization characteristics
from the statistical summary"

# -----
# Phase 5: Multi-Level Diagnostic Exploration
# -----

loop L_exploration:
  purpose: "Explore model decisions through outcome,
explanation, and instance levels to identify
decision patterns, accuracy issues, and root
causes"
  until: "sufficient decision patterns, accuracy
issues, and potential root causes have been
identified for diagnostic synthesis"
  body:

    transform T_explore_views :
      intent: visualise
      manner: "re-sort explanation list by
frequency, odds ratio, or uncertainty;
filter by prediction score range or
feature search; drill down to
instance-level feature matrix for
specific explanation groups"
      input: diagnostic_viz, explanation_groups,
group_statistics, instance_explanations
      output: updated_diagnostic_viz
      actor: hybrid

```

```

description: "Navigate between diagnostic
  levels by re-sorting, filtering, or
  drilling down to explore specific
  decisions"

artifact updated_diagnostic_viz :
  visualisation(test_data, explanation_groups,
    group_statistics)
  layout: "three linked vertical panels: outcome level,
    feature level, instance level"
  form: "histograms and confusion matrix; sortable list
    with embedded distribution bars; item-feature
    binary matrix"
  encoding: "same encoding as diagnostic_viz with
    updated sorting, filtering, or drill-down state"
  description: "Diagnostic visualization reflecting
    current sorting, filtering, or drill-down state"

assign:
  diagnostic_viz := updated_diagnostic_viz

transform T_identify_patterns :
  intent: abstract
  manner: "recognise significant predictors,
    uncertain or ambiguous explanations,
    misclassification patterns, and feature
    associations that explain prediction
    errors"
  input: diagnostic_viz, group_statistics
  output: updated_diagnostic_patterns
  actor: human
  description: "Identify decision patterns,
    accuracy issues, and potential root
    causes from the current view"

artifact updated_diagnostic_patterns :
  pattern(trained_classifier)
  representation form: "identified model
    behavior characteristics"
  description: "Accumulated decision patterns
    including significant predictors,
    uncertain explanations,
    misclassification sources, and root
    cause hypotheses"

assign: diagnostic_patterns :=
  updated_diagnostic_patterns

transform T_assess_exploration :
  intent: assess
  manner: "judge completeness of model
    behavior understanding across decision

```

```

        types and accuracy levels"
    input: diagnostic_patterns, diagnostic_viz
    output: exploration_assessment
    actor: human
    description: "Evaluate whether sufficient
        patterns and root causes have been
        identified for diagnostic synthesis"

    artifact exploration_assessment :
        knowledge(diagnostic_patterns)
        representation form: "quality judgment"
        description: "Assessment of whether model
            behavior is sufficiently understood
            for diagnostic synthesis"

    end loop L_exploration

# -----
# Phase 6: Diagnostic Synthesis and Improvement Decision
# -----

transform T_synthesize :
    intent: generate-knowledge
    manner: "synthesise decision patterns, accuracy
        issues, and root causes into actionable
        diagnostic understanding"
    input: diagnostic_patterns, exploration_assessment
    output: diagnostic_insights
    actor: human
    description: "Formulate diagnostic insights about
        model behavior, data limitations, and
        improvement opportunities"

    artifact diagnostic_insights :
        knowledge(trained_classifier)
        representation form: "diagnostic statements"
        description: "Documented findings about model
            decision patterns, data limitations, and
            whether model or data changes could improve
            prediction quality"

    if diagnostic_insights indicate improvement warranted:
        then:
            transform T_specify_improvements :
                intent: generate-knowledge
                manner: "determine which features to add or
                    remove, what data to capture, and what
                    model changes to make based on
                    diagnostic findings"
                input: diagnostic_insights,
                    diagnostic_patterns
                output: updated_data_model_spec

```

```

        actor: human
        description: "Specify data engineering and
                    model changes to address identified
                    limitations"

    artifact updated_data_model_spec : specification
        representation form: "data and model
                               improvement plan"
        description: "Specification of feature
                     engineering changes, additional data
                     sources, and model adjustments to
                     address identified limitations"

    assign:
        data_model_spec := updated_data_model_spec
    else:
        exit loop L_model_improvement

    end loop L_model_improvement

# =====
# PHASE 7: FINAL KNOWLEDGE SYNTHESIS
# =====

transform T_finalize :
    intent: generate-knowledge
    manner: "integrate diagnostic findings across all modeling
            rounds into comprehensive model assessment"
    input: diagnostic_insights, diagnostic_patterns,
           exploration_assessment
    output: model_understanding
    actor: human
    description: "Synthesize final understanding of model
                 capabilities, limitations, and practical utility"

    artifact model_understanding : knowledge(trained_classifier)
        representation form: "statements and assessment report"
        description: "Documented assessment of the classifier's
                     decision logic, prediction strengths, data
                     limitations, and practical deployment
                     considerations"

```

C.15 Interactive Exploration of Trained Ensemble Classifier

Source:

Eirich, J., Münch, M., Jäckle, D., Sedlmair, M., Bonart, J. and Schreck, T. (2022), RfX: A Design Study for the Interactive Exploration of a Random Forest to Enhance Testing Procedures for Electrical Engines. Computer Graphics Forum, 41: 302-315. doi: 10.1111/cgf.14452

Workflow summary:

This visual analytics workflow enables domain experts without machine learning expertise to extract interpretable decision rules from complex trained classifiers. The approach addresses a common challenge: making black-box ensemble models (like Random Forests) accessible to practitioners who need actionable insights rather than raw model predictions. The workflow consists of three main phases:

1. **Phase 1:** Automated Model Structuring - The system automatically analyzes the ensemble's internal structure by computing component similarities, grouping similar decision components, and creating spatial overviews. This preprocessing transforms an opaque ensemble into an organized landscape ready for exploration.
2. **Phase 2:** Progressive Component Selection - Through multiple coordinated visualizations, users progressively narrow from the entire model down to individual promising components. This involves iterative cycles of visualization, quality assessment, and selection decisions at multiple granularity levels (groups → representatives → individuals).
3. **Phase 3:** Rule Derivation - Once a suitable component is identified, users explore its detailed structure through interactive visualization, assess interpretability, optionally refine decision boundaries using domain knowledge, and extract human-readable rules for operational deployment.

The workflow emphasizes human-in-the-loop refinement, where domain expertise guides the exploration process and validates extracted rules, successfully bridging the gap between complex ML models and practical industrial applications.

Figure 16 provides a simplified flow diagram of the workflow.

ATWL representation:

```
workflow ModelExplorationForRuleExtraction
  template:
    characterise (model structure) →
    define-unit (organize components) →
    contextualise (spatial overview) →
    loop(visualise → assess →
      generate-knowledge (select component)) →
    loop(visualise → assess →
      generate-knowledge (refine)) →
    abstract (extract rules) →
    generate-knowledge (formulate deployable rules)

  description: "Visual analytics workflow enabling domain experts
  without ML expertise to interactively explore trained
  ensemble classifiers, identify interpretable decision
  patterns, and extract actionable rules for operational
  deployment"
```

```

# =====
# Given Inputs
# =====

artifact source_data : entities
  origin: given
  internal structure: elementary
  embedment: set
  features:
    - id: measurements
      value structure: vector
      value type: numeric
      description: "Feature measurements from domain sensors"
    - id: labels
      value structure: atomic
      value type: categorical
      description: "Classification outcomes"
  description: "Domain data with feature measurements and
    classifications used for model training and evaluation"

artifact trained_model : model(source_data)
  origin: given
  description: "Pre-trained ensemble classifier containing
    multiple independent decision components; optimized for
    prediction accuracy but lacks inherent interpretability"

# =====
# Phase 1: Automated Model Structure Analysis
# =====

transform T_extract_components :
  intent: define-unit
  manner: "extract individual decision components from ensemble"
  input: trained_model
  output: model_components
  actor: machine
  description: "Decompose ensemble into individual interpretable
    components for separate analysis"

artifact model_components : entities
  internal structure: elementary
  embedment: set
  features:
    - id: component_id
      value structure: atomic
      value type: categorical
      description: "Component identifier"
    - id: decision_logic
      value structure: relational configuration
      description: "Component's decision tree structure"
  description: "Individual decision-making components extracted

```

```

    from ensemble; each represents independent classification
    logic"

transform T_characterize_structure :
  intent: characterise
  manner: "compute component similarity and performance metrics"
  input: model_components, source_data
  output: component_similarities, component_metrics
  actor: machine
  description: "Analyze relationships between components through
    similarity measures combining prediction agreement and
    structural properties"

artifact component_similarities : feature(model_components)
  value structure: matrix
  value type: numeric
  description: "Pairwise similarity scores between components
    based on prediction behavior and structural
    characteristics"

artifact component_metrics : feature(model_components)
  value structure: vector
  description: "Performance characteristics: classification
    accuracy, structural complexity, error patterns"

transform T_organize_components :
  intent: define-unit
  manner: "group similar components into clusters"
  input: model_components, component_similarities
  output: component_groups
  actor: machine
  description: "Organize components into coherent groups based
    on similarity; reduces exploration space from hundreds
    to manageable clusters"

artifact component_groups : entities
  internal structure: group/cluster
  embedment: set
  features:
    - id: group_id
      value structure: atomic
      value type: categorical
      description: "Group identifier"
    - id: members
      value structure: list
      value type: reference
      description: "Component identifiers belonging to this
        group"
  description: "Clusters of similar decision components; each
    group represents components with related prediction
    behavior and structure"

```

```

transform T_characterize_groups :
  intent: characterise
  manner: "compute aggregate group properties"
  input: component_groups, component_metrics
  output: group_properties
  actor: machine
  description: "Compute aggregate performance and structural
    statistics for each component group"

artifact group_properties : feature(component_groups)
  value structure: vector
  description: "Group-level characteristics: aggregate accuracy,
    size, statistical properties of member components"

transform T_create_spatial_overview :
  intent: contextualise
  manner: "project components to spatial layout"
  input: model_components, component_similarities,
    component_groups
  output: spatial_reference, component_arrangement
  actor: machine
  description: "Create spatial arrangement where proximity
    reflects similarity; projection establishes 2D coordinate
    space and positions components to enable visual overview
    of model structure"

artifact spatial_reference : entities
  internal structure: elementary
  features:
    - id: axes
      value structure: atomic
      value type: numeric
      description: "Number of spatial dimensions (2)"
  description: "Two-dimensional coordinate space created by
    dimensionality reduction projection for component
    visualization"

artifact component_arrangement : arrangement(model_components)
  context: spatial_reference
  principle: "dimensionality reduction preserving similarity
    relationships"
  description: "Spatial layout of components where proximity
    indicates similarity; provides entry point for
    exploration"

transform T_identify_representatives :
  intent: define-unit
  manner: "select exemplar components for each group"
  input: component_groups, component_arrangement,
    group_properties
  output: representative_components
  actor: machine

```

```

description: "Identify representative component per group
for efficient browsing"

artifact representative_components : entities
  internal structure: elementary
  embedment: set
  features:
    - id: component_id
      value structure: atomic
      value type: categorical
      description: "Component identifier"
    - id: group_id
      value structure: atomic
      value type: reference
      description: "Parent group identifier"
  description: "Subset of components representing each group;
selected for typicality within group and quality metrics"

```

```

# =====
# Phase 2: Progressive Interactive Selection
# =====

```

```

loop L_progressive_selection:
  purpose: "Iteratively explore model structure at multiple
granularity levels, assess component quality, and
progressively narrow down to promising individual
components through guided navigation"
  until: "Suitable component identified for detailed
examination and rule extraction"
  body:
    transform T_visualize_overview :
      intent: visualise
      manner: "multi-view display of model structure"
      input: component_arrangement, component_groups,
            group_properties, representative_components,
            component_metrics
      output: model_overview_vis
      actor: machine
      description: "Present multiple coordinated views: spatial
overview showing component relationships, tabular
summaries enabling efficient comparison, structural
previews of representatives"

    artifact model_overview_vis :
      visualisation(component_arrangement,
                    component_groups,
                    representative_components)
      layout: "coordinated multiple views with spatial, tabular,
and hierarchical displays"
      form: "spatial scatterplot, sortable tables, structural
glyphs"
      encoding: "position from component_arrangement; colors by

```

```

        group membership; sizes by performance; tables sorted
        by quality metrics; glyphs show decision structure
        compactly"
description: "Multi-view visualization supporting both
exploratory browsing (via spatial overview) and
goal-directed search (via sortable tables)"

transform T_assess_components :
  intent: assess
  manner: "evaluate component suitability and promise"
  input: model_overview_vis, component_metrics,
        group_properties
  output: component_assessment
  actor: human
  description: "Assess which groups and individual components
are promising based on performance metrics, structural
characteristics, and domain relevance indicators"

artifact component_assessment :
  knowledge(model_components, component_groups)
  representation form: "quality judgment with selection
rationale"
  description: "Assessment identifying promising components
based on accuracy, interpretability potential, error
characteristics, and domain interest"

if component_assessment indicates suitable component found:
then:
  transform T_select_component :
    intent: generate-knowledge
    manner: "decide which component to examine in detail"
    input: component_assessment, component_metrics
    output: selected_component
    actor: human
    description: "Select the most promising component for
detailed examination based on assessment of
performance, complexity, and interpretability"

  artifact selected_component : specification
    representation form: "component identifier"
    description: "Selected component for detailed
examination; chosen for balance of good performance,
low complexity, and interpretability"

  exit loop L_progressive_selection
else:
  # Continue exploring at different granularity levels

end loop L_progressive_selection

# =====
# Phase 3: Detailed Exploration and Rule

```

```

#           Extraction
# =====

transform T_extract_component_detail :
  intent: define-unit
  manner: "retrieve full component structure"
  input: model_components, selected_component
  output: examined_component
  actor: machine
  description: "Retrieve full structural detail for the
               selected component"

artifact examined_component : entities
  internal structure: formation
  features:
    - id: decision_nodes
      value structure: relational configuration
      description: "Decision tree structure with features and
                   thresholds at each node"
    - id: node_statistics
      value structure: vector
      value type: {numeric, categorical}
      description: "Classification statistics at each decision
                   point"
  description: "Selected component with complete structural
               detail including decision logic, features, thresholds,
               and prediction statistics"

transform T_compute_distributions :
  intent: characterise
  manner: "compute feature distributions for decision points"
  input: examined_component, source_data
  output: feature_distributions
  actor: machine
  description: "Compute data distributions for features used
               in the component's decision logic"

artifact feature_distributions : feature(examined_component)
  value structure: vector
  description: "Data distributions for features used in
               component; enables evaluation of decision boundary
               quality"

loop L_component_refinement:
  purpose: "Explore component structure in detail, assess
           interpretability, optionally refine decision boundaries
           using domain knowledge, until component produces
           domain-sensible decision logic"
  until: "Component structure is interpretable and decision
         logic is satisfactory"
  body:
    transform T_visualize_detail :

```

```

intent: visualise
manner: "detailed hierarchical display with decision paths"
input: examined_component, component_metrics,
       feature_distributions
output: component_detail_vis
actor: machine
description: "Display component as detailed hierarchical
             diagram showing decision paths, feature names,
             thresholds, classification statistics, and error
             distributions"

artifact component_detail_vis :
  visualisation(examined_component)
  layout: "hierarchical tree structure with statistics at
          nodes"
  form: "node-link diagram with confusion matrices and
        feature distributions"
  encoding: "hierarchical layout from component structure;
            nodes show classification statistics; edges labeled
            with features and thresholds; colors indicate
            prediction patterns"
  description: "Detailed visualization enabling exploration
              of decision logic, understanding of feature
              interactions, and assessment of decision boundary
              quality"

transform T_assess_interpretability :
  intent: assess
  manner: "evaluate domain sensibility and interpretability"
  input: component_detail_vis, examined_component,
        feature_distributions
  output: interpretability_assessment
  actor: human
  description: "Assess whether component structure makes
              domain sense: do feature combinations reflect known
              relationships? Are decision boundaries appropriate?
              Are error patterns acceptable?"

artifact interpretability_assessment :
  knowledge(examined_component)
  representation form: "quality judgment with
                       interpretability evaluation"
  description: "Assessment of component interpretability:
              domain sensibility of feature logic, appropriateness
              of decision boundaries, acceptability of error
              patterns, identification of refinement needs"

if interpretability_assessment indicates refinement needed:
then:
  transform T_refine_boundaries :
    intent: generate-knowledge
    manner: "adjust decision boundaries based on domain

```

```

        knowledge"
    input: interpretability_assessment,
           component_detail_vis, feature_distributions
    output: boundary_adjustments
    actor: human
    description: "Specify boundary adjustments to achieve
                 better class separation, reduce critical errors,
                 or align with domain-meaningful thresholds"

artifact boundary_adjustments : specification
    representation form: "modified decision boundary
                          parameters"
    description: "User-specified adjustments to decision
                 boundaries; optimizes logic based on domain
                 expertise to reduce errors, align with physical
                 constraints, or improve interpretability"

transform T_update_component :
    intent: define-unit
    manner: "apply boundary modifications"
    input: examined_component, boundary_adjustments
    output: examined_component_updated
    actor: machine
    description: "Update component structure with modified
                 boundaries; recalculate statistics"

artifact examined_component_updated : entities
    internal structure: formation
    features:
        - id: decision_nodes
          value structure: relational configuration
          description: "Updated decision tree structure"
        - id: node_statistics
          value structure: vector
          value type: {numeric, categorical}
          description: "Recalculated statistics"
    description: "Component with user-refined decision
                 boundaries reflecting domain expertise"

assign:
    examined_component := examined_component_updated

else:
    exit loop L_component_refinement

end loop L_component_refinement

# =====
# Rule Extraction and Formulation
# =====

transform T_extract_decision_rules :

```

```

intent: abstract
manner: "extract decision paths as structured rules"
input: examined_component, component_detail_vis
output: decision_rules
actor: human
description: "Extract decision paths from component structure
as rule patterns: identify paths from root to outcomes,
capturing feature-threshold conditions defining each
classification"

artifact decision_rules : pattern(examined_component)
representation form: "conditional rule structures"
description: "Structured decision rules extracted from
component: IF-THEN conditions specifying
feature-threshold sequences leading to classifications"

transform T_formulate_deployable_rules :
intent: generate-knowledge
manner: "synthesize operational rules with domain context"
input: decision_rules, examined_component,
interpretability_assessment
output: deployable_rules
actor: human
description: "Translate extracted rules into domain-specific
operational guidance; add contextual explanation,
rationale, implementation details, and validation
criteria"

artifact deployable_rules : knowledge(decision_rules)
representation form: "operational rules with domain
interpretation"
description: "Human-readable rules for operational deployment:
measurement sequences, threshold values, decision logic,
domain interpretation, implementation guidance, and
validation procedures; ready for testing on new
operational data"

```

C.16 Exploring Deep Learning Models in TensorFlow

Source: Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mané, Doug Fritz, Dilip Krishnan, Fernanda B. Viégas, and Martin Wattenberg "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow," IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 1, pp. 1-12, Jan. 2018, doi: 10.1109/TVCG.2017.2744878.

Workflow summary

This paper presents the TensorFlow Graph Visualizer, a tool for understanding deep learning model architectures through interactive exploration of their dataflow graphs.

The analytical workflow proceeds through progressive graph simplification, automated pattern detection, and iterative interactive exploration.

Graph Simplification. A raw dataflow graph containing thousands of heterogeneous operations is progressively simplified through two stages: first, non-critical operations (constants, summaries) are extracted and encoded as embedded icons while remaining operations are grouped into a hierarchical clustered graph using namespace annotations, with edges bundled along the hierarchy. Second, high-degree non-critical nodes (variable declarations, bookkeeping hubs) are extracted using degree-based outlier detection, preserving connectivity information.

Pattern Detection and Feature Computation. Repeated modular structures are automatically detected across group nodes using heuristic subgraph similarity checking, and quantitative properties (tensor sizes, device allocation, compute time, memory usage) are computed for visual encoding.

Layout and Visualization. A hierarchical flow layout is computed recursively per cluster subgraph, and an interactive overview is rendered showing top-level model components with colour-coded repeated modules, an auxiliary node panel, and tensor-size edge encoding.

Initial Reading. From the overview, the analyst immediately recognises high-level architectural features — top-level component organisation, colour-coded repeated module groups, and main-versus-auxiliary structure — forming an initial understanding before detailed exploration.

Interactive Exploration. The analyst iteratively explores the model by expanding clusters to reveal nested structure, inspecting individual node details, switching between quantitative overlays, and recognising further architectural patterns including layer organisation, modular composition, and data flow paths. This cycle continues until the analyst has sufficient understanding of the model architecture.

Knowledge Synthesis. The analyst formulates a coherent understanding of the model's structure, modular composition, and operational organisation for development, debugging, or communication.

A diagrammatic representation of the process of model exploration in TensorFlow is provided in Fig.17.

ATWL representation:

```
workflow TensorFlowGraphVisualization
  template: define-unit (cluster) → define-unit (simplify) →
           define-unit (detect repeated) → characterise →
           contextualise → visualise → abstract →
           loop(visualise → abstract → assess) →
           generate-knowledge
  description: "Interactive exploration of deep learning model architecture
              through progressive dataflow graph simplification and hierarchical
              visualization"

# =====
```

```

# GIVEN ARTIFACTS
# =====

artifact dataflow_graph : entities
  origin: given
  internal structure: formation
  description: "Directed dataflow graph of a deep learning model
    containing operations connected by data dependency, reference,
    and control dependency edges, with hierarchical namespace
    annotations"

# =====
# PHASE 1: GRAPH SIMPLIFICATION
# =====

transform T_simplify_and_cluster :
  intent: define-unit
  manner: "extract degree-one operations (constants, summaries) as
    embedded icons; group operations by hierarchical namespace;
    bundle edges along the hierarchy"
  input: dataflow_graph
  output: clustered_graph
  actor: machine
  description: "Reduce graph complexity by extracting non-critical
    operations and grouping remaining operations into a
    hierarchical clustered graph"

artifact clustered_graph : entities
  internal structure: formation
  description: "Hierarchically clustered dataflow graph with
    non-critical operations extracted as embedded icons on
    neighbouring nodes and edges bundled along the hierarchy"

transform T_extract_auxiliary :
  intent: define-unit
  manner: "identify auxiliary nodes by degree-based outlier detection
    and domain-specific subgraph patterns; separate from core
    computational structure while preserving connectivity
    information"
  input: clustered_graph
  output: simplified_graph
  actor: machine
  description: "Separate high-degree non-critical nodes from the
    core computational structure to declutter the clustered graph"

artifact simplified_graph : entities
  internal structure: formation
  description: "Decluttered hierarchical graph with auxiliary nodes
    extracted from the main layout, retaining proxy connections"

# =====
# PHASE 2: REPEATED STRUCTURE DETECTION

```

```

# =====

transform T_detect_repeated :
  intent: define-unit
  manner: "partition group nodes by structural equivalence"
  input: simplified_graph
  output: structural_templates
  actor: machine
  description: "Identify sets of structurally identical group
    nodes by metadata-based candidate selection and
    heuristic subgraph similarity comparison"

artifact structural_templates : entities
  internal structure: group/cluster
  embedment: set
  features:
    - id: template_size
      value structure: atomic
      value type: numeric
      description: "Number of group nodes sharing the
        structure"
    description: "Sets of group nodes with identical nested
      subgraph structure, representing repeated module
      types in the model"

# =====
# PHASE 3: FEATURE COMPUTATION
# =====

transform T_compute_features :
  intent: characterise
  manner: "derive tensor sizes from edge dimensions; aggregate
    device allocation, compute time, and memory usage per
    operation and group"
  input: simplified_graph
  output: graph_features
  actor: machine
  description: "Compute quantitative properties of operations and
    edges for visual encoding"

artifact graph_features : feature(simplified_graph)
  value structure: vector
  value type: numeric
  description: "Tensor sizes, device allocation, compute
    time, and memory usage for operations and groups"

# =====
# PHASE 4: LAYOUT AND INITIAL VISUALIZATION
# =====

transform T_layout :
  intent: contextualise

```

```

    manner: "recursively compute flow layout per cluster subgraph
            with edges routed along the hierarchy"
    input: simplified_graph
    output: layout_space, graph_arrangement
    actor: machine
    description: "Position graph elements in a hierarchical flow
                layout conveying dataflow direction"

artifact layout_space : entities
    internal structure: elementary
    features:
        - id: dimensions
          value structure: atomic
          value type: numeric
          description: "Number of spatial dimensions (2)"
    description: "Two-dimensional display space with vertical
                flow direction for conveying dataflow relationships"

artifact graph_arrangement : arrangement(simplified_graph)
    context: layout_space
    principle: "hierarchical flow layout with edges bundled along
              the cluster hierarchy"
    description: "Operations and clusters positioned in a
                bottom-to-top flow layout supporting stable and responsive
                cluster expansion"

transform T_visualize_overview :
    intent: visualise
    manner: "display clustered flow layout with colour-coded repeated
            structures, auxiliary node side panel, embedded icons for
            extracted operations, and tensor size encoded on edges"
    input: graph_arrangement, structural_templates, graph_features
    output: graph_viz
    actor: machine
    description: "Render interactive overview showing top-level model
                components and their dataflow relationships"

artifact graph_viz : visualisation(graph_arrangement,
    structural_templates, graph_features)
    layout: "hierarchical flow layout with auxiliary side panel"
    form: "rounded rectangles for groups, ovals for operations,
          curved edges, embedded icons for extracted operations"
    encoding: "vertical position: dataflow direction; nesting:
              cluster hierarchy; colour: repeated module groups;
              edge thickness: tensor size; side panel: auxiliary
              nodes with proxy connections"
    description: "Overview of model architecture with
                expandable clusters, colour-coded repeated modules,
                auxiliary node panel, embedded icons, tensor-size edge
                encoding, and information card for node detail
                inspection"

```

```

# =====
# PHASE 5: INITIAL READING
# =====

transform T_initial_reading :
  intent: abstract
  manner: "recognise top-level model components, repeated module
    groups, and main-versus-auxiliary organisation from the
    overview"
  input: graph_viz
  output: architecture_patterns
  actor: human
  description: "Identify high-level architectural structure from
    the initial overview"

artifact architecture_patterns : pattern(simplified_graph)
  representation form: "recognised architectural features"
  description: "Identified architectural elements including
    top-level component organisation, repeated module groups,
    and main-versus-auxiliary structure"

# =====
# PHASE 6: INTERACTIVE EXPLORATION
# =====

loop L_exploration:
  purpose: "Progressively explore the model's hierarchical
    structure, modular composition, and operational details"
  until: "the analyst has sufficient understanding of the model
    architecture for their current task"
  body:

    transform T_explore :
      intent: visualise
      manner: "expand group nodes to reveal nested subgraphs;
        select nodes for attribute and connection detail;
        switch colour encoding among structural similarity,
        device distribution, compute time, or memory"
      input: graph_viz, graph_arrangement, graph_features,
        structural_templates
      output: updated_graph_viz
      actor: hybrid
      description: "Adjust the visualization by expanding or
        collapsing clusters, inspecting node details, or
        switching quantitative overlay"

    artifact updated_graph_viz : visualisation(graph_arrangement,
      structural_templates, graph_features)
      layout: "hierarchical flow layout with expanded clusters"
      form: "rounded rectangles for groups, ovals for operations,
        curved edges, embedded icons"
      encoding: "same as graph_viz with expanded cluster detail;

```

```

        colour overlay switchable among structural similarity,
        device distribution, compute time, or memory"
    description: "Updated visualization reflecting expanded
        clusters, selected node details, or alternative
        quantitative overlay"

assign: graph_viz := updated_graph_viz

transform T_identify_structure :
    intent: abstract
    manner: "identify layer organisation, repeated modules,
        data flow paths, and nesting relationships"
    input: graph_viz
    output: updated_architecture_patterns
    actor: human
    description: "Recognise architectural patterns and modular
        composition from the current view"

artifact updated_architecture_patterns :
    pattern(simplified_graph)
    representation form: "recognised architectural features"
    description: "Identified architectural elements including
        layer types, repeated modules, data flow paths, and
        nesting relationships"

assign: architecture_patterns := updated_architecture_patterns

transform T_assess_understanding :
    intent: assess
    manner: "judge completeness of structural understanding
        and identify unexplored areas"
    input: architecture_patterns, graph_viz
    output: understanding_assessment
    actor: human
    description: "Evaluate whether the model structure is
        sufficiently understood for the analyst's task"

artifact understanding_assessment :
    knowledge(architecture_patterns)
    representation form: "quality judgment"
    description: "Assessment of whether model architecture
        is sufficiently understood, identifying areas
        requiring further exploration"

end loop L_exploration

# =====
# PHASE 7: KNOWLEDGE SYNTHESIS
# =====

transform T_synthesize :
    intent: generate-knowledge

```

```

manner: "synthesise architectural patterns, modular composition,
and quantitative characteristics into coherent structural
understanding"
input: architecture_patterns, understanding_assessment
output: model_understanding
actor: human
description: "Formulate understanding of the model architecture
for development, debugging, or communication"

artifact model_understanding : knowledge(simplified_graph)
representation form: "statements and diagrams"
description: "Documented description of the model's
high-level structure, modular composition, layer
organisation, and data flow paths, supporting
development, debugging, and communication"

```

C.17 What-If Probing of ML Models

Source:

Wexler, James, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda B. Viégas and Jimbo Wilson.

“The What-If Tool: Interactive Probing of Machine Learning Models.”

IEEE Transactions on Visualization and Computer Graphics 26 (2019): 56-65. doi: 10.1109/TVCG.2019.2934619

Workflow Summary

Although this paper describes a tool rather than a single analytical methodology, it is possible to extract an assumed composite workflow by synthesising the tool’s three core workflows (Section 6: sense-making, hypothetical probing, fairness evaluation) with the analytical progressions demonstrated in the three case studies (Sections 5.1–5.3). Each case study follows a different path through the tool’s capabilities, and together they describe a coherent analytical pattern for black-box ML model understanding.

This assumed workflow captures the typical analytical process enabled by the What-If Tool for black-box ML model understanding, synthesised from the tool’s three core workflows and three case studies.

Data Understanding. The analyst examines feature distributions in the test dataset, identifying imbalances, anomalous distributions, and data quality issues that may influence model behaviour.

Model Performance Overview. Model predictions are computed on the test dataset and visualised through customisable multi-view displays — confusion matrices, score comparison scatterplots, feature-based histograms, and small multiples. The analyst identifies global performance patterns, regions of model agreement and disagreement, and areas warranting deeper investigation, forming initial observations of model behaviour.

Hypothetical Investigation. The analyst selects data points, features, and sub-groups to probe, then iteratively investigates model behaviour through feature editing

with re-inference, nearest counterfactual identification, and partial dependence analysis. Each cycle reveals feature sensitivities, unexpected responses, and decision boundary characteristics. The analyst assesses whether behavioural understanding is sufficient and refines the investigation focus until key questions are answered.

Fairness Evaluation. The analyst slices the dataset by individual features and feature intersections, computing per-slice performance and fairness metrics. Initial visualisations reveal performance disparities across demographic subgroups. The analyst then iteratively explores threshold optimisation strategies (demographic parity, equal opportunity, equal accuracy), each time visualising the resulting performance changes and identifying how disparities shift under the new configuration, continuing until a satisfactory fairness–performance trade-off is found.

Knowledge Synthesis. The analyst integrates data characteristics, performance patterns, behavioural findings, and fairness assessments into a comprehensive understanding of model behaviour, feature effects, and fairness properties, supporting model improvement and deployment decisions.

A diagrammatic representation of the workflow can be seen in Fig. 18.

ATWL representation

```

workflow WhatIfToolModelProbing
  description: "Assumed composite workflow for black-box ML model
    understanding through interactive data exploration, hypothetical
    probing, and fairness evaluation, synthesised from the tool's
    core workflows and case studies"

  # Three-phase structure:
  #   Phases 1-2: Data and performance understanding
  #   Phases 3-4: Hypothetical probing
  #   Phases 5-6: Fairness evaluation and synthesis

  template: characterise → visualise → abstract →
    characterise → visualise → abstract →
    generate-knowledge (specify) →
    loop(characterise → visualise → abstract → assess →
      generate-knowledge (specify)) →
    characterise → visualise → abstract →
    generate-knowledge (specify) →
    loop(assess → visualise → abstract →
      generate-knowledge (select strategy)) →
    generate-knowledge

  # =====
  # GIVEN ARTIFACTS
  # =====

  artifact trained_models : model
    origin: given
    description: "One or two trained ML models (classification or

```

```

        regression) to be probed as black boxes"

artifact test_data : entities
  origin: given
  internal structure: elementary
  embedment: set
  description: "Sample dataset with ground truth labels for
    model evaluation"

# =====
# PHASE 1: DATA UNDERSTANDING
# =====

transform T_compute_statistics :
  intent: characterise
  manner: "calculate distributional statistics for numeric and
    categorical features; assess non-uniformity and missing
    values"
  input: test_data
  output: feature_statistics
  actor: machine
  description: "Compute summary statistics and distributions for
    all features in the test dataset"

artifact feature_statistics : feature(test_data)
  value structure: vector
  value type: numeric
  description: "Summary statistics and distributional properties
    for all features"

transform T_visualize_distributions :
  intent: visualise
  manner: "render histograms and cumulative distribution plots
    for all features, sortable by non-uniformity or missing
    value count"
  input: test_data, feature_statistics
  output: distribution_viz
  actor: machine
  description: "Display feature distributions for data quality
    assessment and understanding"

artifact distribution_viz : visualisation(test_data,
  feature_statistics)
  layout: "scrollable feature list sorted by non-uniformity
    or missing value count"
  form: "histograms and cumulative distribution plots"
  encoding: "x-axis: feature values; y-axis: frequency or
    cumulative proportion; sorting by distributional
    properties"
  description: "Distribution charts for all features with
    summary statistics, highlighting imbalances and
    anomalies"

```

```

transform T_identify_data_characteristics :
  intent: abstract
  manner: "recognise feature imbalances, skewed distributions,
    and potential quality issues"
  input: distribution_viz, feature_statistics
  output: data_characteristics
  actor: human
  description: "Identify notable data properties that may affect
    model behaviour"

artifact data_characteristics : pattern(test_data)
  representation form: "identified distributional characteristics"
  description: "Notable data properties including feature
    imbalances, anomalous distributions, and quality issues
    relevant to model evaluation"

# =====
# PHASE 2: MODEL PERFORMANCE OVERVIEW
# =====

transform T_compute_performance :
  intent: characterise
  manner: "run inference for all data points; compute accuracy,
    error rates, confusion matrices, score distributions, and
    model comparison metrics"
  input: trained_models, test_data
  output: performance_metrics
  actor: machine
  description: "Compute model predictions and derive performance
    metrics on the test dataset"

artifact performance_metrics : feature(test_data)
  value structure: vector
  value type: {numeric, categorical}
  description: "Model prediction scores and derived labels for
    all data points with aggregate performance metrics
    including accuracy, error rates, and confusion counts"

transform T_visualize_performance :
  intent: visualise
  manner: "display data points positioned, binned, and coloured
    by configurable combinations of features, predictions,
    and performance metrics; support confusion matrices, score
    distributions, histograms, scatterplots, and small
    multiples"
  input: test_data, performance_metrics
  output: performance_viz
  actor: hybrid
  description: "Create customisable visualisations of model
    predictions and performance"

```

```

artifact performance_viz : visualisation(test_data,
    performance_metrics)
    layout: "configurable multi-panel display"
    form: "confusion matrices, scatterplots, histograms, and
        small multiples"
    encoding: "position, binning, and colour configurable by
        combinations of features, predictions, and performance
        metrics"
    description: "Customisable visualisations of model predictions
        and performance including confusion matrices, score
        comparison scatterplots, and feature-based histograms"

transform T_identify_behavior :
    intent: abstract
    manner: "recognise regions of model agreement and disagreement,
        performance variations across subgroups, and areas
        warranting deeper investigation"
    input: performance_viz, data_characteristics
    output: performance_patterns, behavior_patterns
    actor: human
    description: "Identify global performance patterns and initial
        model behaviour characteristics"

artifact performance_patterns : pattern(test_data)
    representation form: "identified performance characteristics"
    description: "Global performance patterns including model
        agreement and disagreement regions, performance variations,
        and areas of interest"

artifact behavior_patterns : pattern(trained_models)
    representation form: "model behaviour observations"
    description: "Initial observations of model behaviour including
        feature sensitivity, prediction patterns, and areas
        requiring hypothetical investigation"

# =====
# PHASE 3: INVESTIGATION FOCUS
# =====

transform T_specify_focus :
    intent: generate-knowledge
    manner: "select investigation targets based on observed
        performance patterns and behavioural observations"
    input: performance_patterns, behavior_patterns
    output: investigation_focus
    actor: human
    description: "Determine which data points, features, and
        subgroups to investigate through hypothetical probing"

artifact investigation_focus : specification
    representation form: "investigation targets"
    description: "Selected data points, features, and subgroups

```

```

    to probe through hypothetical analysis"

# =====
# PHASE 4: HYPOTHETICAL INVESTIGATION (loop)
# =====

loop L_hypothetical:
    purpose: "Probe model behaviour through counterfactual
              reasoning, feature editing, and partial dependence analysis
              to understand feature effects and decision boundaries"
    until: "analyst has sufficient understanding of model behaviour
           for the current investigation questions"
    body:

        transform T_compute_hypotheticals :
            intent: characterise
            manner: "edit feature values and re-infer; find nearest
                   counterfactual examples via distance-based search;
                   compute partial dependence across feature ranges"
            input: trained_models, test_data, investigation_focus
            output: hypothetical_results
            actor: hybrid
            description: "Compute model responses to hypothetical
                       inputs based on current investigation focus"

        artifact hypothetical_results : feature(test_data)
            value structure: vector
            value type: numeric
            description: "Model predictions for edited data points,
                       counterfactual distances and examples, and partial
                       dependence values across feature ranges"

        transform T_visualize_hypotheticals :
            intent: visualise
            manner: "render edited data point predictions with
                   deltas, counterfactual comparisons with highlighted
                   feature differences, and partial dependence plots
                   for numeric and categorical features"
            input: test_data, hypothetical_results,
                 performance_metrics
            output: hypothetical_viz
            actor: machine
            description: "Display hypothetical analysis results for
                       feature effect and decision boundary exploration"

        artifact hypothetical_viz : visualisation(test_data,
                                                  hypothetical_results, performance_metrics)
            layout: "side-by-side original and edited data point panels
                   with partial dependence plots below"
            form: "data point cards with prediction deltas,
                 counterfactual comparison tables, and line/bar charts"
            encoding: "prediction deltas highlighted by magnitude;

```

```

        counterfactual feature differences highlighted;
        partial dependence: x-axis feature value, y-axis
        predicted score"
    description: "Visualisations of prediction changes from
        feature edits, counterfactual comparisons, and
        partial dependence plots showing feature effects
        across value ranges"

transform T_discover_patterns :
    intent: abstract
    manner: "identify feature sensitivities, unexpected
        responses, decision boundary characteristics, and
        salient features from counterfactual and partial
        dependence evidence"
    input: hypothetical_viz, hypothetical_results
    output: updated_behavior_patterns
    actor: human
    description: "Discover model behaviour patterns from
        hypothetical analysis results"

artifact updated_behavior_patterns : pattern(trained_models)
    representation form: "model behaviour observations"
    description: "Accumulated understanding of feature
        effects, decision boundaries, and unexpected model
        responses discovered through hypothetical probing"

assign: behavior_patterns := updated_behavior_patterns

transform T_assess_understanding :
    intent: assess
    manner: "judge completeness of behavioural understanding
        and identify remaining questions"
    input: behavior_patterns, hypothetical_viz
    output: understanding_assessment
    actor: human
    description: "Evaluate whether model behaviour is
        sufficiently understood for current questions"

artifact understanding_assessment :
    knowledge(behavior_patterns)
    representation form: "quality judgment"
    description: "Assessment of whether model behaviour is
        sufficiently understood, identifying remaining
        questions"

if understanding_assessment indicates further investigation needed:
    then:
        transform T_refine_focus :
            intent: generate-knowledge
            manner: "select new data points, features, or
                subgroups to probe based on discovered
                patterns and outstanding questions"

```

```

        input: behavior_patterns,
              understanding_assessment
    output: updated_investigation_focus
    actor: human
    description: "Refine investigation focus based
                 on findings and remaining questions"

    artifact updated_investigation_focus : specification
    representation form: "investigation targets"
    description: "Refined selection of data points,
                 features, and subgroups for continued
                 hypothetical probing"

    assign:
        investigation_focus := updated_investigation_focus
    else:
        exit loop L_hypothetical

    end loop L_hypothetical

# =====
# PHASE 5: FAIRNESS EVALUATION
# =====

transform T_compute_fairness :
    intent: characterise
    manner: "slice data by individual features and feature
            intersections; compute per-slice accuracy, error rates,
            confusion matrices, and fairness measures"
    input: test_data, performance_metrics
    output: fairness_metrics
    actor: machine
    description: "Compute intersectional performance and fairness
                 metrics across demographic subgroups"

    artifact fairness_metrics : feature(test_data)
    value structure: vector
    value type: numeric
    description: "Performance metrics and fairness measures for
                 intersectional subgroups including accuracy, false
                 positive and negative rates, and confusion matrices
                 per slice"

transform T_visualize_fairness :
    intent: visualise
    manner: "render per-slice confusion matrices, ROC curves, and
            performance tables with sortable fairness measures and
            interactive threshold controls"
    input: test_data, fairness_metrics
    output: fairness_viz
    actor: machine
    description: "Display fairness metrics and performance

```

```

        comparisons across demographic subgroups"

artifact fairness_viz : visualisation(test_data,
    fairness_metrics)
    layout: "tabular slice comparison with ROC curves and
        threshold controls"
    form: "per-slice confusion matrices, ROC curves, and
        sortable performance tables"
    encoding: "rows: demographic slices; cells: confusion
        counts and rates; ROC curves per slice; interactive
        threshold slider"
    description: "Per-slice performance tables, confusion
        matrices, and ROC curves with interactive threshold
        controls"

transform T_identify_disparities :
    intent: abstract
    manner: "recognise differential performance across slices,
        identify advantaged and disadvantaged groups, and
        characterise disparity structures"
    input: fairness_viz, fairness_metrics
    output: fairness_patterns
    actor: human
    description: "Identify performance disparities and fairness
        patterns across demographic subgroups"

artifact fairness_patterns : pattern(test_data)
    representation form: "identified fairness disparities"
    description: "Performance disparities across demographic
        subgroups including differentially advantaged and
        disadvantaged groups and disparity structures"

transform T_select_initial_strategy :
    intent: generate-knowledge
    manner: "choose threshold optimisation approach informed by
        observed disparity patterns"
    input: fairness_patterns
    output: strategy_direction
    actor: human
    description: "Select initial fairness optimisation strategy
        based on identified disparities"

artifact strategy_direction : specification
    representation form: "strategy selection"
    description: "Choice of fairness optimisation strategy to
        apply and rationale"

loop L_fairness_strategy:
    purpose: "Explore threshold optimisation strategies to find
        acceptable fairness-performance trade-offs across
        demographic subgroups"
    until: "analyst has identified a satisfactory fairness strategy

```

or concluded that threshold adjustment alone is insufficient"

body:

```
transform T_explore_thresholds :
  intent: assess
  manner: "adjust classification thresholds per
          demographic slice according to chosen fairness
          strategy and evaluate resulting trade-offs"
  input: fairness_viz, fairness_patterns,
         fairness_metrics, strategy_direction
  output: fairness_assessment, threshold_config
  actor: hybrid
  description: "Apply chosen fairness optimisation
               strategy and evaluate its impact on per-slice
               performance"

artifact fairness_assessment : knowledge(trained_models)
  representation form: "fairness evaluation"
  description: "Assessment of trade-offs between fairness
               constraints and overall performance under the
               current threshold strategy"

artifact threshold_config : specification
  representation form: "per-slice classification
                       thresholds"
  description: "Optimised classification thresholds per
               demographic slice under chosen fairness strategy"

transform T_visualize_strategy :
  intent: visualise
  manner: "render updated confusion matrices, ROC curves,
          and performance tables reflecting per-slice
          threshold adjustments"
  input: test_data, fairness_metrics, threshold_config
  output: updated_fairness_viz
  actor: machine
  description: "Display updated performance metrics under
               the current threshold configuration"

artifact updated_fairness_viz :
  visualisation(test_data, fairness_metrics)
  layout: "tabular slice comparison with ROC curves
          and threshold controls"
  form: "per-slice confusion matrices, ROC curves,
        and sortable performance tables"
  encoding: "same as fairness_viz with thresholds
            adjusted per strategy"
  description: "Performance tables, confusion
               matrices, and ROC curves reflecting threshold
               adjustments under the current fairness
               strategy"
```

```

assign: fairness_viz := updated_fairness_viz

transform T_evaluate_strategy_impact :
  intent: abstract
  manner: "recognise shifts in per-slice performance,
          identify remaining disparities, and compare with
          prior strategy results"
  input: fairness_viz, fairness_metrics
  output: updated_fairness_patterns
  actor: human
  description: "Identify how disparities changed under
               the current fairness strategy"

artifact updated_fairness_patterns : pattern(test_data)
  representation form: "identified fairness disparities"
  description: "Updated performance disparities across
               demographic subgroups reflecting threshold
               adjustments and remaining inequities"

assign: fairness_patterns := updated_fairness_patterns

if fairness_assessment indicates further strategy exploration needed:
  then:
    transform T_select_next_strategy :
      intent: generate-knowledge
      manner: "select alternative optimisation
              strategy informed by comparison of
              results across previously explored
              strategies"
      input: fairness_assessment, fairness_patterns
      output: updated_strategy_direction
      actor: human
      description: "Decide which fairness strategy
                   to explore next based on observed
                   trade-offs and remaining disparities"

    artifact updated_strategy_direction : specification
      representation form: "strategy selection"
      description: "Choice of next fairness
                   optimisation strategy to explore and
                   rationale"

    assign:
      strategy_direction := updated_strategy_direction
  else:
    exit loop L_fairness_strategy

end loop L_fairness_strategy

# =====
# PHASE 6: KNOWLEDGE SYNTHESIS

```

```

# =====

transform T_synthesize :
  intent: generate-knowledge
  manner: "integrate data characteristics, performance patterns,
    behavioural findings, and fairness assessments into
    comprehensive model understanding"
  input: behavior_patterns, performance_patterns,
    fairness_assessment, fairness_patterns,
    data_characteristics, understanding_assessment
  output: model_understanding
  actor: human
  description: "Synthesise understanding of model behaviour,
    performance, and fairness for model improvement decisions"

artifact model_understanding : knowledge(trained_models)
  representation form: "statements and recommendations"
  description: "Documented assessment of model behaviour,
    feature effects, decision boundaries, performance
    characteristics, and fairness properties, supporting
    model improvement and deployment decisions"

```

D Diagrammatic Representations of Visual Analytics Workflows

E ATWL Workflow Examples — Correctness and Consistency Review

Below is a systematic audit of all 16 workflows against the ATWL syntax and semantics. Issues are organised into cross-cutting systematic problems (affecting multiple workflows) and workflow-specific problems.

E.1 Cross-Cutting Systematic Issues

S1. Feature artifacts missing mandatory value structure

The `feature` type requires *value structure*: `<atomic| homogeneous vector| heterogeneous vector| relational configuration>`. Several workflows substitute *representation form* (a field belonging to `pattern`, `model`, `knowledge`, and `specification`) in its place, omitting value structure entirely.

Note: Some workflows (e.g., 1.6) correctly use both value structure and representation form on features; the issue arises only when value structure is absent.

Workflow	Affected artifacts
1.5	<code>similarity_scores</code> , <code>cohort_analysis</code> , <code>plan_estimation</code>
1.10	<code>feature_relevance</code> , <code>updated_feature_relevance</code>
1.12	<code>data_statistics</code> , <code>modeling_possibilities</code>
1.13	<code>prediction_scores</code> , <code>instance_explanations</code> , <code>group_statistics</code>
After 1.15	<code>graph_features</code>
1.16	<code>feature_statistics</code> , <code>performance_metrics</code> , <code>hypothetical_results</code> , <code>fairness_metrics</code>

S2. Visualisation artifacts using representation form instead of layout / form / encoding

The `visualisation` type requires *layout*, *form*, and (recommended) *encoding*. Several workflows use a single *representation form* field instead.

S3. Entities artifacts missing embedment

Every `entities` artifact should declare how entities are positioned relative to each other via *embedment*. Multiple workflows omit this field.

S4. generate-knowledge intent producing specification outputs

The `generate-knowledge` intent should produce `knowledge` artifacts. In many workflows the analyst’s decision step is modelled with this intent but the output is typed as `specification`. This is the single most pervasive intent–output mismatch.

Workflow	Affected artifacts
1.5	current_viz, similarity_viz, recommendation_viz, estimation_viz
1.10	relationship_viz, updated_relationship_viz
1.12	data_viz, problems_viz, model_viz
1.13	diagnostic_viz, updated_diagnostic_viz
1.15	graph_viz, updated_graph_viz
1.16	distribution_viz, performance_viz, hypothetical_viz, fairness_viz, updated_fairness_viz

Workflow	Affected artifacts
1.5	current_record, similar_cohort
1.10	measurement_data, training_data, validation_data
1.12	dataset
1.13	raw_data, training_data, test_data, explanation_groups
1.15	dataflow_graph, clustered_graph, simplified_graph, layout_space
1.16	test_data

Possible resolution: Either (a) introduce a dedicated intent (e.g., `specify`) or treat this as a valid manner specialisation of `generate-knowledge`; or (b) split the transform into two—one `generate-knowledge` producing the rationale (`knowledge`) and one producing the specification; or (c) allow `generate-knowledge` to legitimately produce `specification` artifacts when the analyst’s decision is the knowledge being generated.

S5. Model artifacts missing type parameters

ATWL syntax: `model(<artifact-IDs>)`. Several derived model artifacts omit the parameterisation.

Contrast with 1.11 where `M1 : model(F1, F3)` is correctly parameterised.

S6. Loops lacking assign statements for iterative update

ATWL uses explicit `assign` to express artifact replacement between iterations. Several loops rely on implicit re-derivation or implicit parameter change with no `assign` to close the feedback cycle. This means the loop body, if re-entered, would receive identical inputs, making convergence towards the `until` condition unexpressed.

S7. Loops missing explicit exit mechanism

ATWL loops use an `if-then-else` conditional with `exit loop` to terminate iteration. Two loops have no such construct—they simply end with `end loop`, leaving the termination mechanism implicit.

Workflow	Transforms producing specifications
1.5	T_specify_initial_plan → action_plan; T_refine_plan → updated_action_plan
1.8	T_specify_refinement → user_guidance
1.9	T_adjust_encoding → encoding_params_updated; T_decide_refinement → topic_refinement_strategy; T_formulate_query → user_query
1.10	T_specify_model → model_spec; T_decide_refinement → updated_model_spec
1.12	T_specify_problem → problem_spec
1.13	T_specify_improvements → updated_data_model_spec
1.14	T_select_component → selected_component; T_refine_boundaries → boundary_adjustments
1.16	T_specify_focus → investigation_focus; T_refine_focus → updated_investigation_focus; T_select_initial_strategy → strategy_direction; T_select_next_strategy → updated_strategy_direction

Workflow	Affected artifacts
1.10	regression_model, updated_regression_model
1.12	candidate_models
1.13	trained_classifier
1.14	trained_model (given—more excusable)
1.16	trained_models (given)

E.2 Workflow-Specific Issues

1.1 — Cluster-Calendar

- **Given artifact inside loop body.** D_calendar is declared with origin: given inside the body of L1. Given artifacts are exogenous to the workflow and should be declared at the workflow level. Their value does not change between iterations.

1.2 — Dynamic Network Exploration

- **Visualisation under-parameterised.** projection_view is typed visualisation(snapshot_arrangement) but the producing transform T_visualize_projection also takes network_snapshots as input. For traceability, the parameterisation should include all substantive inputs: visualisation(snapshot_arrangement, network_snapshots).

1.4 — EventFlow

- **Non-standard applies to field.** alignment_spec : specification contains applies to: T_align. This field is not part of the ATWL specification syntax (also occurs

Workflow	Loop	What is implicitly updated
1.1	L1	Clustering parameters (number of clusters, distance measure)
1.3	L_spatial_refinement	Proximity/flow-strength thresholds
1.3	L_temporal_refinement	Number of temporal clusters
1.6	L_cluster_refinement	Distance thresholds
1.9	L_distribution_analysis	Exploration focus / context subset

Workflow	Loop
1.13	L_exploration (inner diagnostic exploration loop)
1.15	L_exploration

in 1.8: `user_guidance`).

- **Feature parameterised by non-entity artifact.** `F_complexity : feature(V_overview)`—`V_overview` is a visualisation, not an entities artifact. The ATWL syntax for features is `feature(<entities-ID>)`. The complexity metrics describe the aggregation tree, so the correct parameterisation would be `feature(D_agg_tree)`.
- **define-unit producing an arrangement.** `T_aggregate` outputs both `D_agg_tree` (entities—appropriate for `define-unit`) and `A_agg` (arrangement). Producing an arrangement is more aligned with `contextualise`. Consider splitting into two transforms.

1.5 — EventAction

- **Workflow declaration colon.** `workflow EventActionPrescriptiveAnalytics:`—the trailing colon is inconsistent with other workflow declarations (also in 1.10, 1.12, 1.13).

1.6 — Extracting Significant Places

- **Incomplete embedment.** `movement_events` and `clustered_events` have `embedment : {set, time}` but their features include spatial position. The embedment should be `{set, time, space}` (or `{space, time}`).

1.7 — Progressive Clustering of Trajectories

- **assign outside loop body.** `assign: analysed_trajectories := all_trajectories` appears before `L_destination_clustering`. ATWL specifies that `assign` is used exclusively within loops. This initialisation should instead be handled by declaring `analysed_trajectories` as initially equal to `all_trajectories`, or by restructuring.
- **Artifact declared without producer or origin:** `given. analysed_trajectories` is declared as a standalone entities artifact but is neither given nor produced by a transform before the first assignment.

- **Cross-scope artifact reference.** `T_synthesize_findings` inputs `destination_assessment` (produced inside `L_destination_clustering`) and `route_assessment_subset` (produced inside `L_route_analysis_subsets`). While the last values persist after loop exit, the scoping of loop-internal artifacts is not explicitly addressed in ATWL.

1.8 — Human-Steered Topic Modelling

- **define-unit overloaded.** `T_initial_topic_modeling` (intent: `define-unit`) produces not only `topics` (entities—appropriate) but also `topic_keyword_weights` and `document_topic_weights` (both features). The feature computation is a characterisation concern. Consider splitting.
- **define-unit producing a feature.** `T_assign_topics` (intent: `define-unit`) produces `topic_assignments : feature(documents)`. Assigning each document to its dominant topic is a characterisation (computing a feature value), not creating/re-defining entities.

1.9 — Progressive Abstraction of Multivariate Temporal Data

- **Intent–output type mismatch** (`characterise` \rightarrow `pattern`). `T_encode_patterns` has intent `characterise` but produces `symbolic_patterns : pattern(episodes)`. SAX encoding is a data transformation (characterisation) and the output represents a description of each episode. If the result is a pattern, the intent should be `abstract`; if the intent is `characterise`, the output should be a feature.
- **Inconsistent typing of topics across workflows.** In 1.8, topics are typed as entities; here, `topics` is `pattern(symbolic_patterns)`. For cross-workflow comparability, the same analytical concept should receive the same type. Since topics serve as units of analysis, `entities` is more appropriate.
- **define-unit producing a pattern.** `T_merge_topics` (intent: `define-unit`) produces `topics_updated : pattern(...)`. If topics were typed as entities (as in 1.8), `define-unit` would be appropriate.
- **Given feature parameterised by derived artifact.** `context_info : feature(episodes)` with `origin: given` references `episodes`, which is a derived artifact. A given artifact should not depend on derived artifacts.

1.14 — Interactive Exploration of Random Forest

- **Non-standard internal structure values.** `examined_component` and `examined_component_updated` use `internal structure: hierarchical`. The ATWL standard values are: `elementary`, `group/cluster`, `episode`, `region`, `sequence`, `formation`. A hierarchy is a special case of `formation`.

- **Non-standard value structure values.** `component_similarities` uses value structure: `matrix`; `feature_distributions` uses value structure: `distribution`. Neither is in the standard set `{atomic, homogeneous vector, heterogeneous vector, relational configuration}`. A similarity matrix could be `relational configuration`; distributions could be `heterogeneous vector`.
- **Abbreviated internal structure.** `component_groups` uses internal structure: `group` instead of the standard `group/cluster`.
- **Non-standard continue loop construct.** `continue loop L_progressive_selection` and `continue loop L_component_refinement` appear in conditional branches. ATWL defines `exit loop` but does not define `continue loop`. The standard pattern is for the `then` branch to perform work and fall through to the next iteration, with `else` calling `exit loop`.

1.15 — Exploring Deep Learning Models

- **Non-standard internal structure.** `dataflow_graph` uses internal structure: `network`. The standard value for networks is `formation`.

1.16 — What-If Probing of ML Models

- **assess producing a specification.** `T_explore_thresholds` (`intent: assess`) produces both `fairness_assessment` (`knowledge—correct`) and `threshold_config` (`specification`). The `assess` intent should produce evaluative knowledge only. The threshold configuration is a by-product that should be produced by a separate transform or should be an input specification that gets updated.

E.3 Summary

Issue ID	Category	Workflows affected
S1	Feature missing value structure	1.5, 1.10, 1.12, 1.13, 1.15, 1.16
S2	Visualisation using representation form	1.5, 1.10, 1.12, 1.13, 1.15, 1.16
S3	Entities missing embedment	1.5, 1.10, 1.12, 1.13, 1.15, 1.16
S4	<code>generate-knowledge</code> → <code>specification</code>	1.5, 1.8, 1.9, 1.10, 1.12, 1.13, 1.14, 1.16
S5	Model missing type parameters	1.10, 1.12, 1.13, 1.14, 1.16
S6	Loop missing assign	1.1, 1.3, 1.6, 1.9
S7	Loop missing explicit exit	1.13, 1.15

Observation on systematic clustering: Issues S1, S2, and S3 affect exactly the same six workflows (1.5, 1.10, 1.12, 1.13, 1.15, 1.16), suggesting these were produced by a different LLM agent configuration or prompt than the first group of workflows (1.1–1.4, 1.6–1.9, 1.11) which are largely free of these three issues.

Workflows with no issues or only minor ones: 1.2 (one under-parameterisation) and 1.11 (clean).

F Essential Instructions for ATWL Workflow Extraction

F.1 Core Principles

F.1.1 Appropriate Abstraction Level

ATWL represents the analytical logic, not implementation details.

Capture:

- What analytical questions are asked
- What operations are performed and why
- How artifacts flow through the analysis
- Iterative refinement and assessment patterns

Omit:

- Specific algorithms (k-means, DBSCAN → “partition-based clustering”, “density-based clustering”)
- Tool names (Python, Tableau, D3.js)
- Exact parameters (k=5, $\epsilon=0.3$ → handled as specifications or implicit)
- Implementation details (SQL queries, data formats, rendering methods)

Guideline: If comparing this workflow to 10 similar workflows, what level of detail helps identify patterns and differences in analytical **approach** without getting lost in implementation **specifics**?

F.1.2 Focus on Workflow Structure, Not Results

Artifact descriptions should describe **ROLES** and **TYPES**, not specific findings.

Too specific (describes results):

```
artifact movement_patterns : pattern(trajectories)
  description: "23% are local trips; three route types: peripheral,
    radial inward, radial outward"
```

Appropriate (describes role):

```
artifact movement_patterns : pattern(trajectories)
  representation form: "categorized patterns with characteristics"
  description: "Identified movement patterns: trip distance
    distributions, route types, spatial organization"
```

Why: Different datasets yield different results, but workflow structure remains comparable.

F.1.3 Tangible Artifact Representations

All artifacts must represent explicit, tangible outputs, not mental states.

The representation form field of knowledge, pattern, and other artifacts should describe a concrete, communicable form — not an analyst’s internal understanding.

Wrong (mental state):

```
artifact insights : knowledge(data)
  representation form: "understanding of dynamics"
```

Correct (tangible output):

```
artifact insights : knowledge(data)
  representation form: "statements and explanations"
```

Acceptable representation forms include:

- "statements and explanations"
- "quality judgment"
- "statements and diagrams"
- "statements and recommendations"
- "diagnostic statements"
- "categorized patterns with descriptions"
- "parameter settings"
- "recorded modelling interests and goals"

Avoid:

- "understanding", "comprehension", "awareness"
- "architectural understanding", "diagnostic understanding"
- "synthesized understanding"

F.1.4 Prefer Succinct Descriptions

Descriptions and manner fields should be **concise yet clear**. Include enough detail to convey the analytical role and method type, but omit redundant qualifiers, elaboration that restates what is already captured by the intent or artifact type, and unnecessary detail that does not aid workflow comparison.

Too verbose:

```
manner: "partition individual feature domains and feature-pair domains
  into disjoint regions; approximate local conditional target
  distributions; compute goodness-of-fit relevance measures at
  multiple partition complexities"
```

Succinct:

```
manner: "partition feature domains and compute multi-resolution  
relevance measures for conditional target distributions"
```

Guideline: If removing a phrase does not change the reader's understanding of what the transform does or what role the artifact plays, remove it.

F.2 Syntax Reference

This section summarises the key syntactic rules. Refer to the full ATWL definition document for complete details.

F.2.1 Workflow Declaration

```
workflow <workflow-ID>  
  template: <intent> → <intent> → loop(...) → ...  
  description: "..."
```

No trailing colon after the workflow identifier.

Wrong:

```
workflow MyWorkflow:
```

Correct:

```
workflow MyWorkflow
```

F.2.2 Comment Syntax

Use # for comments. Do **not** use //.

```
# This is a correct comment  
  
// This is NOT valid ATWL
```

F.2.3 Transform Field Order

Transforms should follow this field order:

```
transform <ID> :  
  intent: <generic-intent>  
  manner: "..."  
  input: <artifact-IDs>  
  output: <artifact-IDs>  
  actor: human | machine | hybrid  
  description: "..."
```

Every transform **must** have a `description:` field.

F.2.4 Entities: Internal Structure Keywords

Use only these standard values for `internal structure`:

- `elementary` — indivisible entity
- `group/cluster` — unordered collection of components
- `episode` — bounded time interval
- `region` — bounded spatial extent
- `sequence` — linearly ordered components
- `formation` — network, hierarchy, tree, graph, or other relational structure

Common mistakes:

- `network` → use `formation`
- `hierarchical` → use `formation`
- `group` (without `/cluster`) → use `group/cluster`

F.2.5 Entities: Embedment

Omit embedment: when the artifact represents a single entity (e.g., one aggregation tree, one projection space, one network snapshot). Embedment describes how *multiple* entities are positioned relative to each other; it is inapplicable when there is no collection of peer entities.

Wrong:

```
artifact projection_space : entities
  internal structure: elementary
  embedment: space           # single reference frame, not a collection
```

Correct:

```
artifact projection_space : entities
  internal structure: elementary
  features:
    - id: axes
      value structure: atomic
      value type: numeric
      description: "Number of spatial dimensions (2)"
  description: "Two-dimensional projection space serving as
  reference frame"
```

F.2.6 Unified Feature Syntax

The same two-level scheme applies to both internal features (inside `entities` artifacts) and standalone feature artifacts:

```

# Internal feature (inside entities)

features:
- id: <feature_id>
  value structure: <structure>
  value type: <type>           # optional
  description: "..."/>

```

```

# Standalone feature artifact

artifact <ID> : feature(<artifact-ID>)
  value structure: <structure>
  value type: <type>           # optional
  representation form: "..."/>

```

Value structures:

- **atomic** — single value per entity
- **list** — ordered or unordered enumeration of values
- **vector** — fixed-length array of values
- **matrix** — two-dimensional array indexed by two dimensions
- **relational configuration** — irregular relational structure (graph, tree, network)

Value types:

- **numeric, ordinal, categorical, temporal, spatial, text, reference**

When components are of mixed types, use set notation: `value type: {numeric, temporal}`. May be omitted when types are evident from context.

Note: Use `reference` for entity pointers, `categorical` for binary indicators, `relational configuration` for structural features.

Feature representation form. The `representation form:` field on feature artifacts is **optional** and should be used only when `value structure:` alone is ambiguous about the concrete encoding. Typical uses:

- **relational configuration** — e.g., "k-NN graph", "directed flows with time series"
- **vector** — e.g., "time series per place", "daily measurement profile"
- **matrix** — e.g., "pairwise similarity matrix", "term-document matrix"

Do **not** use `representation form:` as a substitute for `value structure:`, and omit it for atomic and list structures where `value type:` is sufficient.

F.2.7 Visualisation Artifact Syntax

Visualisation artifacts **must** use `layout:`, `form:`, and `encoding:` fields. Do **not** use `representation form:` for visualisations.

Wrong:

```
artifact my_viz : visualisation(data)
  representation form: "interactive scatterplot"
```

Correct:

```
artifact my_viz : visualisation(data)
  layout: "2D scatterplot"
  form: "colored points with connecting lines"
  encoding: "position from arrangement; color by cluster;
    lines connect consecutive items"
  description: "Interactive scatterplot showing ..."
```

F.2.8 Model Artifact Parameterisation

Model artifacts should declare what they are built from:

```
artifact my_model : model(training_data, model_spec)
  description: "..."
```

Given models (pre-trained, exogenous) may omit parameterisation:

```
artifact pretrained : model
  origin: given
  description: "..."
```

F.3 Artifact Types: Specification, Pattern, Knowledge

F.3.1 specification Artifact Type

Purpose: Parameters, settings, constraints that control how transforms operate.

When to use:

- Algorithm parameters being refined (distance thresholds, k values)
- User guidance/constraints (reference specifications, desired properties)
- Methodological choices (distance functions, aggregation strategies)
- Any setting that is iteratively adjusted or optimised

Creation:

- Use **assess** intent when derived from evaluation/optimisation
- Use **generate-knowledge** intent when analyst makes strategic decision

Required field: Every specification must have a `representation form:` field.

Example:

```

artifact clustering_params : specification
  origin: given
  representation form: "parameter settings"
  description: "Distance thresholds and sensitivity parameters
    for density-based clustering"

```

Note: The relationship between a specification and the transforms it controls is implicit from the dataflow — the specification appearing as input to those transforms. Do not add an `applies to:` field; this is not valid ATWL syntax.

F.3.2 assess Intent

Purpose: Evaluate quality/adequacy of results; decide if refinement needed.

Produces: knowledge artifacts (quality judgments), optionally **specification artifacts** (derived parameters from evaluation or optimisation)

When to use:

- Evaluating cluster/topic/aggregation quality
- Checking model performance
- Assessing visualisation effectiveness
- Deciding whether to continue refinement
- Optimising parameters (output includes updated specification)

Do NOT use for:

- Discovering patterns in data (use **abstract**)
- Final pattern characterisation (use **abstract**)

Example (knowledge only):

```

transform T_assess_clusters :
  intent: assess
  manner: "evaluate cluster quality"
  input: cluster_viz, clusters
  output: cluster_assessment
  actor: human
  description: "Evaluate whether clusters are spatially coherent
    and whether refinement is needed"

artifact cluster_assessment : knowledge(clusters)
  representation form: "quality judgment"
  description: "Assessment of cluster spatial coherence and
    whether refinement is needed"

```

Example (knowledge + specification):

```

transform T_explore_thresholds :
  intent: assess
  manner: "adjust thresholds per strategy and evaluate trade-offs"
  input: fairness_viz, fairness_metrics, strategy_direction
  output: fairness_assessment, threshold_config
  actor: hybrid
  description: "Apply fairness optimisation strategy and evaluate
    its impact on per-slice performance"

artifact fairness_assessment : knowledge(trained_models)
  representation form: "fairness evaluation"
  description: "Assessment of fairness-performance trade-offs"

artifact threshold_config : specification
  representation form: "per-slice classification thresholds"
  description: "Optimised thresholds under chosen strategy"

```

F.3.3 generate-knowledge Intent and Specification Outputs

Purpose: Formulate knowledge artifacts from patterns, models, and visual evidence. May also produce specification artifacts when the analyst's primary act is to decide on a course of action.

Produces: knowledge artifacts (insights, conclusions), and/or **specification artifacts** (methodological decisions)

When to use for specification output:

- Analyst selects a modelling strategy
- Analyst chooses features to include
- Analyst sets parameters for the next iteration
- Analyst formulates investigation focus or query criteria

Example:

```

transform T_adjust :
  intent: generate-knowledge
  manner: "adjust clustering parameters based on assessment"
  input: cluster_assessment, cluster_viz, S_clustering
  output: S_clustering'
  actor: human
  description: "Adjust clustering parameters: modify number of
    clusters, select distance measure, or change focus"

artifact S_clustering' : specification
  representation form: "parameter settings"
  description: "Updated clustering parameters after refinement"

```

F.3.4 loop(...) in Templates

Purpose: Show iterative portions of workflow in template summary.

Syntax: loop(intent -> intent -> ...)

When to use:

- Iterative refinement is methodologically important
- Assessment-driven parameter adjustment
- Multiple transforms iterate together
- Helps readers understand structure

Constraint: Templates permit only one level of loop(...) notation. **Nested loops are discouraged in templates.** When a workflow has nested iteration (e.g., an inner exploration loop inside an outer improvement loop), retain the outer — more analytically significant — loop in the template and flatten the inner loop to a single representative pass. The inner loop remains fully specified in the workflow body.

Example — workflow with nested iteration:

```
# WRONG (nested loops in template):  
  
template: loop(build-model → loop(visualise → abstract → assess)  
              → generate-knowledge) → generate-knowledge  
  
# CORRECT (inner loop flattened in template):  
  
template: loop(build-model →  
              visualise → abstract →  
              visualise (explore) → abstract → assess →  
              generate-knowledge (diagnose) →  
              generate-knowledge (specify)) →  
              generate-knowledge (finalize)
```

Annotation convention: When the same intent appears multiple times in a template with distinct analytical roles, use brief parenthetical annotations to distinguish them:

```
template: loop(generate-knowledge (specify) → build-model →  
              visualise → abstract → assess →  
              generate-knowledge (diagnose)) →  
              generate-knowledge (select)
```

Annotations should capture the distinct role concisely (e.g., `specify`, `diagnose`, `select`, `explore`, `refine`).

Artifact	Type	Key Characteristics	Example
Parameters, settings	specification	Controls operations; prescriptive	"k=7", "use route similarity"
Quality judgments	knowledge	Evaluative statements; "is this good?"	"Clusters lack coherence"
Discovered structures	pattern	Observed regularities; "what exists?"	"Three cluster types identified"
Domain insights	knowledge	Interpreted understanding	"Peak usage at 8am"
Strategic decisions	specification	Methodological choices	"Use density-based clustering"

Table 5: Artifact type characteristics

F.4 Artifact Type Decision Guide

F.4.1 Is it specification, pattern, or knowledge?

F.4.2 Decision Tree for Ambiguous Cases

Question 1: Does it control how a transform operates?

- YES → **specification**
- NO → Continue

Question 2: Is it a judgment about quality/adequacy?

- YES → **knowledge**
- NO → Continue

Question 3: Is it a discovered structure/regularity in the data?

- YES → **pattern**
- NO → **knowledge** (likely synthesised insight)

F.4.3 Concrete Entities vs. Patterns

When a transform identifies sets or groups of concrete objects (e.g., partitioning graph nodes by structural equivalence, grouping records by shared properties), the output is **entities** with **internal structure: group/cluster**, not a pattern. Patterns represent *abstracted regularities* (e.g., "morning peak," "three route types"), not the concrete objects themselves.

Wrong:

```
artifact structural_templates : pattern(graph)
  description: "Sets of graph nodes with identical structure"
```

Correct:

```

artifact structural_templates : entities
  internal structure: group/cluster
  embedment: set
  description: "Sets of graph nodes with identical nested
    structure, representing repeated module types"

```

F.4.4 Visualisation Inputs and References

`visualise` transforms consume data, feature, arrangement, pattern, and specification artifacts — **not other visualisation artifacts**. Visualisation artifacts are consumed by `abstract` and `assess` transforms, where the human interprets the display. When two views share layout or are visually coordinated, model this through shared data inputs, not by piping one visualisation into another.

Wrong:

```

transform T_visualize_overlay :
  intent: visualise
  input: base_visualization, new_data

```

Correct:

```

transform T_visualize_overlay :
  intent: visualise
  input: base_data, new_data, shared_features

```

Corollary: Parenthetical references on visualisation artifacts (e.g., `visualisation(...)`) should reference what is depicted — entities, features, patterns — not other visualisations.

F.4.5 Arrangement Context

The `context:` field of an arrangement identifies the reference structure *within which* entities are positioned. It must be a **separate entity**, not the same entity being arranged.

Wrong (self-referential):

```

artifact graph_arrangement : arrangement(graph)
  context: graph

```

Correct:

```

artifact layout_space : entities
  internal structure: elementary
  features:
    - id: axes
      value structure: atomic
      value type: numeric
      description: "Number of spatial dimensions (2)"
  description: "Two-dimensional display space"

artifact graph_arrangement : arrangement(graph)
  context: layout_space

```

```

principle: "similarity-preserving projection"
description: "..."

```

F.5 Intent Selection Guide

F.5.1 assess vs. abstract vs. generate-knowledge

Intent	Purpose	Typical Output	Example Transform
assess	Evaluate quality; decide refinement	knowledge (judgment) + optional specification	“Assess cluster coherence”
abstract	Discover patterns in data	pattern	“Identify route types”
generate-knowledge	Formulate insights or strategic decisions	knowledge + optional specification	“Synthesise findings” or “Decide clustering approach”

Table 6: Intent comparison

F.5.2 characterise vs. define-unit for Cluster Labels

When clustering produces both groups (entities) and per-entity membership labels (feature), both outputs come from the same `define-unit` transform. Cluster labels are an inherent by-product of creating clusters.

However, when a pre-existing model or grouping is *applied* to assign labels without creating new entities (e.g., assigning documents to their dominant pre-computed topic), the intent is `characterise` — it computes a feature, not new units.

define-unit (creates groups + labels):

```

transform T_cluster :
  intent: define-unit
  output: clusters, cluster_labels

```

characterise (assigns labels from existing grouping):

```

transform T_assign_topics :
  intent: characterise
  input: documents, topic_weights
  output: topic_assignments

```

F.5.3 Common Intent Mistakes

Using `abstract` for assessment:

```
# WRONG

transform T_assess :
  intent: abstract # This evaluates, not discovers
```

Using characterise for user decisions:

```
# WRONG

transform T_specify_guidance :
  intent: characterise # This is strategic decision-making
  output: user_guidance # specification
```

Using define-unit for pattern discovery:

```
# WRONG

transform T_find_patterns :
  intent: define-unit # Not creating entities
  output: patterns
```

Using define-unit for feature computation:

```
# WRONG

transform T_assign :
  intent: define-unit # Computing a feature, not creating entities
  output: topic_assignments # feature artifact

# CORRECT

transform T_assign :
  intent: characterise
  output: topic_assignments
```

F.6 Common Workflow Patterns

F.6.1 Pattern 1: Progressive Refinement with Assessment

Structure:

```
loop L_refinement:
  compute (uses parameters_spec) → visualise →
  assess (produces quality_assessment: knowledge) →
  if needed:
    adjust parameters (assess or generate-knowledge
                      produces parameters_spec')
```

Template:

```
template: loop(compute → visualise → assess) →  
         abstract → generate-knowledge
```

F.6.2 Pattern 2: User-Guided Interactive Refinement

Structure:

```
loop L_user_guided:  
  compute (uses user_spec) → visualise →  
  assess (produces quality_judgment: knowledge) →  
  if needed:  
    generate-knowledge (produces updated user_spec: specification)
```

Template:

```
template: loop(visualise → assess → generate-knowledge (specify)) →  
         abstract → generate-knowledge
```

F.6.3 Pattern 3: Multi-Phase Progressive Analysis

Structure:

```
loop L_phase1:  
  cluster by property A → assess → refine  
  
generate-knowledge (decide next approach: specification) →  
  
loop L_phase2:  
  cluster by property B → assess → refine  
  
abstract (discover final patterns) → generate-knowledge
```

Template:

```
template: loop(define-unit (phase1) → assess) →  
         generate-knowledge (strategy) →  
         loop(define-unit (phase2) → assess) →  
         abstract → generate-knowledge
```

F.6.4 Pattern 4: Hypothesis Generation and Verification

Structure:

```
analyze subset → abstract (hypothesis: pattern) →  
generate-knowledge (formulate verification approach: specification) →  
verify on full dataset → assess →  
exclude/refine based on findings → analyze remaining
```

Template:

```
template: define-unit (subset) → abstract (hypothesis) →
```

```

generate-knowledge (verification strategy) →
define-unit (verify on all) → assess →
define-unit (refine) → abstract → generate-knowledge

```

F.7 Loop Termination Patterns

Two legitimate termination patterns exist, depending on whether any transforms are conditionally gated.

F.7.1 Pattern A: Conditional with Gated Refinement Step

Use when a specific transform (typically refinement or specification) should execute only if the analyst decides to continue. One branch contains the gated transforms; the other branch contains exit loop.

```

loop L_refinement:
  purpose: "...
  until: "...
  body:
    transform T_compute : ...
    transform T_visualise : ...
    transform T_assess : ...
      output: assessment

    if assessment indicates refinement needed:
      then:
        transform T_refine : ...
          output: updated_spec
        assign: spec := updated_spec
      else:
        exit loop L_refinement
end loop L_refinement

```

Note: exit loop may appear in *either* branch — whichever is more natural given how the condition is phrased. For example:

```

# Also valid:

if assessment indicates satisfactory:
  then:
    exit loop L_refinement
  else:
    transform T_refine : ...

```

Post-loop transforms. Transforms that execute once after the loop terminates (e.g., final model selection, knowledge synthesis) may be placed either inside the exit branch (before exit loop) or after end loop. Placing them after end loop is preferred when the same transform would otherwise be duplicated across multiple exit points, or when

the transform is conceptually a separate analytical phase rather than the conclusion of the iterative process.

Important: `continue loop` is **not** valid ATWL syntax. Only `exit loop <ID>` is defined. When the analyst decides to continue iterating, the loop body simply falls through to the next iteration. If the gated work is in the `then` branch, leave the `else` branch empty (or with a comment) rather than writing `continue loop`.

```
# WRONG:
    else:
        continue loop L_selection

# CORRECT:
    else:
        # Continue exploring
```

F.7.2 Pattern B: No Conditional; `until`: Governs Termination

Use when all transforms execute every iteration and there is no refinement step to gate. The `until`: clause alone governs termination. This is appropriate when the stopping criterion reflects a gradual, holistic change in the analyst’s understanding that cannot be naturally reduced to a single assessable artifact.

```
loop L_exploration:
    purpose: "...
    until: "...
    body:
        transform T_explore : ...
        transform T_identify : ...
        transform T_assess : ...
end loop L_exploration
```

Choosing between them: Ask “Is there a specific transform that should only execute when the analyst decides to continue?” If yes, use Pattern A. If all transforms naturally execute every iteration, use Pattern B.

F.8 Loop Assignment Rules

When artifacts are updated across loop iterations, ATWL requires explicit `:=` assignment statements.

F.8.1 Rule 1: Pre-Loop Declaration and Initialisation

The target of a `:=` assignment must be a previously declared artifact — either `origin: given`, produced by a transform **before** the loop, or initialised by a pre-loop `assign` statement.

Do not create placeholder artifacts with artificial descriptions; ensure the initial version has proper provenance.

Pre-loop assignment for initialisation:

```
artifact D_all : entities
  ...

assign:
  D_current := D_all

loop L1:
  purpose: "Progressively analyse subsets"
  until: "All relevant subsets have been examined"
  body:
    transform T_process :
      input: D_current, ...
      output: D_next
      ...
    assign:
      D_current := D_next
end loop L1
```

Wrong — orphan artifact with no provenance:

```
artifact approach : knowledge(dataset)
  description: "Initial state: no prior attempts"

loop L_iteration:
  ...
  assign: approach := updated_approach
```

Correct — initial version produced by a pre-loop transform:

```
transform T_explore_data :
  intent: abstract
  ...
  output: data_characteristics, analytical_direction

artifact analytical_direction : knowledge(dataset)
  description: "Analyst's initial modelling interests based
    on data exploration"

loop L_iteration:
  ...
  assign: analytical_direction := updated_analytical_direction
```

F.8.2 Rule 2: Consumption Requirement

Every artifact updated via := must appear as input to at least one transform in the loop body. The assignment is only meaningful if the updated value feeds back into subsequent computation. If an artifact produced inside a loop's then branch feeds back to the next

iteration, verify that the corresponding `:= target` appears as input to a transform in the loop body.

F.8.3 Rule 3: Features Not Needed When Not Consumed

Internal features of entities artifacts (e.g., `cluster_size`) should be declared only when they are consumed by a downstream transform — for example, when cluster size affects visualisation encoding, assessment criteria, or selection decisions. Do not add features purely for documentation if no transform uses them.

F.9 Extraction Guidelines

F.9.1 Step 1: Identify Main Analytical Phases

Read paper’s methodology section. Look for:

- Data preparation steps
- Iterative refinement procedures
- Assessment/evaluation steps
- Interactive user guidance
- Pattern discovery and interpretation
- Knowledge synthesis

Create **workflow template** showing main phases with `loop(...)` where appropriate.

Tool-focused papers: When a paper describes a tool rather than a single methodology, construct an assumed composite workflow by:

1. Identifying the tool’s core analytical workflows or modes
2. Examining case studies or usage scenarios for concrete analytical progressions
3. Synthesising a representative workflow that captures the typical analytical process the tool enables
4. Noting in the workflow description that it is an assumed composite workflow

Prioritise case studies that demonstrate the fullest use of the tool’s capabilities, and choose a linearisation that reflects the most analytically coherent progression.

Paper Language	Likely ATWL Intent
“compute features”, “aggregate”, “calculate”	characterise
“extract”, “filter”, “cluster”, “segment”, “group”	define-unit
“arrange”, “position”, “layout”, “project”	contextualise
“display”, “visualise”, “render”, “show”	visualise
“identify patterns”, “discover”, “find structures”	abstract
“assess quality”, “evaluate”, “check adequacy”	assess
“train model”, “fit”, “learn”	build-model
“formulate insights”, “conclude”, “explain”	generate-knowledge
“assign labels”, “classify (existing model)”	characterise

Table 7: Mapping paper language to ATWL intents

F.9.2 Step 2: Map Paper Operations to ATWL Intents

F.9.3 Step 3: Identify Artifact Types

For each artifact created, determine:

Is it entities?

- Collection of analysis units (documents, trajectories, episodes, clusters)
- Has internal structure (elementary, sequence, episode, formation, etc.)
- Has embedment when there are multiple entities to relate (omit for single entities)
- Internal features use `value structure`: + optional `value type`:

Is it feature?

- Properties computed from entities
- Describes characteristics of entities
- Uses `value structure`: + optional `value type`:
- Do **not** use `representation form`: as a substitute for `value structure`:

Is it arrangement?

- Positioning of entities in a context

- References a context (a **separate** entities artifact serving as reference structure)

Is it visualisation?

- Visual representation for human perception
- References the data artifacts being depicted (entities, features, patterns — not other visualisations)
- Uses `layout:`, `form:`, `encoding:` fields (not `representation form:`)
- Consumed by **abstract** and **assess** transforms, not by other **visualise** transforms

Is it pattern?

- Discovered *abstracted* regularities/structures IN the data
- Typically produced by abstract intent
- Must be an abstraction, not concrete objects (use entities for concrete groups)

Is it knowledge?

- Judgments ABOUT results (from assess)
- Domain insights/understanding (from generate-knowledge)
- Interpreted statements, not just discovered patterns
- Must have tangible `representation form:` (not mental states)

Is it specification?

- Controls how operations behave
- Parameters, settings, constraints, user guidance
- Must have `representation form:`
- Relationship to controlled transforms is implicit from dataflow

Is it model?

- Trained/calibrated for prediction/simulation
- Generalises beyond observed data
- Parameterised by training inputs: `model(training_data, spec)`

F.9.4 Step 4: Handle Iterative Refinement

Paper describes parameter adjustment?

- Create explicit specification artifact (`origin: given` for initial values); create loop with `assess` or `generate-knowledge` producing updated specification; close the feedback with `assign`

Paper describes user guidance?

- Use `generate-knowledge` to create specification artifacts representing user input

Paper describes quality evaluation?

- Use `assess intent`, producing knowledge artifacts with judgments (and optionally specification artifacts)

Loop requires iteratively updated artifacts?

- Use `:=` assignment; ensure the target is declared before the loop with proper provenance

Loop has a gated refinement step?

- Use conditional Pattern A with `exit loop` in one branch; place one-time terminal transforms after `end loop`

Loop has no gated step; all transforms execute every iteration?

- Use Pattern B; `let until:` govern termination

F.9.5 Step 5: Maintain Appropriate Generality

Remove from descriptions:

- Specific numeric results (“23% are local trips”)
- Dataset-specific details (“red cluster”, “northwest region”)
- Exact parameter values in descriptions (put in specifications if iteratively refined)
- Tool/algorithm names (abstract to conceptual approach)

Keep in descriptions:

- Analytical roles and purposes
- Types of patterns/structures
- Evaluation criteria
- Methodological approaches

F.10 Common Mistakes and Corrections

F.10.1 Putting Results in Artifact Descriptions

Wrong:

```
artifact clusters : entities
  description: "Five clusters representing morning, afternoon,
    evening, night, and weekend patterns"
```

Correct:

```
artifact clusters : entities
  internal structure: group/cluster
  embedment: set
  description: "Temporal clusters grouping time periods with
    similar activity patterns"
```

F.10.2 Using pattern for Specifications

Wrong:

```
artifact user_constraints : pattern(topics)
  description: "User-specified desired topic characteristics"
```

Correct:

```
artifact user_constraints : specification
  representation form: "desired topic properties"
  description: "User-specified constraints defining desired
    topic properties"
```

F.10.3 Using pattern for Quality Judgments

Wrong:

```
artifact quality_assessment : pattern(clusters)
  description: "Assessment that clusters lack coherence"
```

Correct:

```
artifact quality_assessment : knowledge(clusters)
  representation form: "quality judgment"
  description: "Assessment that clusters lack spatial coherence
    and require refinement"
```

F.10.4 Over-Specifying Parameters

Too specific:

```
transform T_cluster :
```

```
manner: "k-means clustering with k=5, max_iter=300,
        init='k-means++'"
```

Appropriate abstraction:

```
transform T_cluster :
  manner: "partition-based clustering"
  input: data, clustering_spec
```

F.10.5 Missing Loop Notation in Template

Unclear:

```
template: define-unit → visualise → assess → generate-knowledge
# Hidden: there's iterative refinement in the body
```

Clear:

```
template: loop(define-unit → visualise → assess) →
           generate-knowledge
```

F.10.6 Visualisation Artifacts as Inputs to Visualise Transforms

Wrong:

```
transform T_visualize_overlay :
  intent: visualise
  input: base_visualization, new_data
```

Correct:

```
transform T_visualize_overlay :
  intent: visualise
  input: base_data, new_data, shared_features
```

F.10.7 Orphan Artifacts as Loop Pre-Declarations

Wrong:

```
artifact direction : knowledge(dataset)
  description: "Initial state: no prior attempts"

loop L_iteration:
  ...
  assign: direction := updated_direction
```

Correct:

```
transform T_initial_analysis :
  intent: abstract
```

```

...
output: initial_direction

artifact initial_direction : knowledge(dataset)
  description: "Analyst's initial analytical direction"

loop L_iteration:
  transform T_specify :
    input: ..., initial_direction
    ...
  ...
assign: initial_direction := updated_direction

```

F.10.8 Using representation form for Visualisations

Wrong:

```

artifact my_viz : visualisation(data)
  representation form: "interactive scatterplot with clusters"

```

Correct:

```

artifact my_viz : visualisation(data)
  layout: "2D scatterplot"
  form: "colored points"
  encoding: "position from projection; color by cluster"
  description: "Interactive scatterplot showing clusters"

```

F.10.9 Non-Standard Internal Structure Keywords

Wrong:

```

artifact graph : entities
  internal structure: network

artifact tree : entities
  internal structure: hierarchical

artifact groups : entities
  internal structure: group

```

Correct:

```

artifact graph : entities
  internal structure: formation

artifact tree : entities
  internal structure: formation

artifact groups : entities
  internal structure: group/cluster

```

F.10.10 Embedment on Single Entities

Wrong:

```
artifact agg_tree : entities
  internal structure: formation
  embedment: {set, time}    # single tree, not a collection
```

Correct:

```
artifact agg_tree : entities
  internal structure: formation
  # no embedment --- single entity
```

F.10.11 Using continue loop

Wrong:

```
    else:
      continue loop L_selection
```

Correct:

```
    else:
      # Continue exploring
```

F.10.12 Intangible Representation Forms

Wrong:

```
artifact insights : knowledge(data)
  representation form: "comprehensive understanding"
```

Correct:

```
artifact insights : knowledge(data)
  representation form: "statements and explanations"
```

F.11 Quality Checklist

Before finalising ATWL representation, verify:

- **Template** shows main analytical progression with `loop(...)` where appropriate; **no nested loops**; parenthetical annotations distinguish repeated intents
- **All transforms** have a `description:` field (required) and optionally `manner:`
- **Transform field order:** intent -> manner -> input -> output -> actor -> description

- **Intents** match purposes: assess for evaluation, abstract for discovery, characterise for feature computation (not define-unit)
- **Assessment outputs** are **knowledge** artifacts (judgments), optionally **specification**; not pattern
- **Parameters/settings** are **specification** artifacts when iteratively refined; have representation form::; no applies to: field
- **Discovered structures** are **pattern** artifacts from abstract intent (not concrete groups — those are entities)
- **Internal structure keywords** are standard: elementary, group/cluster, episode, region, sequence, formation
- **Embedment** omitted for single-entity artifacts
- **Feature syntax** uses value structure: + optional value type: (both internal and standalone); no representation form: as substitute
- **Visualisation syntax** uses layout:, form:, encoding: (not representation form:)
- **Model artifacts** parameterised: model(training_data, spec) (given models may omit)
- **Knowledge representation forms** are tangible (not mental states)
- **Descriptions** are succinct: focus on roles/types, not specific results; omit unnecessary detail
- **Abstractions** appropriate: conceptual approaches, not algorithm names
- **Actor types** specified (human, machine, hybrid) for all transforms
- **Artifact types** follow decision guides (specification vs. pattern vs. knowledge)
- **Generic terminology** used (not dataset-specific findings)
- **Visualisation inputs**: visualise transforms consume data artifacts, not other visualisations
- **Visualisation references**: parenthetical references cite depicted data artifacts, not other visualisations
- **Arrangement context**: references a distinct entity, not self-referential
- **Loop := targets** declared before the loop with proper provenance (no orphan placeholders)
- **Loop := targets consumed** by at least one transform in the loop body
- **No continue loop** — only exit loop <ID> is valid

F.12 Summary: Key Distinctions

Concept	Definition	Created By	Example
pattern	Abstracted regularity IN data	abstract	“Three route types: peripheral, inward, outward”
knowledge (from assess)	Quality judgment ABOUT results	assess	“Clusters lack coherence; refinement needed”
knowledge (from generate-knowledge)	Domain insights or strategic decisions	generate-knowledge	“Peak usage at 8am” or “Use density clustering”
specification	Control parameters FOR operations	assess or generate-knowledge	“k=7” or “route similarity distance function”

Table 8: Key concept distinctions

F.13 Final Guidance

When in doubt:

F.13.1 For artifact types:

Ask “What is its purpose?”

- Controls operations → specification
- Quality judgment → knowledge
- Abstracted regularity → pattern
- Concrete grouped objects → entities (group/cluster)
- Domain insight → knowledge

F.13.2 For intents:

Ask “What is the primary goal?”

- Evaluate quality → assess
- Discover patterns → abstract
- Formulate insights/decisions → generate-knowledge

- Create/modify units → define-unit
- Compute properties → characterise
- Assign labels from existing grouping → characterise (not define-unit)

F.13.3 For abstraction level:

Ask “What helps workflow comparison?”

- Keep: analytical approach, workflow structure, artifact roles
- Remove: specific algorithms, tools, exact parameters, dataset results

ATWL captures analytical recipes, not computational cookbooks.

G Instructions for ATWL Workflow Review

G.1 Role of the Reviewer

The reviewer evaluates ATWL workflow representations created from research papers. The review covers the following objectives:

1. **Verify correctness** against ATWL syntax and semantics
2. **Check alignment** with the source paper’s methodology
3. **Assess abstraction level** appropriateness
4. **Identify issues** and suggest specific corrections

Reviews should provide **constructive, specific feedback** with examples of corrections.

G.2 Review Framework Overview

The review proceeds through five phases:

1. Structural Validation
2. Semantic Correctness
3. Paper Alignment
4. Abstraction Level Assessment
5. Overall Quality Judgment

G.3 Phase 1: Structural Validation

G.3.1 Required Elements

Workflow Header. Verify that the workflow:

- Has a `workflow <ID>` declaration (no trailing colon)
- Has a `template:` with intent sequence
- Has a `description:` summarising purpose
- Template uses only valid intents or `loop(...)`
- Template matches the actual body structure (same sequence of intents, loops correspond to loop constructs)

Common issues include missing templates, templates that do not match the workflow body structure, trailing colons on the workflow declaration, and non-standard syntax in templates.

Artifact Declarations. For each artifact, verify:

- Has a valid type: `entities`, `feature`, `arrangement`, `visualisation`, `pattern`, `knowledge`, `specification`, `model`
- `origin: given` used only for exogenous artifacts not produced by any transform
- Artifacts without `origin: given` are outputs of some transform
- Artifact references in parentheses are valid (e.g., `feature(entity_ID)`)
- Comments use `#` syntax (not `//`)

Common issues include `origin: given` on artifacts that are transform outputs, missing `origin: given` on truly exogenous artifacts, invalid artifact type names, `//` comment syntax, and circular dependencies.

Entities Artifacts. For each `entities` artifact, verify:

- Has `internal structure:` with a standard value: `elementary`, `group/cluster`, `episode`, `region`, `sequence`, or `formation`
- Has `embedment:` when the artifact represents a collection of entities positioned relative to each other. Values (combinable with set notation): `set`, `sequence`, `time`, `space`, `relational structure`
- `embedment` is *omitted* when the artifact represents a single entity (e.g., a single aggregation tree, a single reference frame, a single network)
- Internal features use the unified two-level scheme: `value structure` (`atomic`, `list`, `vector`, `matrix`, `relational configuration`) and optional `value type` (`numeric`, `ordinal`, `categorical`, `temporal`, `spatial`, `text`, `reference`). Mixed types use set notation (e.g., `{numeric, temporal}`)
- Internal features are declared under a `features:` keyword
- Internal features are included only when they are consumed by downstream transforms; omit features that serve no analytical purpose in the workflow

Common issues include non-standard internal structure values (e.g., `network` instead of `formation`, `hierarchical` instead of `formation`, `group` instead of `group/cluster`); `embedment` on single entities (e.g., a projection reference space); missing `embedment` on genuine collections; old-style `value type:` without `value structure:` on internal features; missing `features:` keyword; and non-standard value types (e.g., `binary`, `hierarchical`, `set`).

Reference Space Entities. When a `contextualise` transform creates a reference frame (e.g., a 2D projection space), this is modelled as a single `entities` artifact:

- `internal structure: elementary`
- No `embedment` (single entity)
- Features describe the frame itself (e.g., number of dimensions), not positions within it

```

# CORRECT:

artifact projection_space : entities
  internal structure: elementary
  features:
    - id: axes

    value structure: atomic
    value type: numeric
    description: "Number of spatial dimensions (2)"
    description: "Two-dimensional projection space"

# WRONG:

artifact projection_space : entities
  internal structure: elementary
  embedment: space          # single entity, no embedment
  features:
    - id: coordinates      # describes positions, not frame

    value type: spatial

```

Feature Artifacts. For each feature artifact, verify:

- References an `entities` artifact: `feature(entities_ID)`
- Has `value structure:` using the same vocabulary as internal `features:` `atomic`, `list`, `vector`, `matrix`, `relational configuration`
- Optionally has `value type:` using standard types (single or set notation for mixed)
- Optionally has `representation form:` for additional clarification
- Does *not* use `representation form` as a substitute for `value structure`

Common issues include using `representation form` instead of `value structure`; referencing non-entity artifacts (e.g., `feature(visualisation_ID)`); and using the old single-field `value type:` syntax without `value structure:`.

Feature representation form. The optional `representation form:` on feature artifacts should be used only when `value structure:` alone is ambiguous about the concrete encoding:

- `relational configuration` — e.g., "k-NN graph", "directed flows with time series"

- `vector` — e.g., "time series per place", "daily measurement profile"
- `matrix` — e.g., "pairwise similarity matrix", "term-document matrix"

Do *not* use `representation form`: as a substitute for `value structure`:, and omit it for atomic and list structures where `value type`: is sufficient.

Visualisation Artifacts. For each visualisation artifact, verify:

- Has `layout`: describing spatial organisation
- Has `form`: describing mark or glyph types
- Has `encoding`: describing mapping of data attributes to visual channels (recommended)
- Does *not* use a single `representation form` field as a substitute for the three visualisation-specific fields
- References in parentheses point to data artifacts, not other visualisations

Common issues include using `representation form` instead of `layout/form/encoding`; and referencing other visualisation artifacts in parentheses.

Knowledge Artifacts. For each knowledge artifact, verify:

- Has `representation form`: describing a *tangible* output form (e.g., "statements and explanations", "quality judgment", "statements and recommendations")
- Does *not* describe a mental state (e.g., "understanding", "comprehension")

ATWL represents only explicit, tangible artifacts. Knowledge artifacts should describe recorded, communicable outputs.

Specification Artifacts. For each specification artifact, verify:

- Has `representation form`: describing the structure (e.g., "parameter settings", "query criteria", "model configuration")
- Description clarifies prescriptive nature (controls operations)
- Not marked with `origin: given` unless truly an external default
- Does *not* have an `applies to:` field (not valid ATWL syntax; the relationship to controlled transforms is implicit from the dataflow)

Common issues include missing `representation form`; using `pattern type` for specifications; and the non-standard `applies to:` field.

Model Artifacts. For each model artifact, verify:

- Has parameterisation in parentheses referencing artifacts from which the model is built: `model(training_data, spec)` — except for `origin: given` models where training inputs are external to the workflow

- Has a description:

Common issue: model without parameterisation for derived (non-given) models.

Transform Declarations. For each transform, verify:

- Has a valid intent: (define-unit, characterise, contextualise, visualise, abstract, assess, build-model, generate-knowledge)
- Has input: referencing declared artifacts
- Has output: declaring produced artifacts
- Has actor: (human, machine, or hybrid)
- Has description:
- Optional manner: is reasonable
- Consistent field ordering across the workflow (recommended: intent → manner → input → output → actor → description)

Common issues include invalid intent names, missing required fields (especially description), undeclared input/output artifacts, missing or invalid actor types, and inconsistent field ordering.

Control Structures. For loops, verify:

- Has loop <ID>: with unique ID
- Has purpose: describing analytical goal
- Has until: with qualitative termination condition
- Contains body: section
- Ends with end loop <ID> matching opening ID
- Assignment statements use := syntax
- Does *not* use continue loop (not a valid ATWL construct)

Common issues include missing purpose or until statements, quantitative thresholds in until instead of qualitative goals, mismatched loop IDs, use of the non-standard continue loop construct, and assignment syntax errors.

Loop Termination Patterns. Two termination styles are valid:

Pattern A: Explicit assessment with conditional exit. Used when the loop body contains an assess transform and a refinement step that should execute only when assessment indicates further work is needed:

- An assess transform produces evaluative knowledge

- An `if-then-else` conditional gates the refinement
- One branch contains substantive transforms (refinement, re-specification) typically with `:=` assignment
- The other branch contains `exit loop <ID>` (optionally preceded by one-time terminal transforms)
- `exit loop` may appear in either the `then` or `else` branch, depending on how the condition is phrased
- The branch opposite `exit loop` may be empty or contain only a comment when the loop simply re-enters (e.g., “continue exploring”); however, when a loop’s purpose involves parameter refinement, the non-exit branch must contain substantive transforms and an `assign` statement closing the feedback — comment-only branches that hide parameter adjustment are not permitted

```

# exit in else:

if assessment indicates refinement needed:
  then:
    transform T_refine : ...
    assign: spec := updated_spec
  else:
    exit loop L1

# exit in then (equally valid):

if assessment indicates satisfactory:
  then:
    exit loop L1
  else:
    transform T_refine : ...
    assign: spec := updated_spec

```

Pattern B: Implicit termination. Used when all transforms execute every iteration and the loop continues until the analyst is satisfied, with no conditionally gated transforms:

- No conditional needed; the `until:` clause governs termination
- Appropriate when the stopping criterion reflects gradual, holistic change in the analyst’s understanding that cannot be naturally reduced to a single assessable artifact
- An `assess` transform may still appear in the body (its output feeds post-loop synthesis) but does not gate any conditional

Common issues include:

- Conditionals where both branches execute no differential transforms (the loop should use Pattern B instead)
- Comment-only non-exit branches that *hide parameter adjustment* — if the paper describes iterative parameter tuning, the branch must contain substantive transforms with `assign`
- Empty or comment-only non-exit branches in *exploration* loops where iteration is driven by the analyst choosing different facets to examine — such loops should use Pattern B instead, removing the conditional entirely
- Missing conditionals when a refinement step should be gated
- Using the non-standard `continue loop` construct instead of letting the loop body fall through naturally

Assignment Contexts. Assignments appear in two valid contexts:

- **Inside a loop body**, to express iterative update
- **Before a loop**, to initialise an artifact identifier that will be re-assigned within the loop

```
# Pre-loop initialisation:
assign:
  D_current := D_all

loop L1:
  ...
  assign:
    D_current := D_next
end loop L1
```

Post-Loop Terminal Actions. Transforms that execute once after iteration is complete (e.g., final model selection, knowledge synthesis) may be placed either inside the exit branch (before `exit loop`) or after `end loop`. Placing them after `end loop` is preferred when the same transform would otherwise be duplicated across multiple exit points, or when the transform is conceptually a separate analytical phase rather than the conclusion of the iterative process.

Nested Loops.

- Nested loops in the workflow **body** are permitted when the paper describes genuinely nested iteration
- Nested loops in the **template** are **not** permitted; the template must flatten nested iteration to a single loop level

G.3.2 Artifact Dependency Graph

Verify no cycles exist except in loops with explicit assignment:

- Each artifact’s dependencies should form an acyclic graph
- Circular dependencies only through explicit `assign:` statements in loops
- All input artifacts should be declared before use (or produced by earlier transforms)

Artifact Provenance. Every artifact must have exactly one of:

- `origin:` `given` (exogenous input)
- A producing transform (appears in some transform’s `output:`)
- An `assign:` statement binding it to another artifact (for pre-loop initialisations or iterative updates within loops)

Artifacts that have none of these are *orphan artifacts*.

Dead-End Artifacts. Every non-terminal artifact should be consumed as input by at least one downstream transform. Artifacts that are produced but never consumed indicate broken dataflow. This is particularly critical inside loop conditionals, where a refinement artifact produced in the `then` branch must be consumed in the next iteration (via `:=` assignment to an artifact that appears in a loop-body transform’s input list).

Exception: Terminal knowledge artifacts (the final output of the workflow) are naturally not consumed by any downstream transform. The dead-end check applies to intermediate artifacts only.

G.4 Phase 2: Semantic Correctness

G.4.1 Intent Appropriateness

For each transform, verify the intent matches its purpose:

define-unit Creating or redefining analysis units.

- Should: Extract, filter, cluster, segment, partition, merge, group, select subsets
- × Should not: Compute features, evaluate quality, discover patterns

characterise Computing features or descriptors of entities.

- Should: Calculate attributes, aggregate statistics, compute distances, encode values, apply model for prediction, assign labels from existing groupings
- × Should not: Create entities, make decisions, evaluate quality

contextualise Arranging entities in reference structures.

- Should: Position in space/time/projection, create arrangements with a reference context
- Must produce: An **arrangement** artifact (and optionally the reference space entity)
- × Should not: Just transform coordinates without creating an arrangement

visualise Creating visual representations.

- Should: Produce **visualisation** artifacts from data, feature, arrangement, pattern, and specification artifacts
- × Should not: Produce non-visual artifacts as primary output
- × Should not: Take other visualisation artifacts as input — *except* in interactive exploration loops where a **visualise** transform updates a previous view (e.g., expanding clusters, switching overlays). In such cases, the previous visualisation state is a legitimate input, and the updated visualisation is assigned back via :=

abstract Discovering or refining patterns in data.

- Should: Identify structures, regularities, motifs; interpret cluster meanings; synthesise observations into named patterns; consolidate or refine existing patterns (e.g., merging semantically similar topics)
- Produces: **pattern** artifacts
- × Should not: Evaluate quality (use **assess**), make strategic decisions, produce concrete groups of identified objects (use **define-unit**)

assess Evaluating quality of results.

- Should: Judge adequacy, evaluate quality, decide if refinement needed
- Produces: **knowledge** artifacts (evaluative judgments); may also produce **specification** artifacts (parameter adjustments triggered by assessment)
- × Should not: Discover patterns in data (use **abstract**)

build-model Constructing predictive or explanatory models.

- Should: Train, calibrate, fit models
- Produces: **model** artifacts
- × Should not: Evaluate model quality (use **assess**)

generate-knowledge Formulating explicit knowledge or methodological decisions.

- Should: Synthesise insights, formulate conclusions, make strategic decisions about analysis direction
- Produces: **knowledge** artifacts; may also produce **specification** artifacts when the analyst's primary act is formulating a methodological decision
- × Should not: Just describe patterns (use **abstract**)

G.4.2 Characterise vs. Define-Unit for Labels

When clustering produces both groups (entities) and per-entity membership labels (feature), both outputs come from the same `define-unit` transform — the labels are an inherent by-product of group creation.

However, when a pre-existing model or grouping is *applied* to assign labels without creating new entities (e.g., assigning documents to their dominant pre-computed topic), the intent is `characterise` — it computes a feature, not new units.

```
# VALID (creates groups + labels):

transform T_cluster :
  intent: define-unit
  output: clusters, cluster_labels

# WRONG (applies existing grouping):

transform T_assign_topics :
  intent: define-unit
  output: topic_assignments # feature, not entities

# CORRECT:

transform T_assign_topics :
  intent: characterise
  output: topic_assignments
```

G.4.3 Artifact Type Appropriateness

Common Mismatches to Check.

specification vs. pattern:

```
# WRONG:

artifact clustering_params : pattern(clusters)
  description: "Parameters for clustering algorithm"

# CORRECT:

artifact clustering_params : specification
  representation form: "parameter settings"
  description: "Distance thresholds and sensitivity
  parameters"
```

knowledge vs. pattern (from assess):

```
# WRONG:
```

```

artifact quality_assessment : pattern(results)

# CORRECT:

artifact quality_assessment : knowledge(results)
  representation form: "quality judgment"

```

pattern vs. knowledge (from abstract):

```

# WRONG:

artifact route_types : knowledge(trajectories)

# CORRECT:

artifact route_types : pattern(trajectories)
  representation form: "categorized route types"

```

feature vs. pattern (from characterise): Symbolic encoding, vectorisation, and other data transformations produce features (descriptors of entities), not patterns (abstractions of regularities).

```

# WRONG:

transform T_encode :
  intent: characterise
  output: symbolic_patterns
artifact symbolic_patterns : pattern(episodes)

# CORRECT:

artifact symbolic_codes : feature(episodes)
  value structure: vector
  value type: categorical

```

entities vs. pattern (concrete vs. abstract): When an operation produces concrete collections of identified objects (e.g., sets of structurally equivalent components), the result is entities, not pattern.

```

# WRONG:

artifact structural_templates : pattern(graph)

# CORRECT:

artifact structural_templates : entities
  internal structure: group/cluster
  embedment: set

```

G.4.4 Artifact References

Verify correct references in parentheses:

- **feature artifacts** must reference entities: `feature(entities_ID)`
- **arrangement artifacts** must reference entities being arranged and have a distinct context entity; the entity being arranged must not be its own context
- **visualisation artifacts** should reference the data artifacts they depict, not other visualisations
- **knowledge artifacts** should reference what they are about
- **pattern artifacts** should reference source data artifacts
- **model artifacts** (non-given) should reference artifacts from which they are built

G.4.5 Given Artifact Dependencies

Artifacts marked `origin: given` must not reference derived artifacts in their parameterisation or feature declarations. A given artifact is exogenous to the workflow and cannot depend on artifacts that do not yet exist at workflow start.

```
# WRONG (given artifact references derived entity):
artifact context_info : feature(episodes)
  origin: given
  # episodes is derived by T_define_episodes

# CORRECT (derive from given + derived inputs):
transform T_characterise_context :
  intent: characterise
  input: episodes, raw_timeseries
  output: context_info

artifact context_info : feature(episodes)
  value structure: vector
  value type: {temporal, spatial, categorical}
```

G.4.6 Loop-Invariant Transforms

Transforms whose inputs never change between loop iterations should be placed **before** the loop, not inside it. This clarifies which computations are iteratively refined and which are performed once.

```

# WRONG (T_arrange inputs are loop-invariant):

loop L1:
  body:
    transform T_cluster : ...
    transform T_arrange :
      input: D_day, D_calendar # never change
      output: A_calendar
    transform T_visualise :
      input: A_calendar, cluster_labels
      ...
end loop L1

# CORRECT:

transform T_arrange :
  input: D_day, D_calendar
  output: A_calendar

loop L1:
  body:
    transform T_cluster : ...
    transform T_visualise :
      input: A_calendar, cluster_labels
      ...
end loop L1

```

The arrangement `A_calendar` is computed once before the loop and remains accessible inside it. Only transforms that depend on iteratively updated artifacts belong in the loop body.

G.5 Phase 3: Paper Alignment

G.5.1 Workflow Structure Matches Paper

Main Phases. Compare workflow phases to the paper’s methodology sections:

- Does the template capture the paper’s main analytical progression?
- Are major steps represented as transforms?
- Are iterative procedures shown as loops?
- Are decision points captured in conditionals?

Procedure: list the paper’s methodology steps, map to workflow transforms, identify missing or extra steps, and check if the order matches.

Iterative Refinement. Papers often describe iteration with phrases such as “iteratively adjusted parameters until...”, “repeated with different settings...”, “user guidance to refine...”. Verify:

- Loops present where paper describes iteration
- Assessment steps included where paper describes quality evaluation
- Parameter adjustment mechanisms captured as specification artifacts with explicit `assign` statements closing the feedback loop
- No loops with implicit parameter changes (comment-only `then` branches)

G.5.2 Terminology Matches Paper's Domain

Artifact Names. Should reflect the paper's terminology while remaining generic:

```
# GOOD: Uses paper's terminology
artifact trajectories : entities

# TOO GENERIC: Loses domain context
artifact data : entities

# TOO SPECIFIC: Dataset-specific
artifact milan_car_trajectories : entities
```

Transform Descriptions. Should align with the paper's described operations without including implementation specifics:

```
# GOOD:
description: "Cluster trajectories by destination
similarity"

# WRONG: Too implementation-specific
description: "Apply k-means clustering"

# WRONG: Not in paper
description: "Cluster trajectories by source similarity"
```

G.5.3 Key Contributions Captured

Identify the paper's main methodological contributions and verify:

- Core methodological innovation is represented
- Not over-detailed (algorithm specifics)

- Not under-detailed (missing key stages)

For example, if a paper contributes “two-stage clustering (spatio-temporal then spatial),” the workflow should show both stages explicitly.

G.6 Phase 4: Abstraction Level Assessment

G.6.1 Too Detailed (Over-Specification)

Warning signs:

Algorithm names in intent or manner:

```
# TOO DETAILED:
manner: "DBSCAN with eps=0.5 and min_samples=5"

# APPROPRIATE:
manner: "density-based clustering"
```

Specific numeric results in descriptions:

```
# TOO DETAILED:
description: "Five clusters representing 23%, 18%..."

# APPROPRIATE:
description: "Clusters grouping trajectories with
             similar characteristics"
```

Tool/library names:

```
# TOO DETAILED:
manner: "D3.js force-directed layout"

# APPROPRIATE:
manner: "force-directed graph layout"
```

Verbose descriptions: Descriptions and manner fields should be concise. Unnecessary repetition and elaboration reduce readability and hinder workflow comparison.

G.6.2 Too Abstract (Under-Specification)

Warning signs:

Missing essential analytical steps:

```

# TOO ABSTRACT:

template: prepare → analyze → visualize → conclude

# APPROPRIATE:

template: characterise → define-unit (cluster) →
         visualise → assess → generate-knowledge

```

Vague descriptions:

```

# TOO ABSTRACT:

description: "Process the data"

# APPROPRIATE:

description: "Compute movement characteristics: speed,
            direction, acceleration"

```

G.6.3 Appropriate Abstraction Level

Characteristics:

- Captures analytical approach without algorithm details
- Domain context present but not dataset-specific
- Method types specified (density-based, hierarchical) but not implementations
- Artifact roles clear without technical formats
- Iteration patterns evident without exact convergence criteria
- Enables comparison across similar workflows
- Descriptions are concise

G.7 Phase 5: Common Issues Catalogue

This section catalogues issues frequently encountered in LLM-generated ATWL workflows, organised by category.

G.7.1 Feature and Visualisation Syntax

Issue: representation form substituted for type-specific fields. The single most common systematic error. LLM agents frequently use a generic representation form field on feature and visualisation artifacts instead of the type-specific mandatory fields.

```

# WRONG (feature):

artifact F1 : feature(D1)
  representation form: "per-record similarity scores"

# CORRECT:

artifact F1 : feature(D1)
  value structure: atomic
  value type: numeric
  description: "Per-record similarity scores"

# WRONG (visualisation):

artifact V1 : visualisation(D1)
  representation form: "interactive scatter plot"

# CORRECT:

artifact V1 : visualisation(D1)
  layout: "2D scatterplot"
  form: "colored points"
  encoding: "position from arrangement; color by
            cluster"

```

Issue: Old-style internal features. Internal features declared with only value type: instead of the unified two-level scheme.

```

# OLD STYLE:

features:
- id: speed

  value type: numeric

# CURRENT:

features:
- id: speed

  value structure: atomic
  value type: numeric

```

G.7.2 Entities and Embedment

Issue: Embedment on single entities. Reference spaces, single aggregation trees, and single networks are single analytical objects. Embedment describes how *multiple* entities relate to each other and should be omitted for single entities.

Issue: Missing embedment on collections. Collections of records, events, or

clusters positioned relative to each other should declare embedment. Spatial events need space in their embedment even if spatial position is also listed as a feature.

Issue: Non-standard internal structure values.

```
# WRONG:
internal structure: network      # use: formation
internal structure: hierarchical # use: formation
internal structure: group        # use: group/cluster
```

G.7.3 Intent–Output Mismatches

Issue: define-unit producing features. Assigning labels (e.g., topic assignments) is computing a feature value, not creating entities. Use `characterise`.

Exception: define-unit producing both entities and label features. When clustering creates new groups *and* assigns membership labels simultaneously, both outputs from a single `define-unit` transform are valid — the labels are an inherent by-product of group creation. The issue arises only when an existing grouping is *applied* to assign labels without creating new entities; in that case, use `characterise`.

```
# VALID (creates groups + labels):
transform T_cluster :
  intent: define-unit
  output: clusters, cluster_labels

# WRONG (applies existing grouping):
transform T_assign_topics :
  intent: define-unit
  output: topic_assignments # feature, not entities

# CORRECT:
transform T_assign_topics :
  intent: characterise
  output: topic_assignments
```

Issue: characterise producing patterns. Encoding, vectorisation, and other data transformations produce features. Use `abstract` for pattern discovery.

Issue: abstract producing entities. When the result is concrete identified objects (sets, groups), use `define-unit`.

G.7.4 Pattern vs. Feature from Characterise

Issue: characterise producing patterns. Data transformations such as symbolic encoding (e.g., SAX), vectorisation, normalisation, and feature extraction produce *de-*

scriptors of entities, not abstractions of regularities. Their outputs should be typed as **feature**, not **pattern**, even when the method name includes the word “pattern.”

```
# WRONG:

transform T_encode :
  intent: characterise
  output: symbolic_patterns

artifact symbolic_patterns : pattern(episodes)

# CORRECT:

transform T_encode :
  intent: characterise
  output: symbolic_codes

artifact symbolic_codes : feature(episodes)
  value structure: vector
  value type: categorical
```

The downstream abstract transform that discovers co-occurrence topics from these codes produces the actual pattern artifacts.

G.7.5 Loop Feedback

Issue: Loops with no assign closing the feedback. When a loop uses Pattern A (explicit conditional), the **then** branch must contain substantive transforms and an **assign** statement. Comment-only branches indicate that the parameter adjustment mechanism is invisible to ATWL. Add a specification artifact and a **generate-knowledge** (or **assess**) transform producing it.

```
# WRONG (then branch is comment-only):

if assessment indicates refinement needed:
  then:
    # Continue with adjusted parameters
  else:
    exit loop L1

# CORRECT:

if assessment indicates refinement needed:
  then:
    transform T_adjust :
      intent: generate-knowledge
      ...
      output: S_params'
    assign: S_params := S_params'
```

```
else:
    exit loop L1
```

Issue: continue loop **construct**. Not a valid ATWL construct. Restructure the conditional so that the substantive work is in one branch and `exit loop` is in the other, letting the loop naturally re-enter after the substantive branch completes.

G.7.6 Knowledge Tangibility

Issue: Mental states as representation form.

```
# WRONG:
representation form: "architectural understanding"
representation form: "comprehensive model assessment"

# CORRECT:
representation form: "statements and diagrams"
representation form: "statements and recommendations"
```

G.7.7 Model Parameterisation

Issue: Derived models without parameterisation.

```
# WRONG:
artifact M1 : model
    description: "Trained classifier"

# CORRECT:
artifact M1 : model(training_data, model_spec)
    description: "Trained classifier"
```

Given models (`origin: given`) may omit parameterisation since training inputs are external.

G.7.8 Description Issues

Issue: Specific results in descriptions.

```
# WRONG:
description: "Finding that 23% of trips are local"

# CORRECT:
```

```
description: "Understanding of trip distance
distributions and route type organization"
```

Issue: Implementation-focused descriptions.

```
# WRONG:
```

```
description: "Apply DBSCAN with eps=0.5"
```

```
# CORRECT:
```

```
description: "Group events with similar spatial
positions using density-based clustering"
```

G.8 Review Report Template

The reviewer should produce a structured report:

```
## ATWL Workflow Review Report
```

```
Paper Title: [Title]
```

```
Workflow ID: [workflow-id]
```

```
---
```

```
### Overall Assessment
```

```
Summary: [2-3 sentence overall assessment]
```

```
Recommendation:
```

```
[ ] Approve as-is
```

```
[ ] Approve with minor revisions
```

```
[ ] Requires revision
```

```
[ ] Reject - major rework needed
```

```
---
```

```
### Strengths
```

```
- [Strength 1]
```

```
- [Strength 2]
```

```
- [Strength 3]
```

```
---
```

```
### Critical Issues (Must Fix)
```

```
#### Issue 1: [Title]
```

```
Location: [Transform/Artifact ID]
```

```

Problem: [Specific description]
Correction: [Corrected ATWL code]
Why: [Explanation]

[Repeat for each critical issue]

---

### Recommended Improvements (Should Fix)

#### Improvement 1: [Title]

Location: [Transform/Artifact ID]
Suggestion: [Specific recommendation]
Why: [Explanation]

[Repeat for recommendations]

---

### Minor Suggestions (Optional)

- [Suggestion 1]
- [Suggestion 2]

---

### Alignment with Paper

Captured from paper:
- [pass/fail] Main methodology phases
- [pass/fail] Iterative refinement procedures
- [pass/fail] Key methodological contributions
- [pass/fail] Domain terminology

Missing or incorrect:
- [List any missing or misrepresented elements]

---

### Conclusion

[Final paragraph summarizing assessment and
next steps]

```

G.9 Quick Reference Checklist

G.9.1 Red Flags (Critical Issues)

- representation form used instead of value structure on feature artifacts

- representation form used instead of layout/form/encoding on visualisation artifacts
- assess producing pattern artifacts
- Parameters/settings as pattern artifacts
- origin: given on transform outputs
- Missing loop(...) when paper describes iterative refinement
- Intent types clearly mismatched to purposes
- Comment-only conditional branches that hide parameter adjustment (should contain substantive transforms with assign)
- continue loop construct
- applies to: field on specifications
- Orphan artifacts (no provenance)
- Dead-end artifacts in loop conditionals
- Non-standard internal structure values
- Embedment on single entities
- Knowledge representation form describing mental states
- Trailing colon on workflow declaration
- Given artifacts referencing derived artifacts in parameterisation
- characterise intent producing pattern artifacts
- Loop-invariant transforms inside loop bodies
- // comment syntax (use #)

G.9.2 Yellow Flags (Likely Issues)

- abstract used for quality evaluation
- characterise used for strategic decisions
- define-unit producing features instead of entities
- abstract intent producing entities instead of pattern (verify whether the result is concrete objects or abstracted regularities)

- Missing `embedding` on entity collections
- Missing `value type` on feature artifacts (optional but improves clarity)
- Missing `description` on transforms
- Template not matching body structure
- Model artifacts without parameterisation
- Specification artifacts missing `representation form`:
- Arrangement context referencing the same entity being arranged
- Post-loop terminal action duplicated across multiple exit branches (consider moving after `end loop`)
- Comment-only non-exit branch in a refinement loop (verify that no parameter adjustment is being hidden)
- Verbose descriptions without added information
- Internal features with no downstream consumers
- Old-style `value type`: without `value structure`: on internal features
- Transforms with inputs that never change between loop iterations (candidates for moving before the loop)

G.9.3 Green Flags (Quality Indicators)

- Clear template matching body structure
- `assess` produces `knowledge`
- `abstract` produces `pattern`
- Feature artifacts have `value structure` (and optionally `value type`)
- Visualisation artifacts have `layout/form/encoding`
- Embedment present on collections, absent on single entities
- Generic but domain-appropriate terminology
- Loop conditionals gate substantive transforms with `assign` closing feedback
- All non-given artifacts traceable to a producing transform
- Post-loop terminal actions clearly placed (consistently after `end loop` or within exit branches)

- Concise, informative descriptions
- Knowledge representation forms describe tangible outputs
- Consistent transform field ordering

G.10 Final Guidance

Reviews should be:

Specific Point to exact locations with artifact/transform IDs

Constructive Provide corrected code examples

Educational Explain why changes are needed

Prioritised Critical issues first, then recommendations, then suggestions

Balanced Acknowledge strengths as well as issues

Key principles to remember:

- ATWL represents analytical logic, not implementation details
- Features use `value structure + value type`; visualisations use `layout/form/encoding`
- `assess` → `knowledge` (and optionally `specification`); `abstract` → `pattern`
- `generate-knowledge` may produce both `knowledge` and `specification`
- Specifications control operations; patterns describe data regularities
- Embedment characterises collections, not single entities
- Knowledge artifacts represent tangible, recorded outputs
- Every non-given artifact needs a producer; every non-terminal artifact needs a consumer
- Loop feedback must be closed with explicit `assign` statements
- Given artifacts must not reference derived artifacts
- Loop-invariant transforms belong before the loop
- Abstraction level should enable workflow comparison

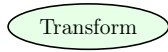
The goal is helping create high-quality ATWL representations that accurately capture research workflows at appropriate abstraction for comparison and reuse.

Diagram Legend

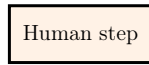
Node types



Data artifact: dataset, feature, arrangement, visualisation, pattern, or knowledge



Machine-driven processing or computation step



Human-driven or hybrid step: assessment, interpretation, parameter adjustment



Parameter settings or configuration guiding a transform



Loop exit condition: determines whether to iterate (refine) or proceed

Edge types



Primary data or control flow
Secondary or cross-boundary data flow



(e.g., loop-internal artifact feedback loop step)



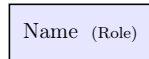
dated artifact replaces original for next iteration

Grouping

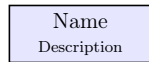


Iterative refinement loop: enclosed steps repeat-until exit-condition is satisfied

Node labels



Single-line format: artifact or transform name followed by ATWL role in parentheses



Two-line format: used when horizontal space is constrained



Decision branch labels: *yes* continues the loop; *no* exits to subsequent steps

Figure 1: Legend of visual encodings used in the ATWL workflow diagrams.

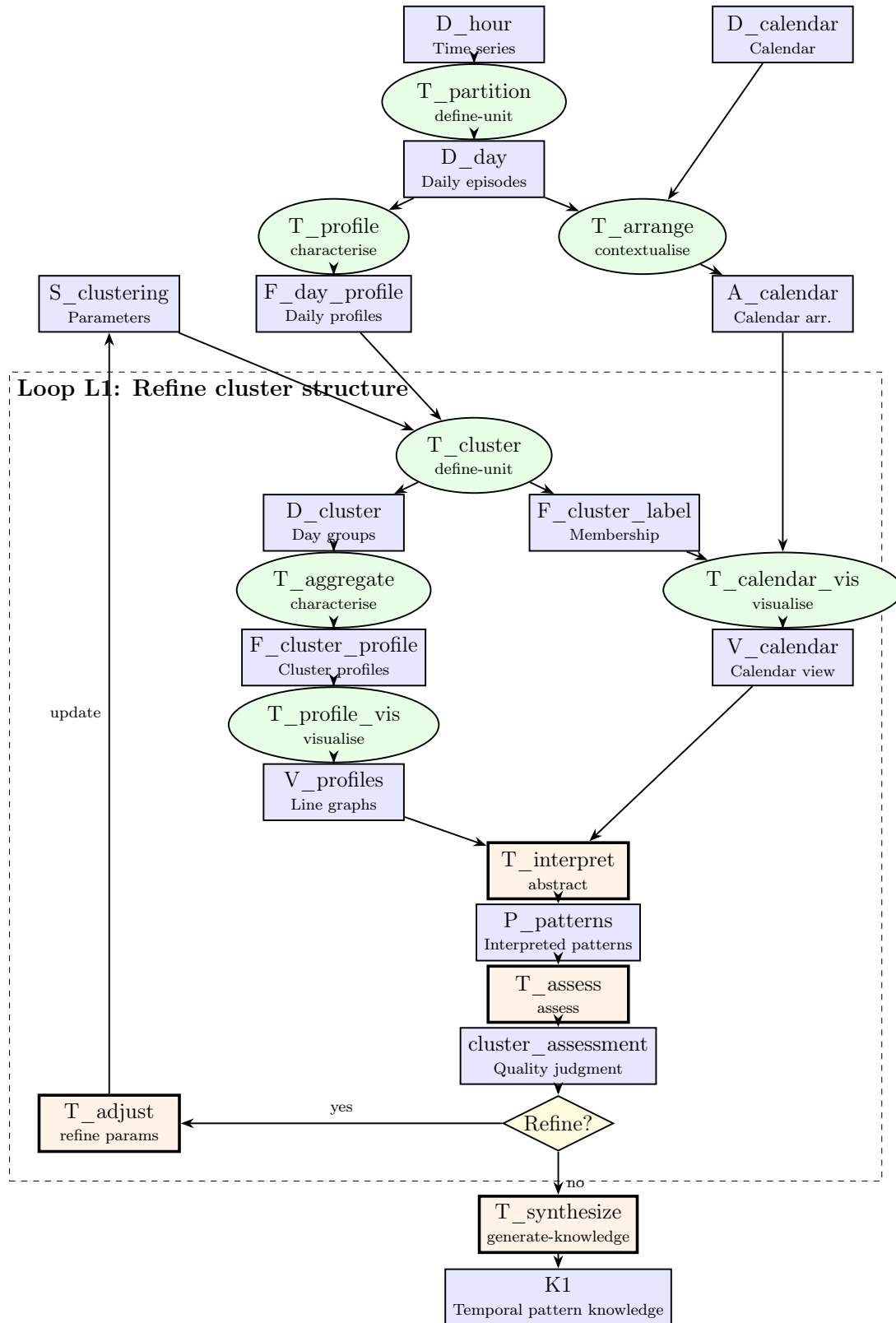


Figure 2: Diagrammatic representation of the Cluster-Calendar workflow

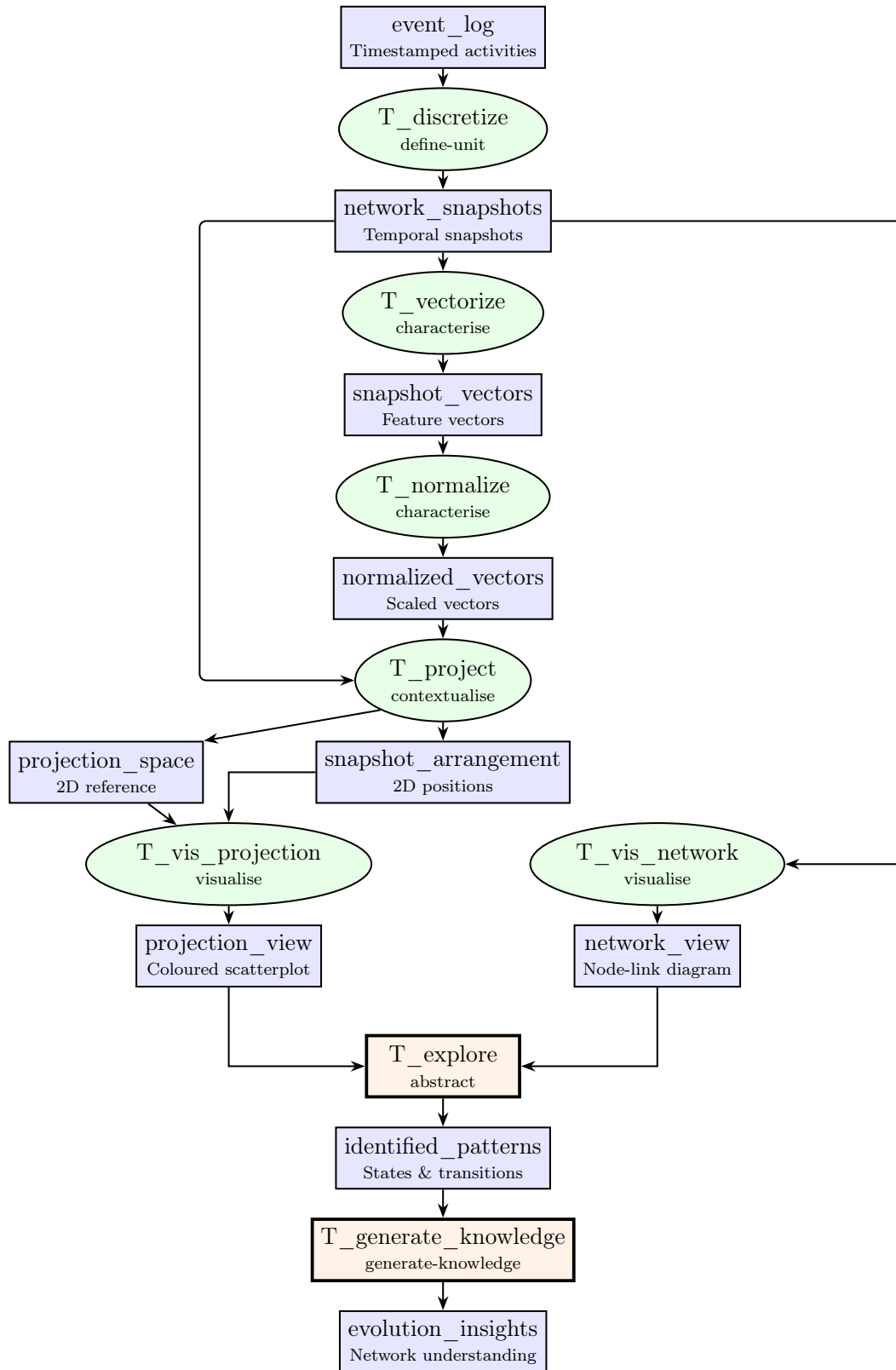


Figure 3: Diagrammatic representation of the Dynamic Network Exploration workflow

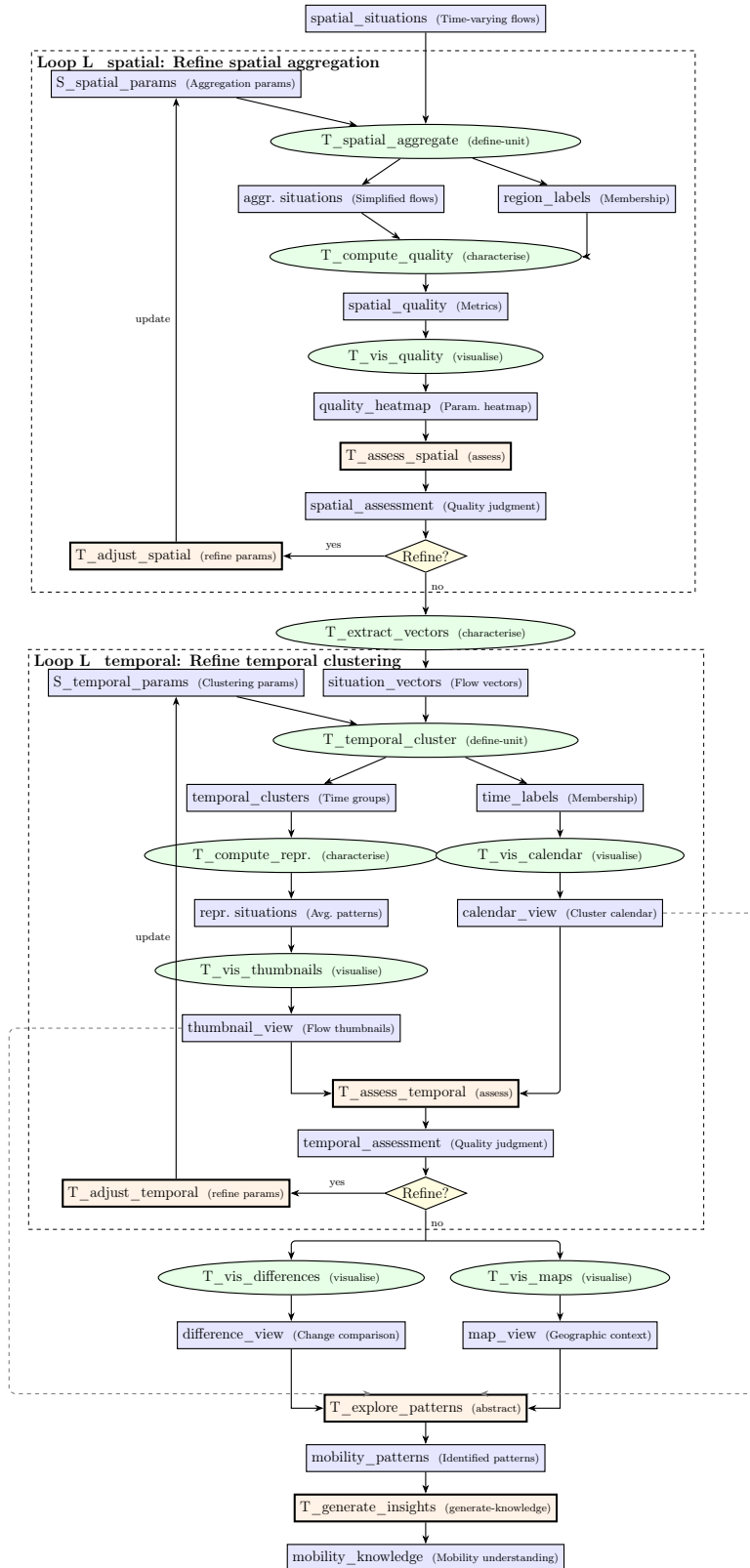


Figure 4: Diagrammatic representation of the MobilityGraphs workflow

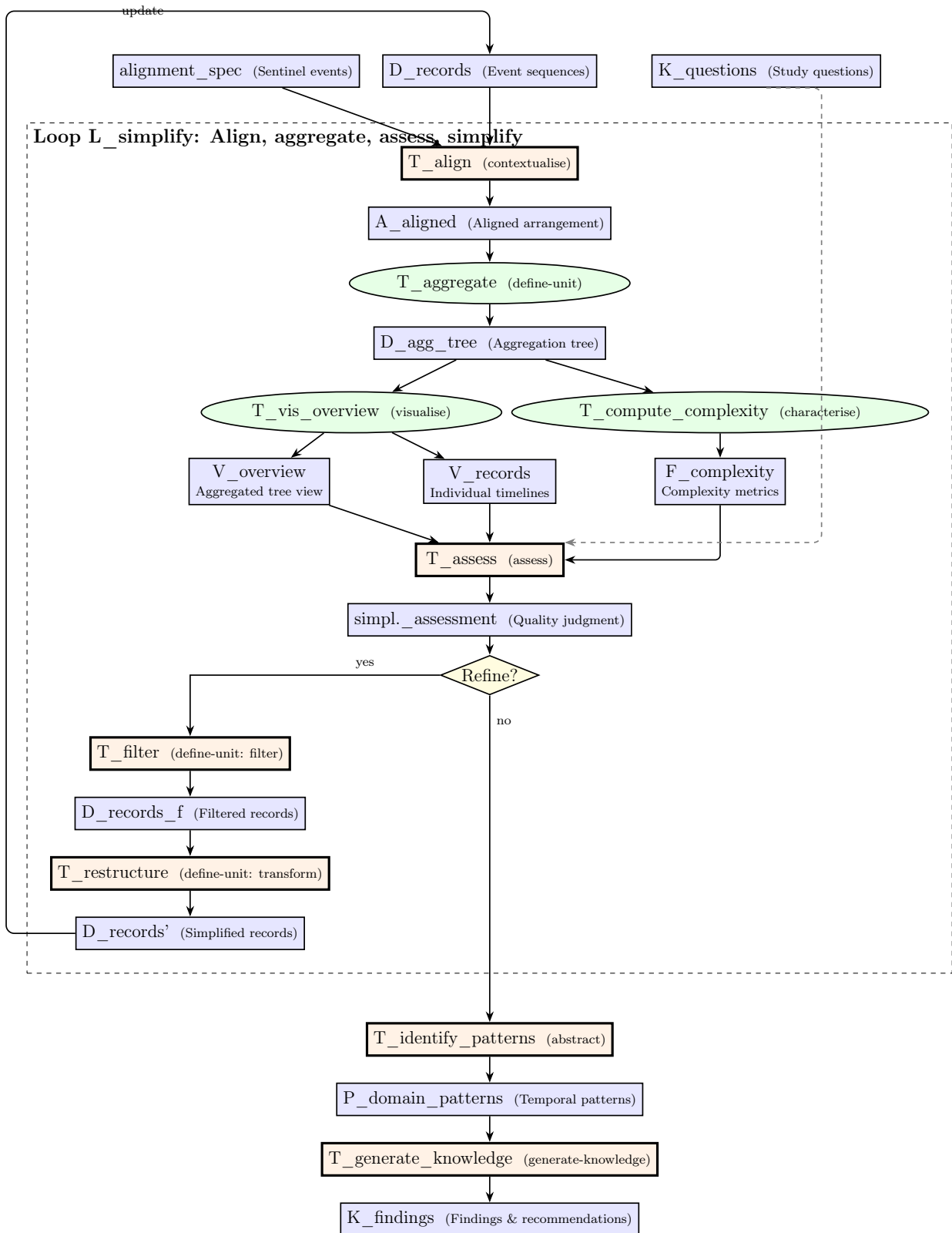


Figure 5: Diagrammatic representation of the EventFlow workflow

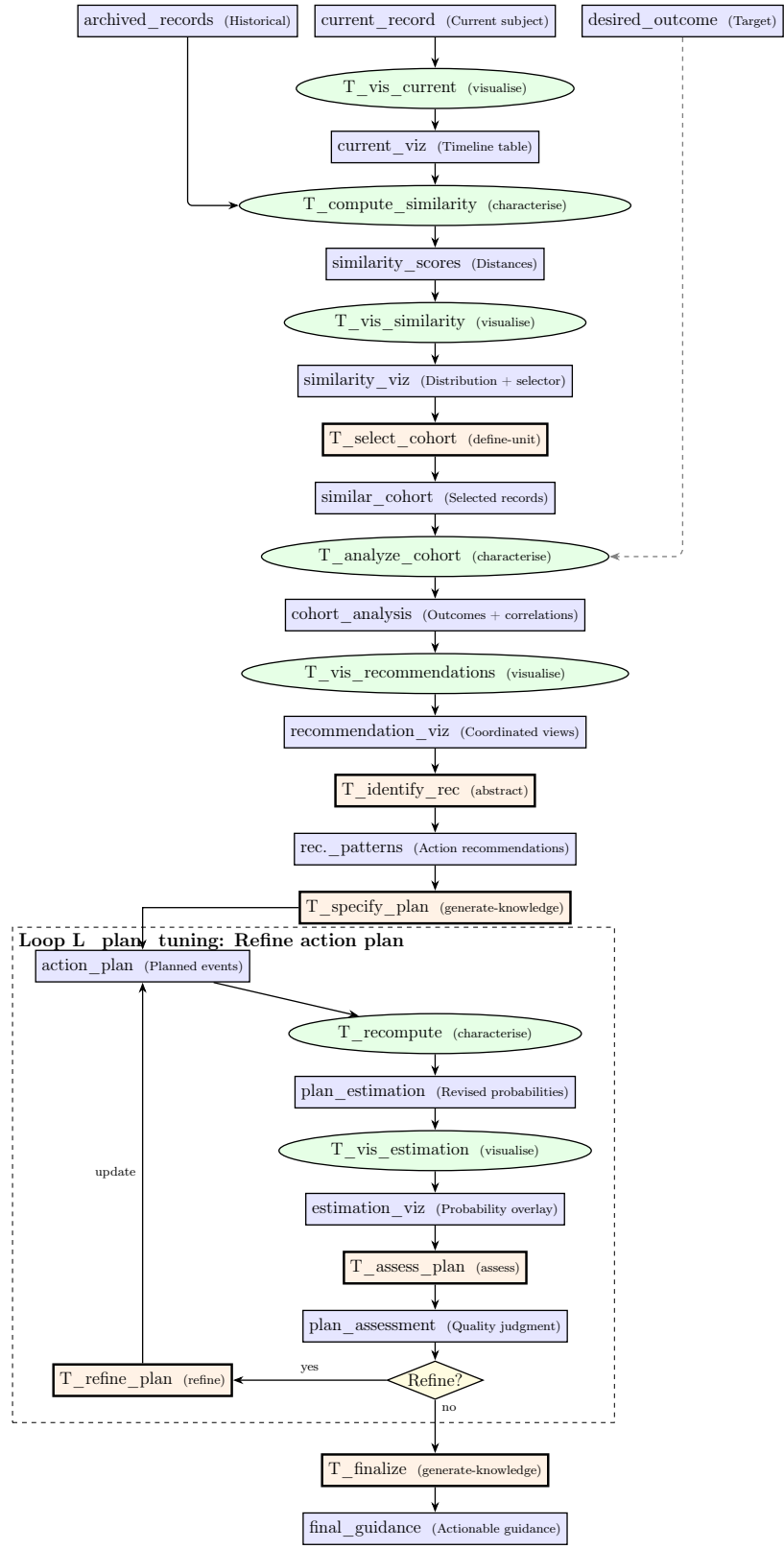


Figure 6: Diagrammatic representation of the EventAction workflow

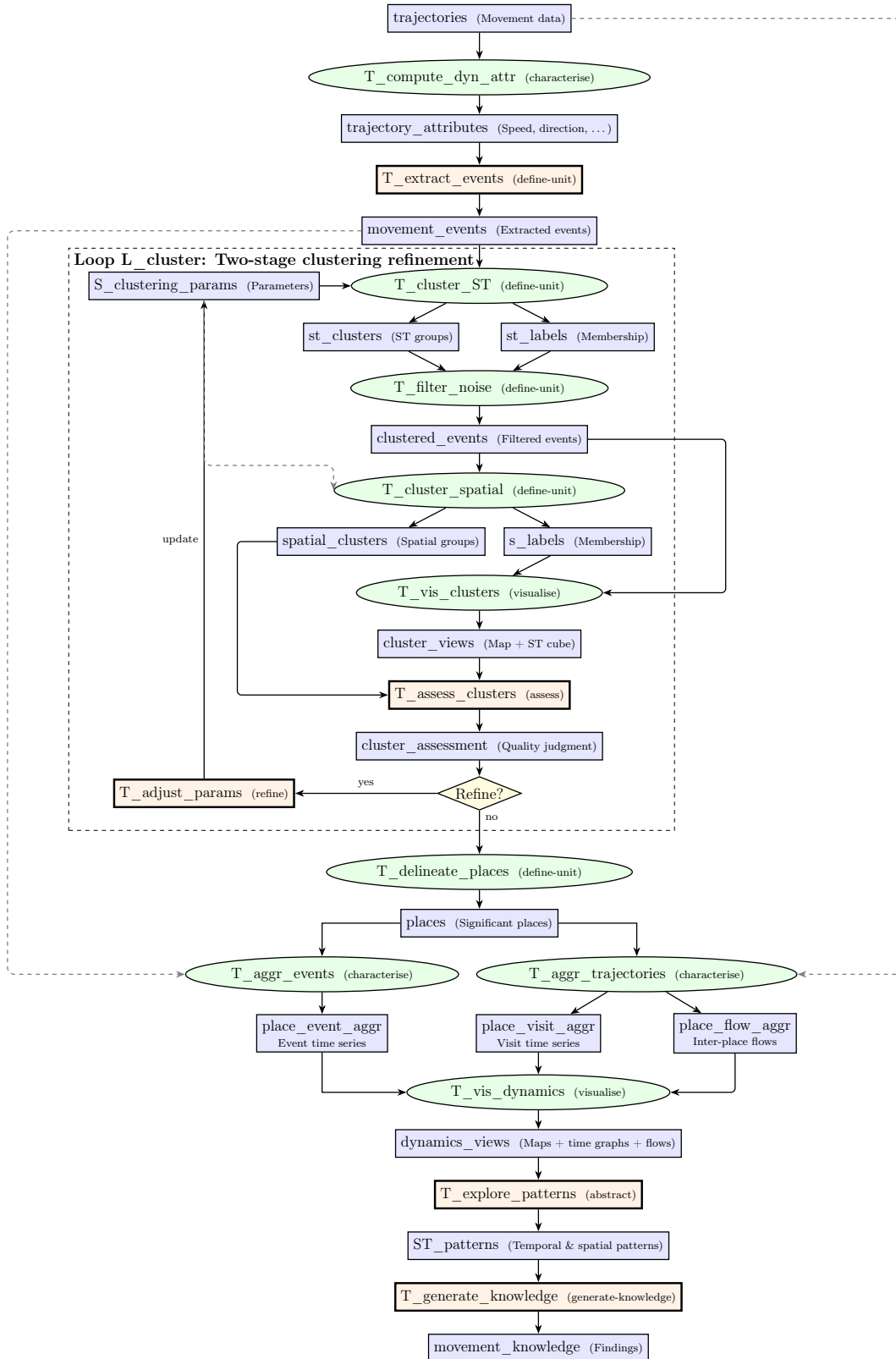


Figure 7: Diagrammatic representation of the Events-to-Places workflow

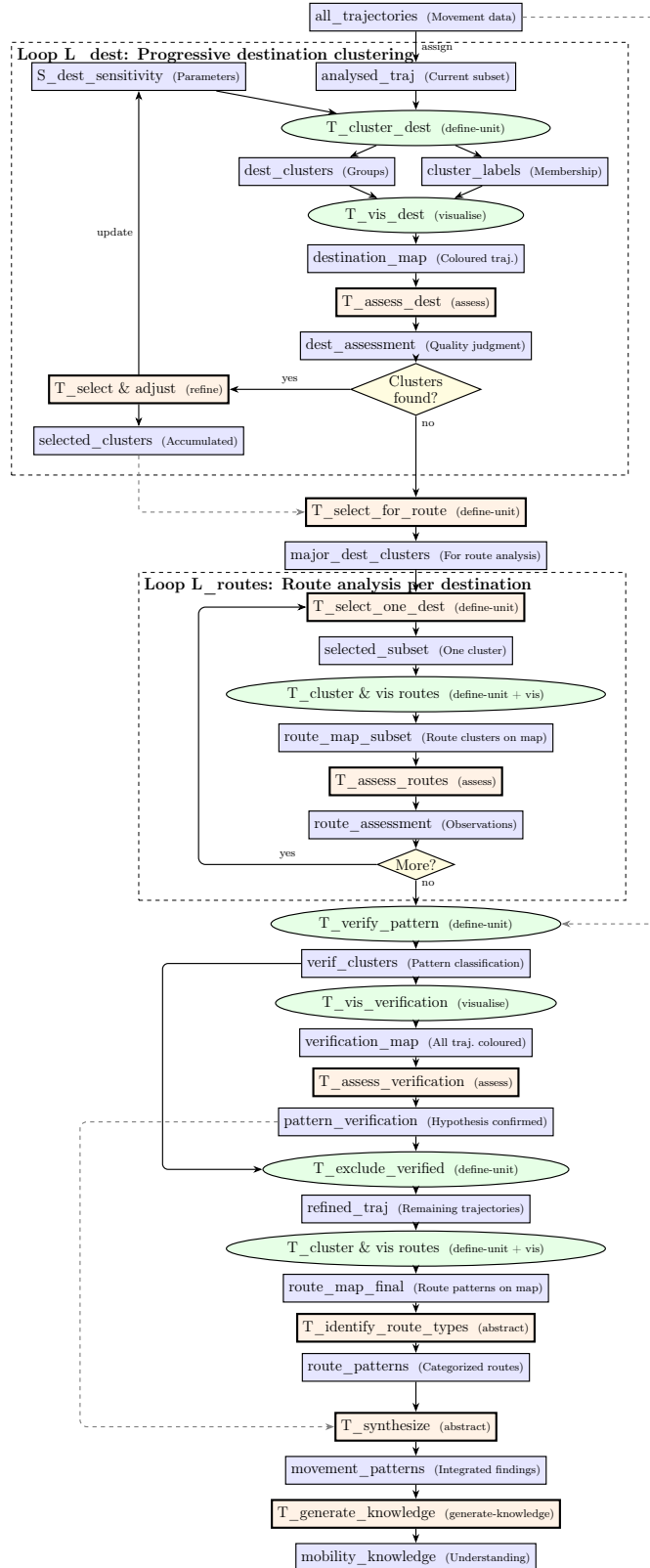


Figure 8: Diagrammatic representation of the Progressive Clustering workflow

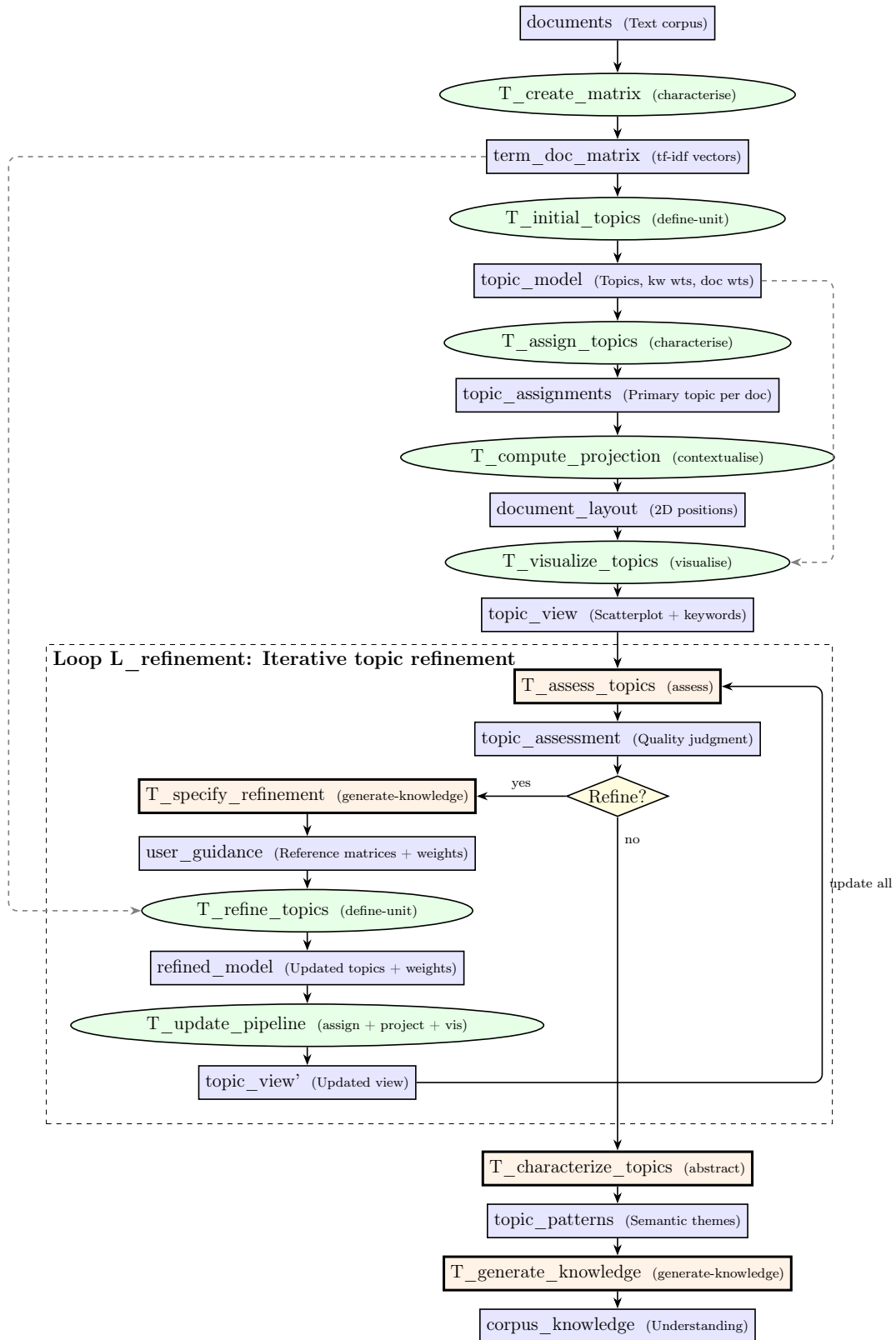


Figure 9: Diagrammatic representation of the Human-Steered Topic Modelling workflow

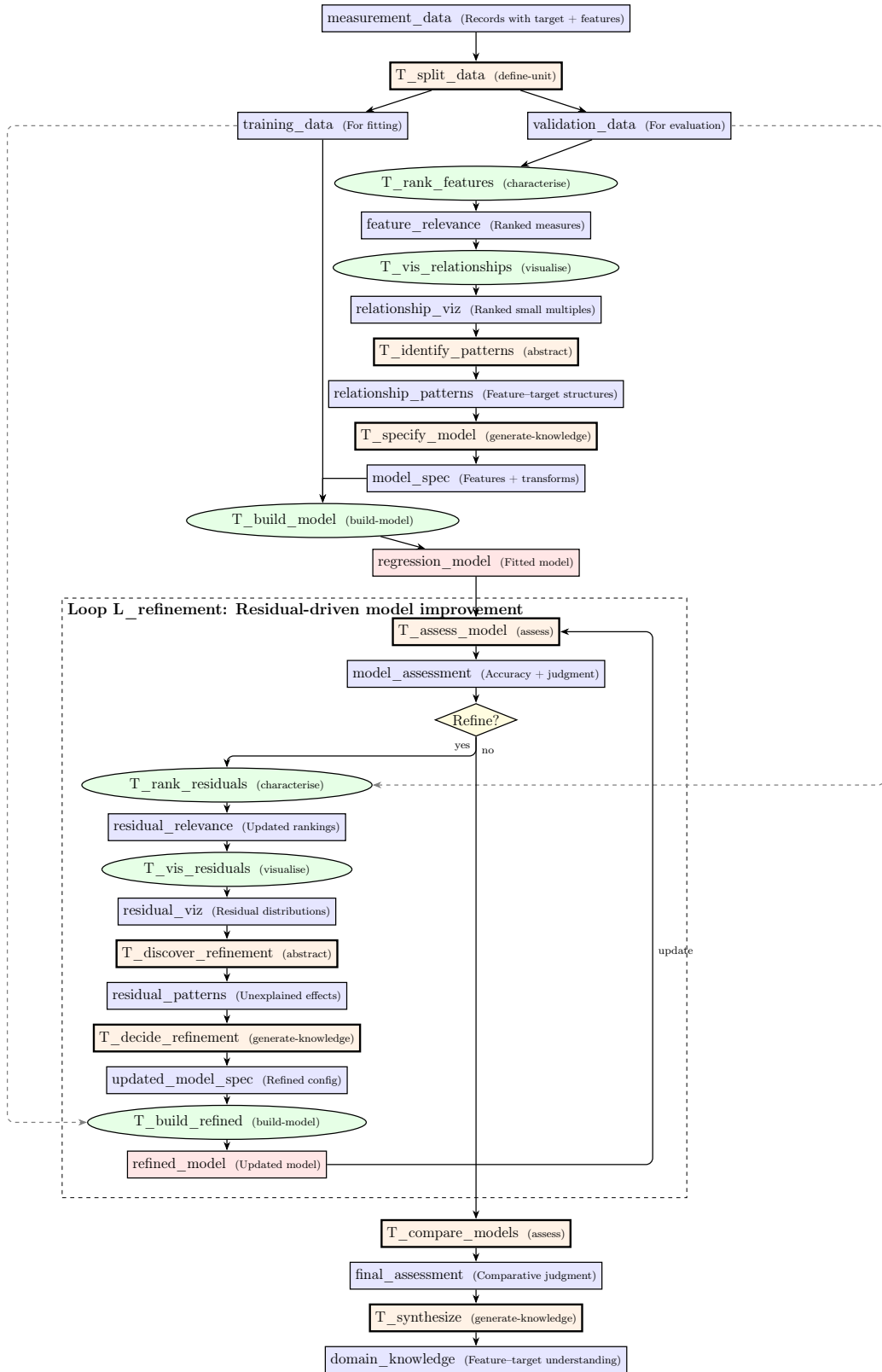


Figure 11: Diagrammatic representation of the Partition-Based Regression Modelling workflow

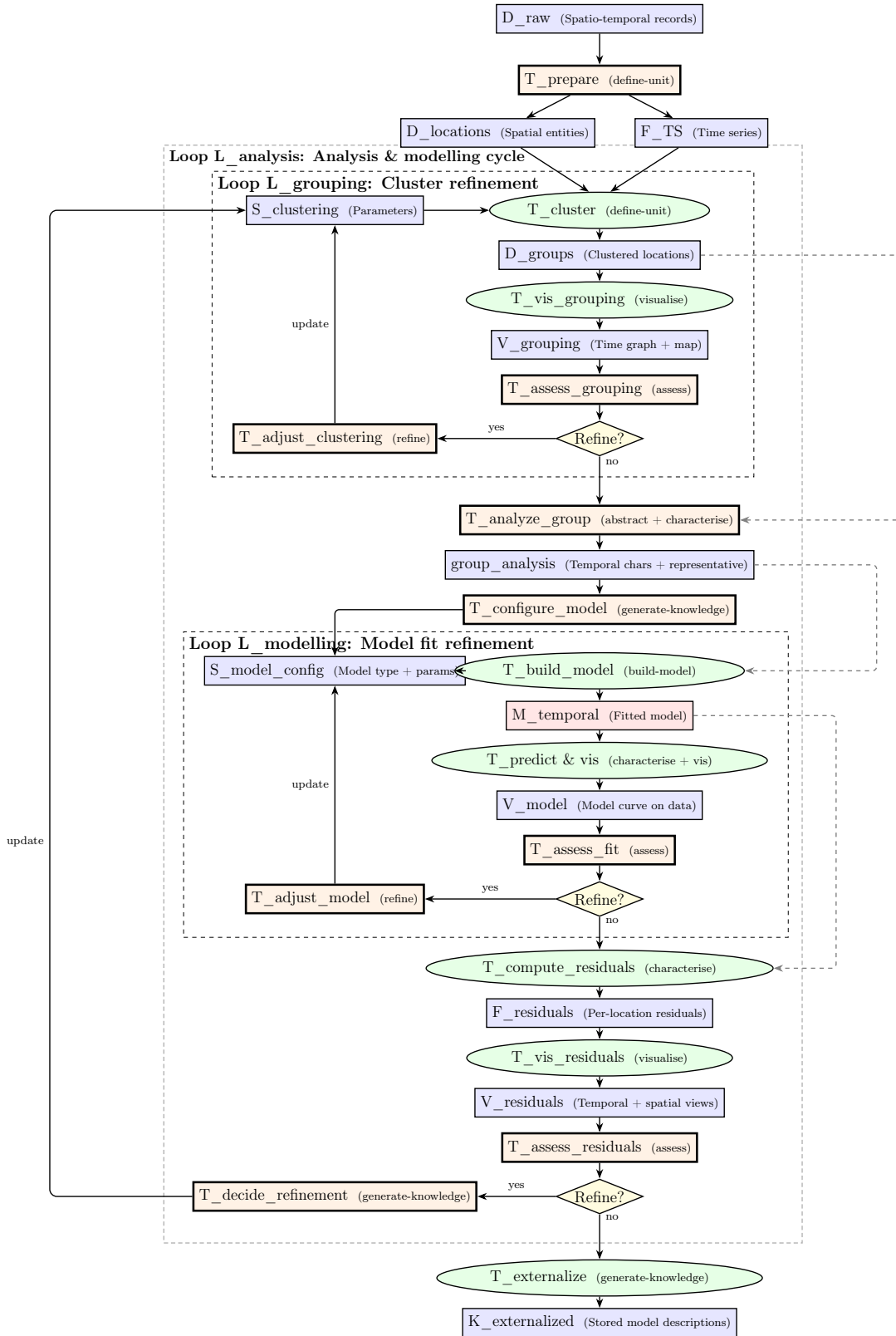


Figure 12: Diagrammatic representation of the Spatio-Temporal Modelling workflow

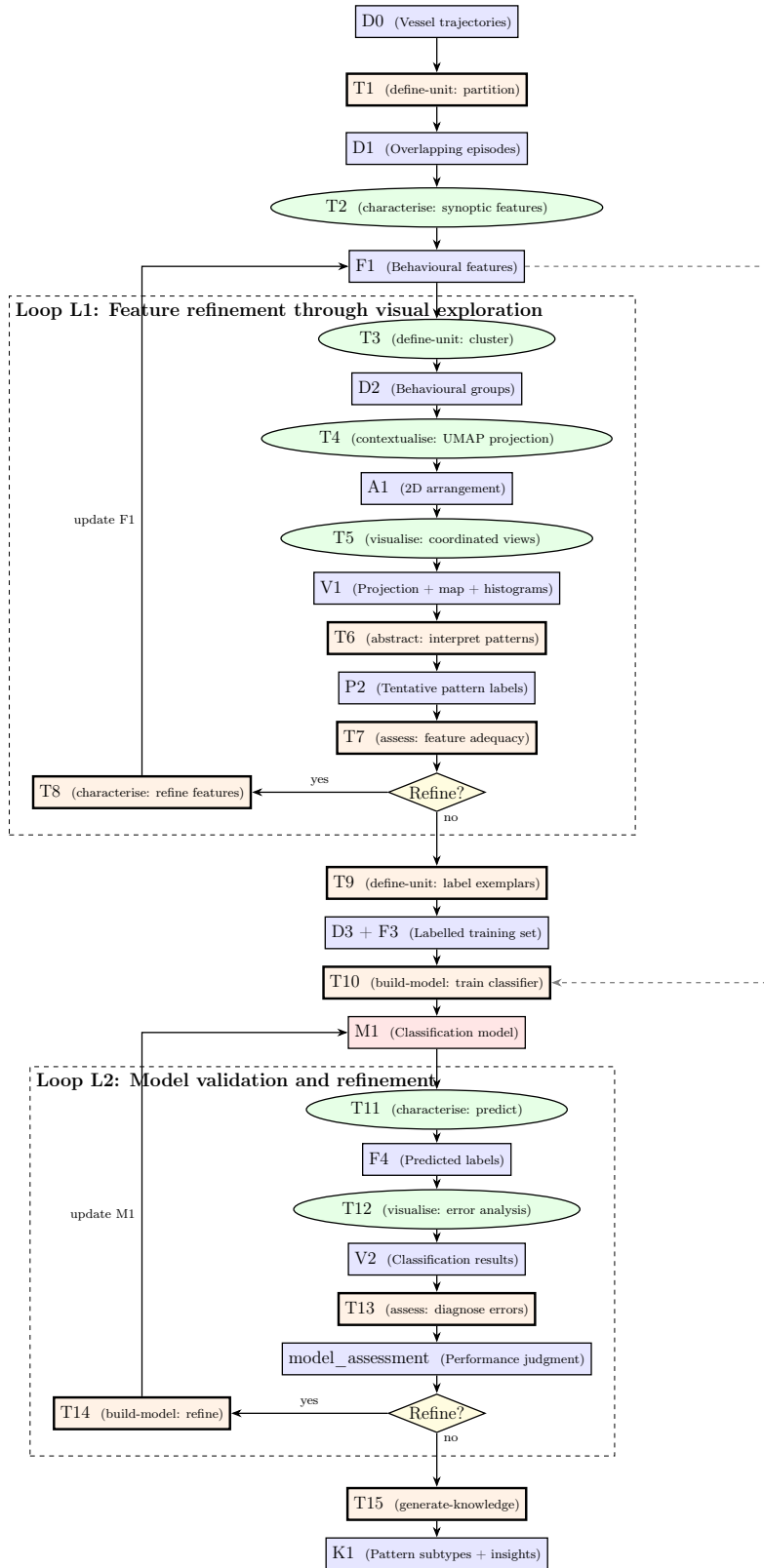


Figure 13: Diagrammatic representation of the Data Engineering for Movement Pattern Classification

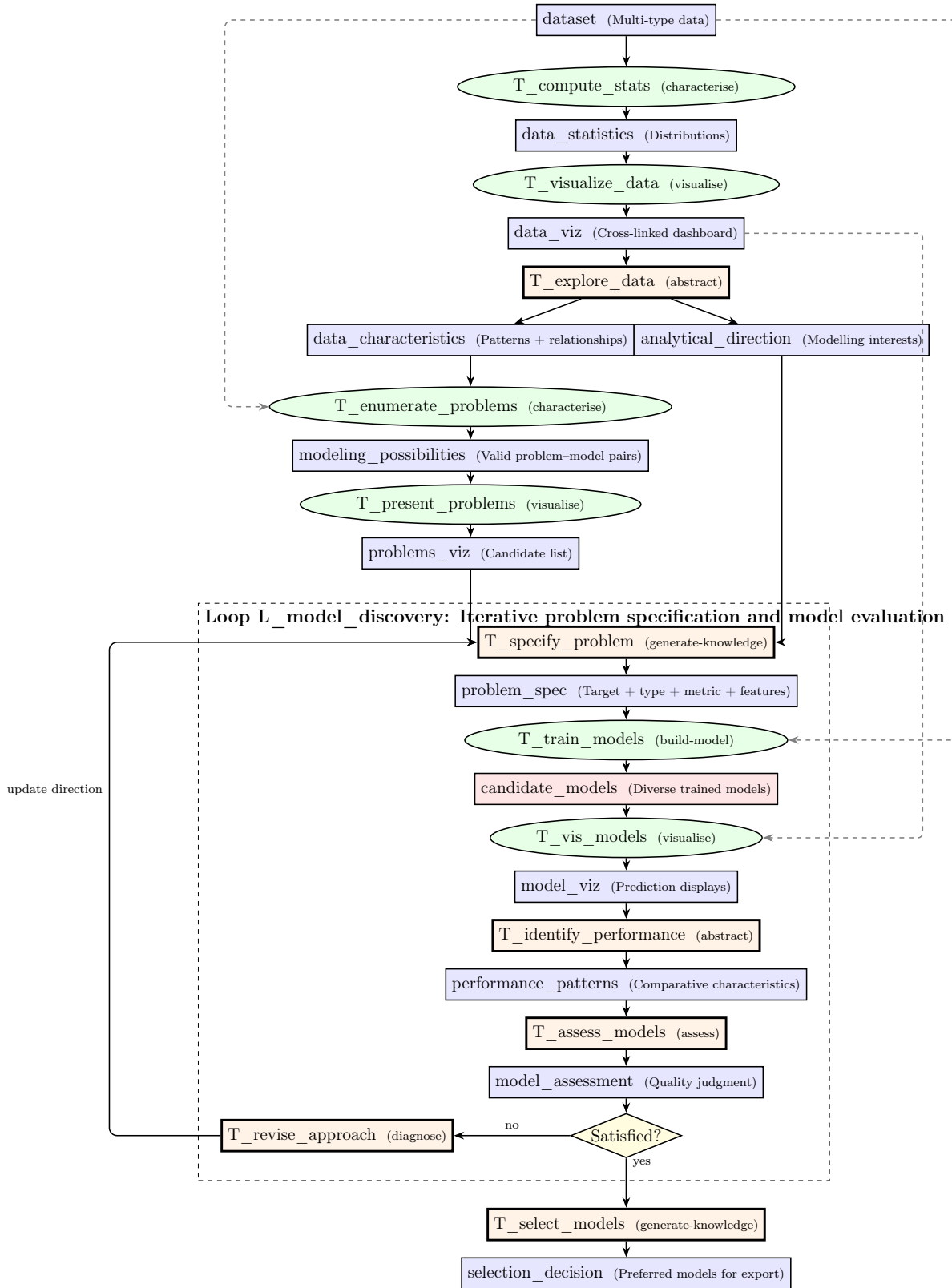


Figure 14: Diagrammatic representation of the Exploratory Model Analysis workflow

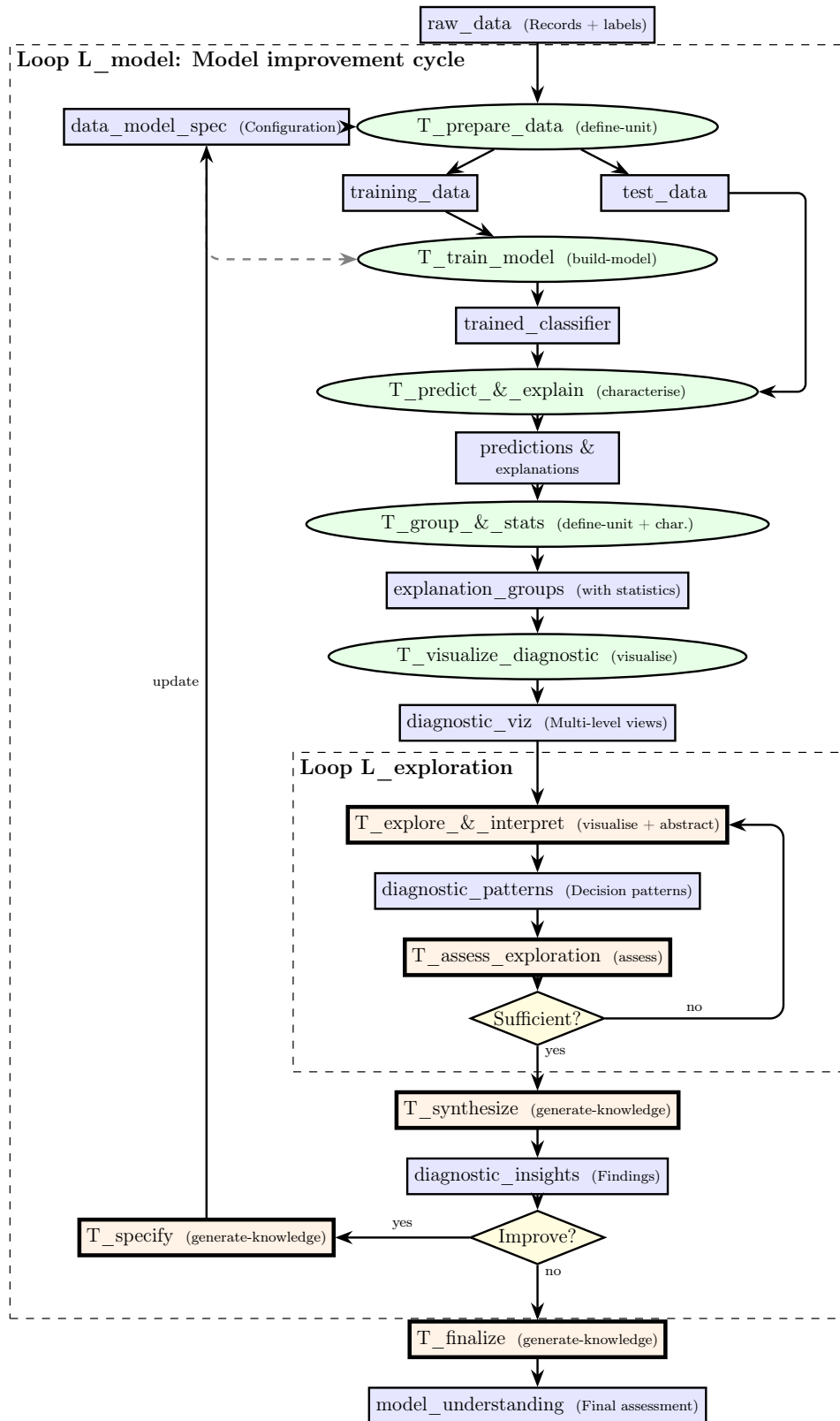


Figure 15: Diagrammatic representation of the Binary Classifier Diagnostics workflow

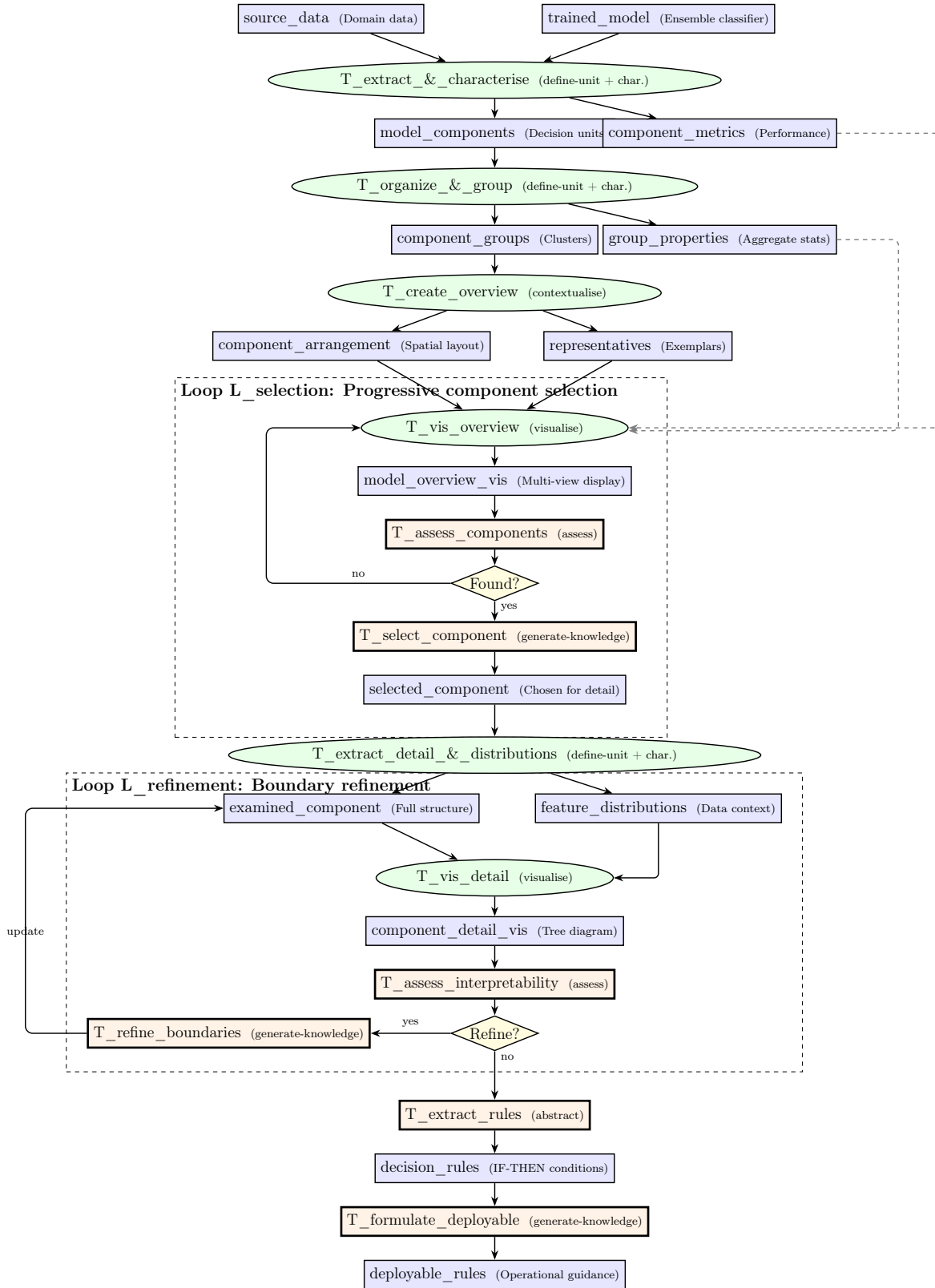


Figure 16: Diagrammatic representation of the Ensemble Classifier Exploration workflow

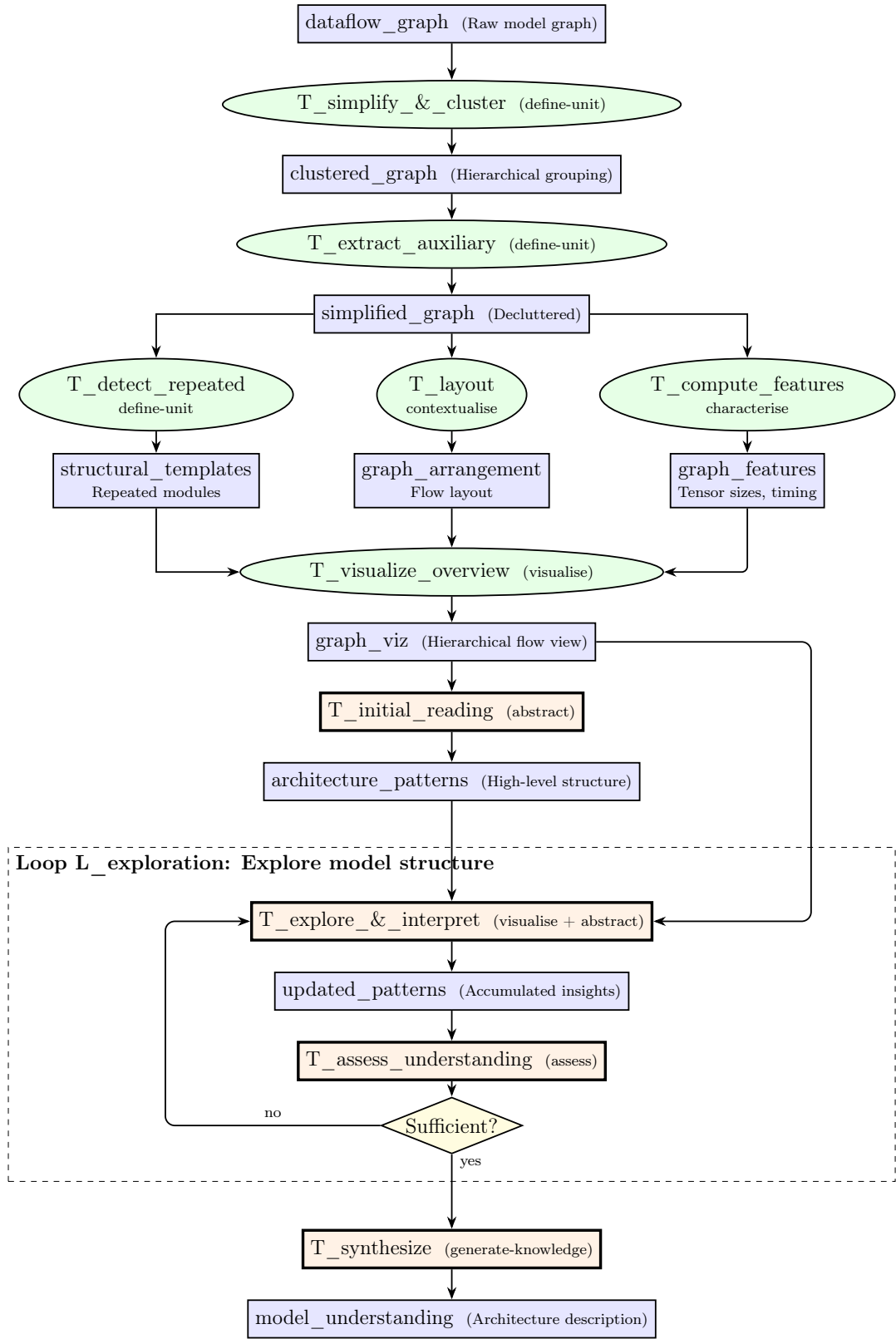


Figure 17: Diagrammatic representation of the Deep Learning Models Exploration in TensorFlow

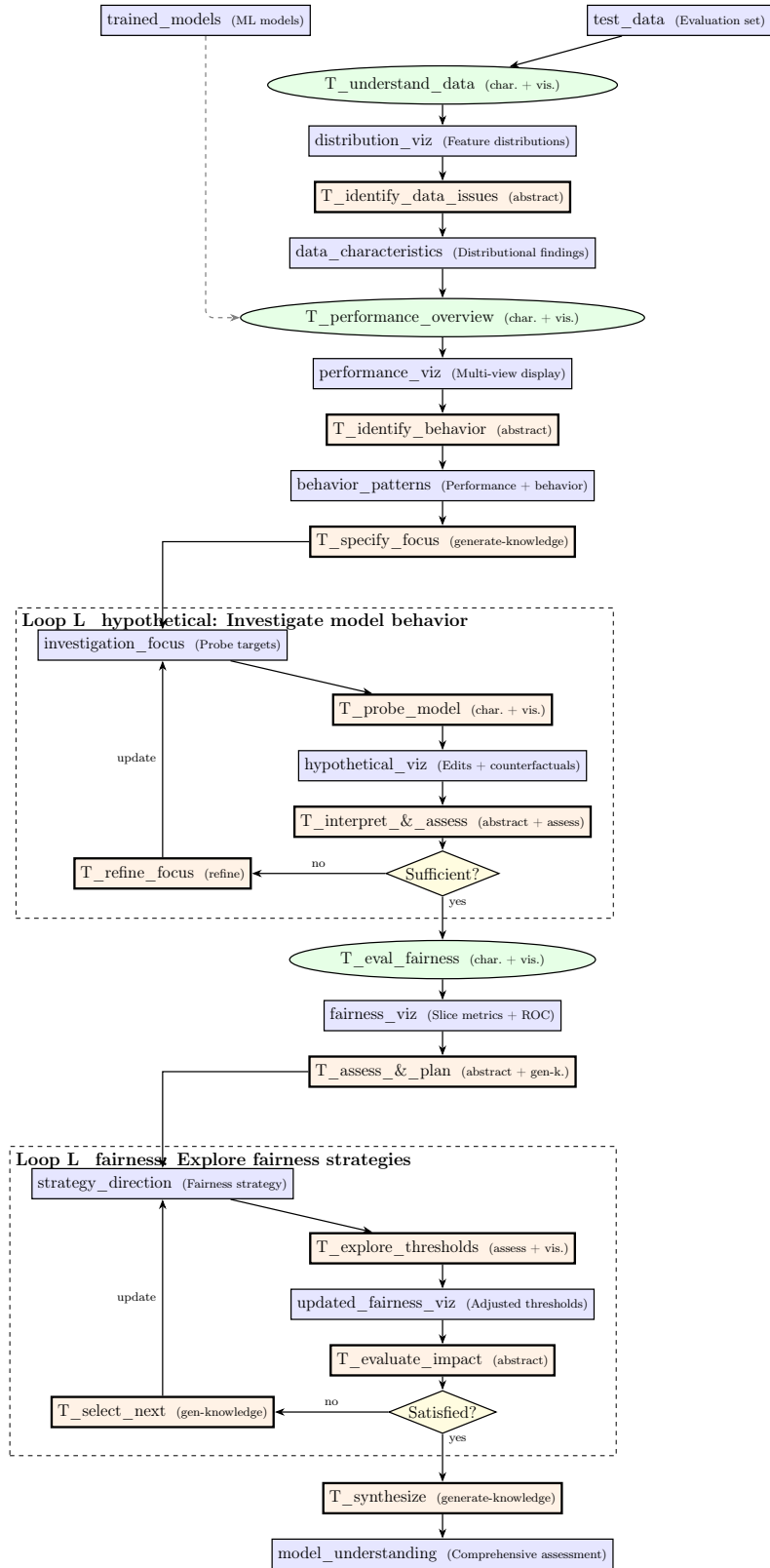


Figure 18: Diagrammatic representation of the process of What-If Probing of ML Models