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**.

---

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	`similarity_scores`, `cohort_analysis`, `plan_estimation`
1.10	`feature_relevance`, `updated_feature_relevance`
1.12	`data_statistics`, `modeling_possibilities`
1.13	`prediction_scores`, `instance_explanations`, `group_statistics`
After 1.15	`graph_features`
1.16	`feature_statistics`, `performance_metrics`, `hypothetical_results`, `fairness_metrics`

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.

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`

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.

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`

Workflow	Affected artifacts
1.15	`dataflow_graph`, `clustered_graph`, `simplified_graph`, `layout_space`
1.16	`test_data`

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	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`

Workflow	Transforms producing specifications
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`

> **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.

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)

> 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.

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

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	Loop
1.13	`L_exploration` (inner diagnostic exploration loop)
1.15	`L_exploration`

---

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 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/redefining entities.

1.9 — Progressive Abstraction of Multivariate Temporal Data

- **Intent–output type mismatch (`characterise` → `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.

---

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	`generate-knowledge` → specification	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

Issue ID	Category	Workflows affected
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).