ATWL: A Formal Language for Representing, Comparing, and Reusing Visual Analytics Workflows

Natalia Andrienko, Gennady Andrienko, Jürgen Bernard, and Michael Sedlmair

Abstract—Visual analytics (VA) workflows are inherently complex, involving data transformation, feature engineering, visual representation, and human interpretation. While these processes are central to research and practice, they are typically described in unstructured prose, hindering systematic comparison, reuse of proven analytical strategies, and training of novice practitioners. We present Artifact–Transform Workflow Language (ATWL), a domain-agnostic, declarative language designed to formally represent VA workflows by capturing their structure and underlying analytical intent. ATWL is built upon a modular ontology of eight artifact types (entities, features, arrangements, visualisations, patterns, models, knowledge, and specifications) and transforms characterised by standardised intents (e.g., define-unit, characterise, contextualise, abstract). To demonstrate that the formalisation effort can be moderate and may not impede adoption, we show that workflows can be extracted from research papers through supervised interaction with LLM agents, reducing the human role to review and refinement. Using this process, we constructed a library of seventeen ATWL workflows extracted from published VA papers. Cross-workflow analysis within this library reveals structural regularities—a recurrent meta-structure, recurring structural motifs, reusable methodological building blocks, diverse iterative strategies, and cross-domain equivalences—that remain invisible when comparing the original prose descriptions. This analysis illustrates the analytical affordances of formal representation. We further evaluate practical utility through a controlled experiment in which the same LLM addressed two analytical problems with the library supplied either as the original research papers or as ATWL representations. Both forms enabled useful recommendations, but the formal representation systematically added explicit iteration structure, typed data flow, fragment-level adaptation provenance, and compactness that supports scaling beyond what prose libraries can fit in an LLM's context. By providing a common vocabulary for analytical structure and intent, ATWL enables a transition from narrative descriptions to formally represented, comparable, and reusable analytical knowledge.

Index Terms—Visual analytics, workflow, conceptual model, formal language.

1 Introduction

Visual analytics (VA) is characterised by the tight integration of automated computational methods and human cognitive capabilities to derive insights from complex datasets. Joint activities of computers and humans form sophisticated analytical workflows including data transformations, feature computation, visual representations, interpretation, and iterative refinements. These workflows are predominantly communicated through unstructured natural language in research papers and technical reports, such as design studies [1].

This reliance on prose creates a significant *communication gap* in VA research. First, the lack of formalisation hinders systematic comparison of methodologies across different application domains. For instance, it is difficult to determine whether a trajectory clustering approach and a topic modelling pipeline share a common underlying logical structure. Second, the absence of a standardised representation limits reproducibility, as implementation details are often conflated with higher-level analytical intent. Third, for novice practitioners and domain researchers, the barrier to designing effective workflows remains high, as there is no formal repository of proven analytical strategies to consult or adapt. More broadly, the absence of a shared formal language prevents the emergence of a systematic, cumulative science of human–computer analytical processes in which workflows can be decomposed, compared, composed, and optimised on principled grounds rather than through ad hoc narrative description.

To address these challenges, we present *Artifact–Transform Workflow Language* (ATWL), a domain-agnostic, declarative language designed to formally represent the structure and intent of VA processes. ATWL is built upon a modular ontology of structural artifacts (such as entities, features, and arrangements) and transforms defined by standardised analytical intents (e.g., `define-unit`, `characterise`, `contextualise`, `abstract`). The language was developed through multiple iterations of conceptual design informed by progressive formalisation of existing VA workflows across diverse application areas. Through this process, we identified a core set of concepts and categories that are sufficiently abstract to be domain-independent, yet sufficiently expressive to capture the structural variety of human–machine analytical processes.

Section 2 outlines research directions that adoption of

- *N. Andrienko and G. Andrienko are with Fraunhofer Institute IAIS, the Lamarr Institute for Machine Learning and Artificial Intelligence, Sankt Augustin, Germany, and City St George's, University of London, UK. E-mail: {natalia, gennady}.andrienko@iais.fraunhofer.de*
- *J. Bernard is with the University of Zurich, Switzerland. E-mail: juergen.bernard@uzh.ch*
- *M. Sedlmair is with the University of Stuttgart, Germany. E-mail: michael.sedlmair@visus.uni-stuttgart.de*

Manuscript received xx xxx. 202x; accepted xx xxx. 202x. Digital Object Identifier: xx.xxxx/TVCG.202x.xxxxxxx

a formal workflow language would open, from compositional workflow design to formal analysis of analytical strategies and human–machine collaboration structures. The present paper provides a foundation: the language itself, a demonstration that formalisation is practical, and initial illustrations of what formal representation makes possible.

The specific contributions of this work are as follows.

1) **A formal language for representing VA workflows (conceptual contribution).** We introduce ATWL, to our knowledge the first formal representation of VA workflows that combines (a) domain-agnostic scope, (b) analysis-time focus, (c) machine-readable syntax, and (d) restriction to observable, externalised analytical products. Its design is based on four substantive choices that distinguish it from prior frameworks: a typed artifact ontology; classification of transforms by analytical intent rather than computational method; treatment of explicitly externalised human knowledge and specifications as first-class artifacts; and domain independence. These decisions enable structural comparison of workflows that use different domain vocabulary and computational techniques.
2) **A library of seventeen formally represented workflows spanning six application domains (resource contribution).** Using ATWL, we formalised workflows from published VA research covering temporal pattern analysis, movement analytics, event sequence simplification, topic modelling, statistical model building, and machine learning model diagnostics. The library exercises all constructs defined in the language. It is publicly available as an open resource for the community.
3) **A human–LLM collaborative process for workflow formalisation (methodological contribution).** To demonstrate that the adoption barrier need not be prohibitive, we developed a collaborative process in which LLM agents extract ATWL representations from research papers under human supervisory oversight. Validation with fresh LLM instances confirmed that the process produces correct extractions reproducibly. We release reusable instruction sets to enable other researchers to formalise their own workflows.
4) **A demonstration that formal representation enables systematic cross-workflow analysis (analytical contribution).** We demonstrate that formally represented workflows become suitable for systematic structural comparison, revealing regularities invisible in prose descriptions, including a recurrent meta-structure, recurring motifs, reusable building blocks, diverse iterative strategies, and cross-domain structural equivalences. We present these as empirically grounded hypotheses whose generality can be tested as the library grows.
5) **A comparative evaluation of formal vs. prose libraries for LLM-based workflow recommendation (practical contribution).** In a controlled experiment, the same LLM addressed two analytical problems with the library supplied in two forms: as the original research papers (PDFs) and as ATWL representations. Both forms produced usable recommendations, so we do not claim that formalisation is a precondition for LLM-based design support. The formal representation does, however, systematically deliver four things that the prose baseline delivers only partially: explicit iteration structure, typed data flow that makes the composition of fragments structurally checkable, fragment-level adaptation provenance, and compactness that supports scaling to libraries larger than the LLM's context budget can accommodate as prose. The two formats are complementary, and we argue that using ATWL recommendations as an index into the source papers combines their respective strengths.

By providing a common vocabulary for analytical structure and intent, ATWL enables a transition from narrative descriptions to formally represented, comparable, and reusable analytical knowledge.

2 Research Directions Enabled by Formal Workflow Representation

A shared formal language for representing analytical workflows opens research directions that are difficult to pursue when workflows exist only as prose. We do not claim that this paper realises these directions; rather, we outline them as opportunities that we hope the community will explore once a shared formal foundation is in place. The present paper provides such a foundation: a language, a library, and initial demonstration of analytical and practical affordances. The directions below indicate why we believe the investment in formalisation is worthwhile beyond the immediate contributions demonstrated here.

Composable building blocks and workflow algebra. Complex workflows, when expressed in a formal language with typed constructs, can be systematically decomposed into smaller sub-workflows — reusable *building blocks* with defined input and output artifact types that serve as interface specifications. A library of such blocks, abstracted to the level of analytical intent, could enable *compositional workflow design*: assembling new workflows from proven components according to composition rules. Type compatibility between blocks provides a basic validity criterion (the output types of one block must match the input types of the next); richer constraints such as preconditions and postconditions could eventually provide stronger guarantees. Such a framework would also make the *design space* of analytical workflows navigable: which compositions are valid but unexplored? Which configurations recur so frequently that they constitute standard idioms? The building blocks identified in Section 6 represent early empirical instances; a compositional framework would elevate them from observations to first-class design resources.

Formal requirements for analytical goals. A formal language invites a new class of question: given a desired analytical outcome, what structural properties must a workflow possess? Are certain transform sequences necessary? Do particular output types require iterative structures that include human assessment? Our analysis reveals that all seventeen library workflows share certain structural features: at least one visualise–abstract cycle, at least one assessment point, and an explicit knowledge-generation step. Whether these represent necessary conditions or merely conventions of current practice is an open question, but it is one that formal representation makes answerable: with a sufficiently

Framework	Scope	Cognition	Formalisation	Primary purpose
Van Wijk [2]	Analysis	Modelled	Semi-formal	Formalise visualisation value via specification feedback
VA as model building [3]	Analysis	Observable only	Conceptual	Conceptualise VA as goal-oriented model building
Sacha et al. KG model [4]	Analysis	Modelled	Conceptual	Explain analyst reasoning
He et al. SoK [5]	Analysis	Modelled	Conceptual	Insight lifecycle taxonomy
NBGM [6]	Design	—	Semi-formal	Compare design decisions
Design Activity [7]	Design	—	Descriptive	Guide design process
IVAS [8]	Design	—	Semi-formal	Optimise system components
Wu et al. [9]	Design	—	Semi-formal	Extract design patterns
CWL, BPMN, VisTrails, Kepler, KNIME [10]–[14]	Execution	—	Formal (executable)	Computational reproducibility and portability
VIS4ML [15]	Analysis (ML)	Partial	Formal (OWL)	Map VA support in ML
Beaucamp et al. [16]	Analysis	—	Quantitative	Measure process quality
ATWL	Analysis	Observable only	Formal (declarative)	Describe, compare, reuse analytical workflows

TABLE 1: Positioning of ATWL in respect to related frameworks.

large library, one could test whether workflows violating these patterns still produce reliable outcomes, distinguishing structural necessities from methodological habits.

Analytical strategies and human–machine collaboration. Because ATWL distinguishes human-dominated from machine-dominated transforms and captures their sequencing, it provides a vocabulary for characterising *analytical strategies* — different ways of arranging the same building blocks to pursue an analytical goal. Exploratory strategies place pattern discovery before hypothesis formation; confirmatory strategies begin with specifications that constrain computation; comparative strategies juxtapose multiple branches before synthesis. These distinctions, currently informal in the literature, become formally identifiable as composition patterns over typed primitives. Formal representation also enables systematic study of *human–machine labour division*: which transforms are exclusively human-dominated across known workflows? As AI capabilities expand, a formal account of workflow structure offers a framework for reasoning about where automation can substitute for human cognition and where it cannot.

The present paper provides foundation rather than instances of this programme: a language with sufficient precision for structural reasoning (Section 4), and evidence that formalisation is practical (Section 5) and analytically productive (Sections 6–7). Realising the directions above requires community adoption and library growth, as discussed in Section 8.

3 Related Work

ATWL draws on and complements a substantial body of research on conceptual models, ontologies, and design frameworks for visual analytics. Rather than reviewing each contribution in isolation, we organise the discussion around five questions that together define ATWL's position in the landscape: (1) what are the conceptual origins of the present work? (2) how have researchers modelled the *cognitive* side of VA processes? (3) how have they modelled the *design* of VA systems? (4) how have formal languages been used to represent *computational workflows* and their provenance in adjacent fields? and (5) what formal representations have been proposed for VA *workflows* themselves?

3.1 Conceptual Origins

The present work develops ideas first introduced by Andrienko et al. [3], who proposed viewing visual analytics as a goal-oriented model-building process directed at constructing an appropriate behavioural model of a piece of reality. That framework defined a conceptual vocabulary including subjects and their aspects, structural and behavioural models, focus relationships, and model appropriateness criteria, and represented the analytical process as a directed workflow of data transformation, initial model generation, evaluation, and iterative development. Several core ideas of ATWL originate in that framework: the distinction between entities and their attributes, the role of data transformations in making focus relationships observable, the centrality of model evaluation as a driver of iteration, and the explicit treatment of prior knowledge as an input to the process.

A second conceptual origin is van Wijk's formal model of the visualisation process [2] introducing *specification-mediated feedback* as the mechanism by which human analytical judgment enters the computational process. ATWL adopts this concept as a core design element (`specification` is one of its eight artifact types) and generalises van Wijk's formulation beyond visualisation parameters to encompass all forms of analyst-produced directives, including feature selections, model configurations, query constraints, and method choices.

However, both frameworks remained *conceptual accounts* of aspects of visual analytics; neither provided a formal language in which concrete workflows could be specified, compared, or computationally processed. ATWL addresses this gap by developing their conceptual vocabulary into a declarative specification language with typed artifacts, defined transform intents, well-formedness constraints, and control structures for iteration and branching. Where the model-building framework identified the need for systematic cataloguing of analytical approaches and their transfer across application domains (Sections 6.1–6.3 of [3]), ATWL provides the representational machinery to realise that vision.

3.2 Models of Analytical Cognition

Several influential frameworks model visual analytics as a cognitive process. Sacha et al. [4] propose a knowl-

edge generation model that integrates the computer side of VA (data, models, visualisations) with the human cognitive side, representing the analyst's reasoning as three nested loops—exploration, verification, and knowledge generation—driven by internal constructs such as hypotheses, findings, and insights. He et al. [5] complement this macro-level view with a micro-level analysis of the insight lifecycle, tracing how individual insights are discovered through interaction with visualisations, externalised through annotation, and communicated through reports and data stories; their distinction between *insights into the data* (correlations, trends) and *insights into the domain* (contextualised understanding) provides a useful taxonomy of analytical products.

ATWL shares with these frameworks the overarching goal of providing a structured account of VA processes, and certain correspondences are direct: ATWL's *pattern* artifacts map onto data-level insights, while *knowledge* artifacts map onto domain-level insights in He et al.'s taxonomy; the computer-side concepts in Sacha et al.'s model (data, models, visualisations) can be seen as coarse-grained precursors of ATWL's eight artifact types. However, ATWL differs from both frameworks in a fundamental design choice: it deliberately restricts itself to what is explicitly externalised by the analyst (recorded statements, specifications, labels, judgements) and does not attempt to describe cognitive processes that remain internal to the human mind. This sacrifices explanatory reach for formal precision: ATWL representations are machine-readable and verifiable against observable analytical products, whereas cognitive models offer richer explanatory power for understanding *why* analysts act as they do, at the cost of concepts that cannot be directly observed or formalised. The restriction aligns with He et al.'s emphasis on *insight externalisation* as the critical bridge between cognition and shareable products: ATWL can be seen as a formal language for representing exactly those externalised artifacts that their framework identifies as the observable outputs of the insight process. The approaches are thus complementary: cognitive models provide the context within which ATWL-encoded workflows are executed; ATWL captures the larger analytical architecture within which individual insights are produced.

3.3 Models of VA System Design

A second family of frameworks addresses how VA systems should be *designed*. Meyer et al. [6] extend Munzner's four-level nested model [17] into the Nested Blocks and Guidelines Model (NBGM), introducing finer-grained *blocks* (design outcomes) and *guidelines* (relationships between blocks) within and across design levels. McKenna et al. [7] propose the Design Activity Framework, modelling visualisation design as four overlapping activities—understand, ideate, make, deploy—linked to the nested model's levels. Chen and Ebert [8] take an information-theoretic perspective in their ontological framework IVAS, categorising design shortcomings as *symptoms*, reasoning about their *causes*, and identifying *remedies* using 24 abstract entities derived from crossing four system components with three cost–benefit measures. Wu et al. [9] conduct a meta-analysis of 220 VA papers to map relationships between analytical requirements and design solutions, extracting problem-driven design patterns organised in knowledge graphs.

ATWL and these design frameworks share a conviction that formalisation enables cross-domain comparison, and all arrive at structured vocabularies that abstract away domain-specific terminology. The key distinction is temporal. Design frameworks operate at *design time*: they reason about which abstractions, encodings, and interactions a system should provide. ATWL operates at *analysis time*: it captures what happens when an analyst uses a deployed system to progress from data to knowledge. This shift is reflected in classification vocabulary: a clustering algorithm is a "Clustering & Grouping" manipulation in Wu et al.'s solution typology but a `define-unit` transform in ATWL—the difference between *what the system does* and *why the analyst does it*.

The two perspectives are complementary. Design frameworks could inform the selection of tools that populate individual ATWL transforms; conversely, ATWL's workflow structures - iterative loops, feedback paths, progressive abstraction - could provide design frameworks with the analytical process context they currently lack.

3.4 Scientific Workflow Languages and Provenance Frameworks

A substantial body of work addresses formal representation of computational workflows. BPMN [11] specifies processes as typed task sequences with execution semantics; scientific workflow systems such as Kepler [13], VisTrails [12], and KNIME [14] compose and execute computational pipelines from typed modules; CWL [10] provides portable pipeline specifications emphasising cross-platform reproducibility; and W3C PROV [18] represents derivation relationships between entities through activities attributed to agents, a structure superficially similar to ATWL's artifact–transform–actor triad.

ATWL differs in two respects. First, it operates at the level of *analytical intent* rather than computational execution: a clustering step in CWL specifies an algorithm and its parameters; in ATWL it specifies the intent to `define-unit` by similarity-based grouping, enabling comparison regardless of method. Second, ATWL treats human-dominated transforms as first-class typed components, whereas scientific workflow systems model purely computational pipelines [19]. For visual analytics, where human interpretation is constitutive, this distinction is essential. The two levels are complementary: CWL could implement machine-dominated ATWL transforms, while PROV could record their provenance. ATWL contributes the layer these systems do not address: a vocabulary for the analytical logic of human–computer collaboration.

3.5 Formal Workflow Representations

The most directly related work concerns formal representations of VA workflows themselves.

Sacha et al. [15] propose VIS4ML, a formal ontology for VA-assisted machine learning implemented in OWL. VIS4ML decomposes ML workflows into four phases—Prepare-Data, Prepare-Learning, Model-Learning, Evaluate-Model—and represents individual workflows as pathways

through a fixed ontological structure, with "bus stops" marking points where visualisation assists the ML process. Its bipartite architecture of *Processes* and *IO-Entities* (Data, Model, Knowledge) corresponds directly to ATWL's transform–artifact structure, making VIS4ML the closest predecessor to our work.

The two frameworks differ in three key respects. First, *scope*: VIS4ML is specialised to VA-assisted ML, whereas ATWL is domain-agnostic, covering temporal analysis, movement analytics, text mining, and statistical modelling with the same vocabulary. Second, *granularity*: VIS4ML's three IO-Entity subclasses are coarse-grained compared to ATWL's eight artifact types, which distinguish computed features from raw entities, organisational arrangements from visual representations, and evaluative knowledge from control specifications. Third, *representational strategy*: VIS4ML traces workflows as pathways through a single fixed ontological template, which is powerful for identifying under-explored phases but constrains every workflow to the same structure; ATWL represents each workflow as a standalone specification with its own dependency structure and control flow, enabling the discovery of emergent cross-workflow patterns, such as structural isomorphisms and a taxonomy of iterative strategies, that arise from comparing freely structured specifications rather than mapping them onto a predetermined schema.

Beaucamp et al. [16] take a complementary approach, translating Chen and Golan's [20] information-theoretic cost–benefit framework into a methodology for quantitatively analysing hybrid decision workflows. Their approach decomposes workflows into component processes and measures how well each transforms information, using entropy and divergence to quantify benefit and distortion. While ATWL is a *structural* language capturing what operations are performed, on what objects, in what order, and by whom, Beaucamp et al. provide a *quantitative* framework for evaluating how well each operation performs. The two are directly composable: ATWL could supply the structural decomposition that their methodology requires as input, while their information-theoretic measures could enrich ATWL-encoded workflows with quantitative performance annotations.

3.6 Positioning ATWL

Table 1 summarises the positioning of ATWL relative to the reviewed frameworks along four dimensions: temporal scope (design time vs. analysis time), treatment of cognition (modelled vs. restricted to observables), level of formalisation, and primary purpose.

ATWL occupies a distinctive position in this landscape. It is the only framework that combines (a) domain-agnostic scope, (b) analysis-time focus, (c) formal, machine-readable syntax, and (d) restriction to observable, externalised analytical products. The first three properties it shares partially with VIS4ML; the fourth it shares with the model-building framework [3] from which ATWL descends and, in spirit, with He et al.'s emphasis on insight externalisation. Their conjunction into a formally precise, domain-independent language for the observable structure of analytical processes enables the contributions reported in this paper: automated extraction of workflow representations from research literature, systematic cross-workflow analysis yielding general knowledge about VA practice, and LLM-assisted workflow design that the formal library augments with explicit structure and traceable provenance.

4 Language Design

This section presents the design of ATWL in three parts: the conceptual model that defines the general structure of workflows, the ontologies that classify the building blocks of analytical processes, and a concise overview of the language syntax. The complete definition of the language is available at the URL https://geoanalytics.net/VAworkflows/ATWL_definition.pdf, the language design process is described in Section 4.5.

4.1 Conceptual Model

The central abstraction in ATWL is the *analytical workflow* consisting of two kinds of building blocks: informational objects, called *artifacts*, and operations, called *transforms*, producing new artifacts from existing ones. Each transform consumes one or more input artifacts and produces one or more output artifacts; the resulting input–output dependencies capture the logical progression of analysis from raw data to derived insights. Figure 1 gives an overview of the conceptual architecture with artifacts as nodes and transforms as directed edges, as described below.

Artifacts represent different kinds of objects with different analytical roles, structured across five layers: data, organization, presentation, interpretation, and epistemic. Each artifact has a *type* drawn from a fixed ontology (Section 4.2, Table 2), which constrains its internal description and determines how it can participate in transforms.

Transforms are classified not by computational method or algorithm but by their analytical *intent*, drawn from a fixed ontology (Section 4.3, Table 3). The intent declares the purpose of the transform (what it achieves analytically), enabling structural comparison of workflows that employ different methods to achieve the same analytical goal.

Every artifact in a workflow is either *exogenous* or *derived*. Exogenous artifacts, such as raw datasets, prior domain knowledge, and initial parameter settings, are supplied as inputs to the workflow and are marked with the designation `origin: given`. Derived artifacts are produced as outputs of transforms. This distinction makes explicit the boundary between exogenous inputs and artifacts derived within the workflow.

The design of ATWL is guided by four principles:

1) **Distinct artifact roles.** The artifact ontology assigns each artifact to one of five roles in the analytical process: data (entities, features), organisation (arrangements), representation (visualisations), interpretation (patterns, models), and epistemic output (knowledge, specifications). As illustrated by the layered structure in Figure 1, this stratification makes explicit the progressive transitions between data manipulation, visual encoding, and human reasoning that are typically conflated in prose descriptions. The layers are bridged upward by transforms that enact an analytical ascent (e.g., *visualise*

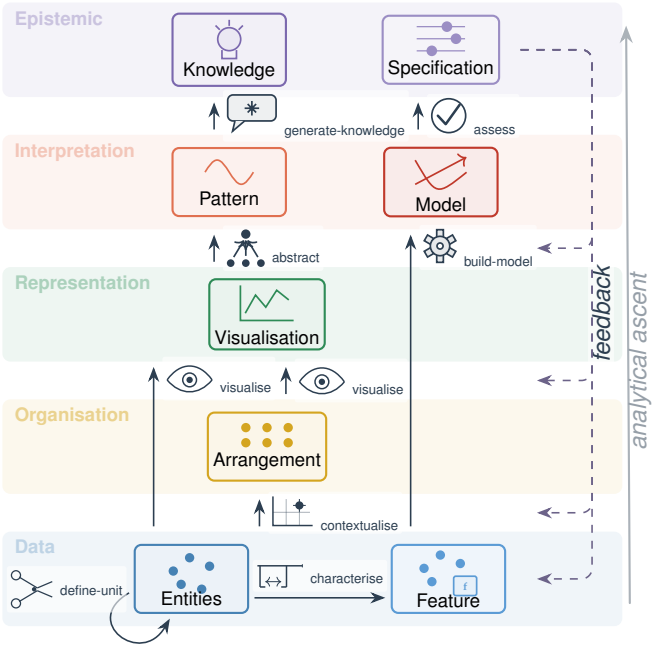


Fig. 1: Conceptual architecture of ATWL. Five analytical-role layers contain eight artifact types (nodes); transform intents (edges) connect them across layers, ascending from raw data to epistemic outcomes.

maps data and organisational artifacts into representations), and downward by feedback, through which specifications and knowledge at the epistemic layer re-enter earlier processing stages.

2) **Intent-driven transforms.** Transforms are classified by their analytical purpose rather than their computational implementation. A clustering algorithm and a manual grouping by a domain expert may both realise the same intent (`define-unit`) despite radically different implementations.
3) **Knowledge as a first-class artifact.** Explicitly externalised knowledge in the form of recorded statements, judgments, specifications, labels, rules, etc., is given the same formal status as data, features, and visualisations: it is typed, named, and connected to transforms as input or output. At the same time, ATWL does not attempt to describe cognitive processes that remain internal to the analyst's mind, unlike conceptual frameworks that model hypothesised mental activities (e.g., [4]). Every element of a workflow description thus corresponds to an observable, communicable analytical product.
4) **Domain independence through composable primitives.** Artifact types and transform intents are defined at a level of abstraction that surpasses specific application domains while remaining sufficiently expressive to capture the essential structure of diverse workflows.

4.2 Artifact Ontology

ATWL defines eight artifact types that collectively span the full lifecycle of a visual analytics process, structured along five analytical role layers. Table 2 provides an overview.

Artifact type	Description
Data	
Entities	Collections of identifiable objects, each treated as a unit of analysis
Feature	Explicit descriptor of entity properties or relationships
Organisation	
Arrangement	Positioning of entities within a reference context
Representation	
Visualisation	External visual representation for human perception
Interpretation	
Pattern	Regularity, trend, or structure identified through abstraction
Model	Formal/computational representation for prediction or explanation
Epistemic	
Knowledge	Explicitly formulated substantive knowledge: insights, judgments, expertise
Specification	Control directives: parameters, constraints, or method choices directing transforms

TABLE 2: The eight ATWL artifact types, organised by analytical role.

Intent	Purpose
`define-unit`	Create or redefine entities as units of analysis
`characterise`	Compute or transform features describing entities
`contextualise`	Position entities within a reference context
`visualise`	Create visual representations for human perception
`abstract`	Derive patterns or conceptual structures
`build-model`	Construct or refine a formal/computational model
`generate-knowledge`	Formulate explicit knowledge or specifications
`assess`	Evaluate quality or appropriateness of artifacts

TABLE 3: The eight ATWL transform intents characterising analytical purpose rather than implementation.

4.2.1 Data Layer: Entities and Features

Entities represent the fundamental units of analysis. Each entities artifact describes a collection of analytical objects characterised along three orthogonal dimensions.

Internal structure describes how components within each entity are organised. Table 4 lists the six structure types, which fall into three categories: *atomic* entities are indivisible; *container* entities (groups, episodes, regions) enclose components in an unstructured or implicitly structured manner; and *relational* entities (sequences, formations)

organise components through explicit relational structure.

Category	Type	Description	Examples
Atomic	`elementary`	Indivisible object; internal composition not analysed.	Events, measurements, whole images.
Container	`group`	Unordered collection without internal structure.	Clusters of days, related object groups.
	`episode`	Bounded time interval with referenced components.	Time series segments, sports match episodes.
	`region`	Bounded spatial extent with spatial components.	City districts, spatial measurement cells.
Relational	`sequence`	Components in a linear, essential order.	Event sequences, sentence words, task lists.
	`formation`	Networks, hierarchies, or spatial neighbourhoods.	Network snapshots, hierarchies, team formations.

TABLE 4: Six types of internal structure for entities, categorized by Atomic, Container, and Relational.

Embedment describes the shared environment in which the entities of a collection reside and relate to one another. Table 5 lists the five embedment types. Multiple embedment types may co-occur, for example, entities embedded in both time and space are denoted `{time, space}`. Embedment is omitted when the artifact represents a single entity.

Features within entities. A declaration of an entities artifact may include one or more features. Every feature declaration specifies a *value structure* (how the values are organised: `atomic`, `list`, `vector`, `matrix`, or `relational configuration`) and, optionally, a *value type* (the nature of the atomic components: `numeric`, `ordinal`, `categorical`, `temporal`, `spatial`, `text`, or `reference`). When atomic components are of mixed types, set notation is used (e.g., `{numeric, temporal}`).

Features artifacts describe properties of entities or relationships between them that are important for subsequent analysis. Internal features (inside an `entities` artifact) provide background semantics; `feature` artifacts are used when features become *first-class operands* of transforms, i.e., their inputs or outputs. Feature artifacts are typically produced by transforms with intent `characterise` (Section 4.3).

A `feature` artifact references the `entities` artifact it describes and can have the same value-structure and value-type descriptors as internal features of `entities` artifacts. It may additionally specify a *representation form* that clarifies encoding for complex features (e.g., "distance matrix", "topic-distribution vector", "k-NN graph").

4.2.2 Organization Layer: Arrangements

Arrangements organise entities in a reference structure (context), establishing positions or placements to support analysis. It does not create new entities or values; it defines how existing entities are positioned within a reference structure, such as a calendar, geographic space, grid, or artificial projection space resulting from dimensionality reduction. The reference structure itself must be previously explicitly declared as an `entities` artifact. The *principle* field in the declaration of an `arrangement` artifact describes the organising logic, e.g., "calendar(year, month, weekday)", "2D projection based on feature similarity".

Embedment	Description
`set`	Unordered collection
`sequence`	Linearly ordered positions
`time`	Temporal axis
`space`	Spatial reference
`relational structure`	Network, hierarchy, or similar

TABLE 5: Embedment types for entities.

4.2.3 Representation Layer: Visualisations

Visualisations are an external visual representation of an arrangement, designed for human perception. It references the artifact(s) it depicts and is characterised by three properties: *layout* (the structure of the visual space, e.g., "calendar grid", "node-link layout"), *form* (the type of visual marks, e.g., "coloured cells", "line segments"), and *encoding* (the mapping of data attributes to visual channels, e.g., "colour $\rightarrow$ category, size $\rightarrow$ frequency").

Note: The concepts *'embedment'*, *'arrangement'*, and *'visualisation'* capture distinct levels of organisation. *Embedment* is a data-level property of entities: the environment in which they inherently reside (time, space, a relational structure). *Arrangement* is an analytic-level artifact produced by a `contextualise` transform: it maps entities onto positions in a context structure (calendar cells, geographic partitions, projection coordinates) and can serve as input to further computation or as a basis for visualisation. *Visualisation* is a perceptual-level artifact produced by a `visualise` transform: it maps artifacts into visual space, deriving layout from an arrangement when one exists or directly from entities or features otherwise. The three levels form a progression: inherent structure $\rightarrow$ analytical organisation $\rightarrow$ perceptual encoding.

4.2.4 Interpretation Layer: Patterns and Models

Patterns represent regularities, trends, or structures identified through abstraction performed by automated algorithms, human perception, or both. They reference the artifacts from which they are derived and declare a *representation form*, e.g., "ranked list of motifs", "textual descriptions", "cluster labels".

Models are formal or computational representations used for prediction, simulation, or explanation. They specify a *model type* (e.g., "classifier", "topic model", "regression model") and optionally a representation form (e.g., "decision tree", "neural network weights").

Distinction between models and patterns: Patterns describe *observed* regularities *within* the available data (descriptive and bounded by observations), while models provide mechanisms that *generalise beyond* the available data. Models support interpolation between available data points and extrapolation to new contexts (future time points, unobserved spatial locations, new populations or products, etc.).

4.2.5 Epistemic Layer: Knowledge and Specifications

Knowledge artifacts represent explicitly formulated understanding. They may be *derived* during analysis (insights, explanations, rules, decisions, recommendations) or *injected* by the analyst or from external sources (domain assumptions, constraints, rankings, labels, feedback). Derived knowledge is typically produced by `generate-knowledge` transforms; an important sub-case is *evaluative* knowledge produced by `assess` transforms, which captures quality judgments or adequacy decisions about other artifacts. Injected knowledge is marked `origin: given` and enters the workflow as input to transforms, making the role of prior expertise formally explicit.

Specifications represent parameters, configurations, constraints, or method choices that control how transforms operate (e.g., distance thresholds, number of clusters, aggregation strategies, desired output properties). They encode *control directives*, i.e., decisions about *how* subsequent analysis should be carried out, and are consumed by transforms whose behaviour depends on configurable choices. Specifications may be exogenous (`origin: given`) or derived: most commonly by `generate-knowledge` transforms in which the analyst formulates a methodological decision, or by `assess` transforms that trigger parameter adjustments. Within iterative loops, specifications are frequently updated as the analyst progressively refines the analytical strategy.

The distinction between the two epistemic types reflects different functional roles: `knowledge` captures *what is understood*, while `specification` captures *what is decided*. The role of specifications as mediating artifacts between human judgment and computational execution was formalized by van Wijk [2], whose model of the visualisation process represents the analyst's decisions as specifications that control image generation. ATWL generalises this concept beyond visualisation parameters to encompass all forms of machine-consumable directives produced by human analytical reasoning, including method choices, constraints, selection criteria, and model configurations.

4.3 Transform Ontology

Transforms consume the input artifacts and produce output artifacts, towards the upper layers in the ATWL architecture. Each transform is characterised by three properties: *intent*, *manner*, and *actor*.

Intent is the main property, indicating the analytical purpose of the transform. Table 3 lists and describes the eight generic intents defined in ATWL. **Manner** optionally specialises the intent with a free-text description of how the transform achieves its purpose, when appropriate. For example, a `define-unit` transform might specify manner as "cluster-by-similarity" or "time-partitioning"; a `contextualise` transform might specify "projection-based" or "calendar-based". Manner provides methodological detail without expanding the intent vocabulary, preserving the ability to compare workflows at the level of intent alone.

Actor specifies the agency responsible for the transform: `human` (analyst only), `machine` (computation only), or `hybrid` (human–machine collaboration). This dimension makes explicit the distribution of cognitive and computational labour across the workflow and opens up the ATWL language for human-AI interaction and collaboration analysis.

4.4 Syntax and Control Structures

ATWL uses a declarative, YAML-like syntax (summarised in Appendix A). A workflow specification begins with a `workflow` declaration and consists of interleaved artifact and transform declarations. Listing 1 illustrates the core syntax patterns with a minimal workflow fragment.

Listing 1: An illustrative ATWL fragment.

```
workflow example_workflow

artifact D_events : entities
  origin: given
  internal structure: elementary
  embedment: {set, time}
  features:
    - id: timestamp
      value structure: atomic
      value type: temporal
    - id: category
      value structure: atomic
      value type: categorical
  description: "Timestamped event records"

transform T1 :
  intent: characterise
  manner: "aggregate by time period"
  input: D_events
  output: F_daily
  actor: machine

artifact F_daily : feature(D_events)
  value structure: vector
  value type: numeric
  description: "Daily counts per category"

transform T2 :
  intent: visualise
  input: D_events, F_daily
  output: V_timeline
  actor: machine

artifact V_timeline :
    visualisation(D_events, F_daily)
  layout: "temporal axis"
  form: "stacked area chart"
  encoding: "x: date, y: count, colour:
    category"
```

Artifact declarations specify the artifact's type and, for exogenous artifacts, the marker `origin: given`. The type keyword may be followed by parenthesised references to related artifacts (e.g., `feature(D_events)`). Transform declarations specify intent, optional manner, input and output artifact identifiers (comma-separated), and actor.

Control structures. ATWL provides three constructs for expressing iterative and conditional workflow logic:

- **Loops** (`loop` . . . `end loop`) declare iterative refinement with a named identifier and a qualitative stopping condition (`until`). The loop body may contain an explicit `assess` transform with a conditional exit, or rely on implicit analyst-monitored termination.

- **Conditionals** (`if ... then ... else`) express branching based on artifact properties or human judgments. The directive `exit loop <ID>` in a branch terminates an enclosing loop.
- **Assignments** (`assign`) bind an artifact identifier to a new version. They appear before loops (for initialisation) or inside loop bodies (for iterative update), providing the mechanism by which the otherwise acyclic dependency graph accommodates iteration.

Workflow template. The `workflow` declaration begins with a `template` field: a high-level summary of the main analytical stages expressed as a chain of intents (e.g., `define-unit → characterise → visualise → abstract`). Iterative stages may be enclosed in `loop(...)` notation. The template serves as a compact signature of the workflow's analytical strategy.

Validity. A formalized ATWL workflow satisfies four structural constraints: (1) all artifact identifiers are unique; (2) every transform input references a declared artifact; (3) exogenous artifacts never appear as transform outputs; and (4) the chain of input-output dependencies among artifacts and transforms is acyclic, except through explicit `assign` statements within loops.

4.5 Language Design Process

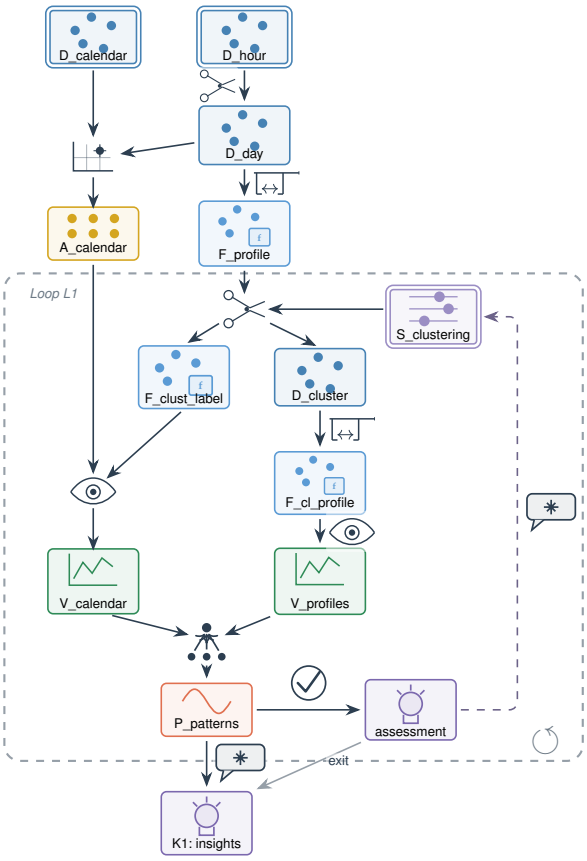
ATWL's design evolved through multiple iterations of conceptualisation, formalisation, and empirical testing against published workflows. Through this process, a minimal set of artifacts, artifact roles, and intent-driven transforms was identified, sufficient to describe elements of analytical workflows observed in practice.

The most significant design transition was a shift from an operation-centric formalism (in which named operations were the primary constructs) to the current artifact–transform duality. The driving insight was that workflow meaning is determined primarily by the types and roles of artifacts produced rather than by operational details. This motivated the architecture presented above, in which transforms are uniform connectors characterised by intent, manner, and actor, while artifacts carry the semantic differentiation through their types.

Several further design decisions resolved ambiguities encountered during formalisation: separating the identification of instances in data (`define-unit`) from generalisation across instances (`abstract`); treating spatial/temporal/similarity-based positioning (`contextualise`) as distinct from visual encoding (`visualise`); elevating computed features from mere data attributes to first-class artifacts; and introducing `assess` as a distinct intent when evaluation proved irreducible to either knowledge generation or model building. The final vocabulary of eight artifact types and eight transform intents represents a deliberate reduction from a larger initial set, guided by the principle that intents producing the same artifact type under the same analytical logic should be merged into a single intent with free-text manner specialisation.

4.6 Visual Representation of ATWL Workflows

Although ATWL is a textual language, its structured artifact–transform dependencies enable automated translation into flow diagrams. We found that current LLMs can generate such diagrams directly from ATWL specifications, producing publication-ready vector drawings with modest corrective feedback. Appendix B provides a complete example: an ATWL representation of the Cluster-Calendar workflow [21] and a flow chart generated by an LLM agent (Fig. 4). Flow diagrams complement textual representations by making the overall workflow topology immediately perceptible by a human. A simplified visualisation of the workflow with the use of the icons is shown in Fig. 2.



Legend: Double border = given artifact. Dashed box = iterative loop. Dashed arrow = feedback (parameter update).

Fig. 2: Visual representation of the cluster-calendar workflow [21] using ATWL icons. Small icons on arrows indicate transform intents.

5 Workflow Library

To evaluate the expressive power of ATWL, assess the feasibility of LLM-assisted workflow formalisation, and create a resource for future experiments on intelligent workflow design support, we constructed a library of seventeen visual analytics workflows drawn from published research papers. The selection was guided by three criteria. First, we sought papers that describe *complete analytical workflows* consisting of multiple steps from data to knowledge. This requirement proved selective: many VA papers focus on individual techniques or tools with usage examples that illustrate capabilities rather than demonstrate a complete analytical process. Second, we aimed for diversity of data types and analytical paradigms but did not strive for representative coverage of the entire field of visual analytics and creating a statistical sample. Third, ATWL is designed to be extensible:

#	Workflow	Source	Data type	Analytical focus
Exploratory pattern discovery				
1	Cluster-Calendar	[21]	Time series	Temporal pattern clustering and calendar visualisation
2	Snapshots-to-Points	[22]	Dynamic network	Network state identification via projection
3	MobilityGraphs	[23]	Spatio-temporal flows	Mobility pattern analysis via spatial and temporal simplification
6	Events-to-Places	[24]	Movement trajectories	Significant place extraction through event clustering
7	Progressive Clustering	[25]	Movement trajectories	Multi-phase progressive trajectory clustering
9	Episodes and Topics	[26]	Multivariate temporal	Progressive abstraction through encoding, topic modelling, and distribution analysis
Data transformation for comprehensibility				
4	EventFlow	[27]	Event sequences	Iterative sequence simplification for visual comprehensibility
5	EventAction	[28]	Event sequences	Prescriptive recommendation through cohort similarity
Model building and validation				
8	UTOPIAN	[29]	Text documents	Interactive topic model refinement via semi-supervised NMF
10	Partition-based Regression	[30]	Tabular numeric	Iterative regression building through residual analysis
11	ST Analysis & Modelling	[31]	Spatial time series	Spatio-temporal time series model fitting and validation
12	Behaviour Pattern Recognition	[32]	Movement trajectories	Feature engineering for movement behaviour classification
Model understanding and diagnostics				
13	Exploratory Model Analysis	[33]	Multi-type dataset	Predictive model discovery and selection
14	Binary Classifier Diagnostics	[34]	Tabular (binary target)	Classifier diagnosis via instance-level explanations
15	Random Forest Exploration	[35]	Trained ensemble	Interpretable rule extraction from ensemble classifier
16	TensorFlow Graph Visualiser	[36]	DL dataflow graph	Deep learning architecture understanding
17	What-If Tool	[37]	ML model + tabular	Model probing, counterfactual analysis, and fairness evaluation

TABLE 6: Overview of the ATWL workflow library. Group headers indicate the analytical paradigms.

should new workflows require constructs not yet defined, the ontology can be expanded. We therefore prioritised breadth over exhaustiveness, anticipating that the library will grow through community contributions (Section 8).

The library was built through a human–LLM collaborative process described in Appendix C: LLM agents extracted initial ATWL representations from research papers, which were then reviewed and corrected by human experts with knowledge of the original workflows. The complete ATWL representations are accessible at https://geoanalytics.net/VAworkflows/library/. To confirm that the methodology is reproducible, the extraction was repeated with fresh LLM instances that had not participated in any prior work. These independently produced correct representations demonstrating that the documented procedures transfer beyond the original agents.

Table 6 summarises the library. The workflows represent four distinct analytical paradigms, reflected in the grouping: exploratory pattern discovery, data simplification for comprehensibility, iterative model building, and model understanding through diagnostic exploration. Table 7 shows ATWL construct coverage. The library exercises all artifact types and transform intents. Structural complexity ranges from a single linear pipeline (workflow 2) to three levels of nested loops with conditional branching (workflow 11).

6 Cross-Workflow Analysis: What Formal Representation Makes Visible

Representing VA workflows in a formal language transforms them from narratives into comparable, analysable objects. A library of formally encoded workflows enables questions that no single paper can answer: What analytical patterns recur? Do workflows from different domains share common structure? How do analysts iterate, and how is labour divided between humans and machines? These questions become tractable through three properties of formal representation: a shared vocabulary abstracting away domain-specific terminology, typed artifact interfaces making structural compatibility explicit, and machine-readable specifications enabling computational search and matching.

We demonstrate this potential by analysing the seventeen-workflow library for structural regularities. The analysis was conducted by a human researcher with LLM assistance: the formal nature of ATWL representations enabled the LLM to systematically compare workflows and compute structural statistics, while the human researcher directed the inquiry and assessed significance. This collaboration illustrates that formal representations make workflows accessible to machine reasoning, amplifying the scale of analysis beyond manual inspection.

We report four types of findings, each characterising a different facet of workflow structure at a different level of granularity.

Meta-structure (Section 6.1) is a five-stage progression shared by all seventeen workflows, characterising the overall shape of a complete workflow from data ingestion to knowledge synthesis. It provides the frame within which the remaining findings are situated.

Recurring motifs (Section 6.2) are structural regularities in the arrangement of transforms, artifacts, and actors that can be recognised by their intent-level structure alone, independent of domain or implementation. They characterise fundamental aspects of how VA workflows are organised: how human judgment enters computation, how abstraction proceeds, and how iteration is structured at multiple levels. Unlike building blocks, which are defined by typed interfaces for composition and transfer, motifs are identified by their recurring *presence*. They are signatures of analytical organisation rather than units of reuse.

Methodological building blocks (Section 6.3) are named, composable sub-workflows with *defined input and output artifact types*. They carry *typed interfaces* that specify what they consume and produce, making them units of principled composition and cross-domain transfer. Some motifs corre-

#	Workflow	Ref.	Artifact types								Transform intents								Iteration
1	Cluster-Calendar	[21]	■	■	■	■	■		■	■	■	■	■	■	■		■	■	1
2	Snapshots-to-Points	[22]	■	■	■	■	■		■		■	■	■	■	■		■		—
3	MobilityGraphs	[23]	■	■		■	■		■	■	■	■		■	■		■	■	2
4	EventFlow	[27]	■	■	■	■	■		■	■	■	■	■	■	■		■	■	2
5	EventAction	[28]	■	■		■	■		■	■	■	■		■	■		■	■	1
6	Events-to-Places	[24]	■	■		■	■		■	■	■	■		■	■		■	■	1
7	Progressive Clustering	[25]	■	■		■	■		■	■	■	■		■	■		■	■	2
8	UTOPIAN	[29]	■	■	■	■	■		■	■	■	■	■	■	■		■	■	1
9	Episodes and Topics	[26]	■	■	■	■	■		■	■	■	■	■	■	■		■	■	3
10	Partition-based Regression	[30]	■	■		■	■	■	■	■	■	■		■	■	■	■	■	1
11	ST Analysis & Modelling	[31]	■	■		■	■	■	■	■	■	■		■	■	■	■	■	3†
12	Behaviour Patterns	[32]	■	■	■	■	■	■	■		■	■	■	■	■	■	■	■	2
13	Exploratory Model Analysis	[33]	■	■		■	■	■	■	■	■	■		■	■	■	■	■	1
14	Bin. Classifier Diagnostics	[34]	■	■		■	■	■	■	■	■	■		■	■	■	■	■	2†
15	Random Forest Exploration	[35]	■	■	■	■	■	■	■	■	■	■	■	■	■		■	■	1
16	TensorFlow Graph Vis.	[36]	■	■	■	■	■		■		■	■	■	■	■		■	■	1
17	What-If Tool	[37]	■	■		■	■	■	■	■	■	■		■	■		■	■	2
	Total		**17**	**17**	**8**	**17**	**17**	**7**	**17**	**15**	**17**	**17**	**8**	**17**	**17**	**6**	**17**	**16**	**26**

TABLE 7: ATWL construct coverage across the library. Filled squares indicate presence. Left block: artifact types; right block: transform intents; ↻: loop count; †: nested loops. Icon legend: see Figure 1 and Tables 2–3.

Stage	Dominant transforms	Actor
1. Representation construction	`define-unit`, `characterise`	Machine
2. Contextualisation *(optional)*	`contextualise`	Machine
3. Iterative analysis	`visualise`, `assess`, `define-unit`	Hybrid
4. Pattern recognition	`abstract`	Human
5. Knowledge synthesis	`generate-knowledge`	Human

TABLE 8: Five-stage meta-structure observed across all seventeen library workflows.

spond directly to building blocks (the assess–refine motif is formalised as a building block with explicit interface types); others describe structural organisation that is not directly composable.

Iterative strategy types (Section 6.4) characterise the different ways in which loops are organised across the library, distinguished by the depth of human intellectual engagement they require, from adjusting parameters within a fixed framing to restructuring the analytical representation itself.

Methodological note. The findings reported below are based on seventeen workflows selected to span diverse domains and paradigms, but not sampled randomly from the population of all VA workflows. We present them as empirically grounded hypotheses that can be tested, refined, or refuted as the library grows. The primary contribution of this section is not the specific patterns themselves but the demonstration that formal cross-workflow analysis can reveal regularities that require systematic structural comparison to detect. Such comparison is not supported by prose descriptions.

6.1 A Recurrent Meta-Structure

When expressed in ATWL, all seventeen workflows, despite originating from six application domains, spanning fifteen years of research, and ranging from linear pipelines to nested-loop architectures, share a common five-stage progression (Table 8). This meta-structure is not a definitional consequence of the ATWL vocabulary: nothing in the language requires that these intents appear in this order or that all five stages be present. That all seventeen workflows exhibit this progression is an empirical observation, not a logical necessity of the formalism.

The overall shape—machine computation followed by iterative human–machine dialogue followed by human interpretation—is consistent with existing conceptual models of VA reasoning [3], [4], [38]. However, the formal cross-workflow analysis adds specificity that these models do not provide: two distinct entry modes, contextualisation as an identifiable design decision (present in 8/17 workflows), specific transform–actor assignments across stages, and the precise location of knowledge generation both within and beyond loops. These details are elaborated in Appendix D.2.

Two entry modes: data-centric and model-understanding. Stage 1 takes two forms: in *data-centric* workflows (12/17), raw inputs are converted into analytical units with computed features; in *model-understanding* workflows (5/17), characterisations are computed for pre-existing objects. This initial difference propagates through subsequent stages: data-centric workflows predominantly employ computational refinement loops with specification-mediated feedback and machine-executed visualisation, while model-understanding workflows predominantly employ exploratory loops with interactive (hybrid) view configuration. Whether these downstream differences reflect genuinely distinct modes of analytical reasoning with different tool-support requirements is a question a larger and more diverse library could help answer.

Cognitive vs. computational labour. The meta-structure reveals a consistent division of labour across the library: machine actors dominate Stages 1–2, human actors dominate Stages 4–5, and Stage 3 is characteristically hybrid. This pattern gives formal specificity to the foundational VA principle of combining "the best of both sides" [38]: machines consistently handle scalable computation while humans consistently handle semantic interpretation, with typed specification artifacts mediating the transfer between them. Details of how this division manifests differently in data-centric versus model-understanding workflows appear

in Appendix D.5.

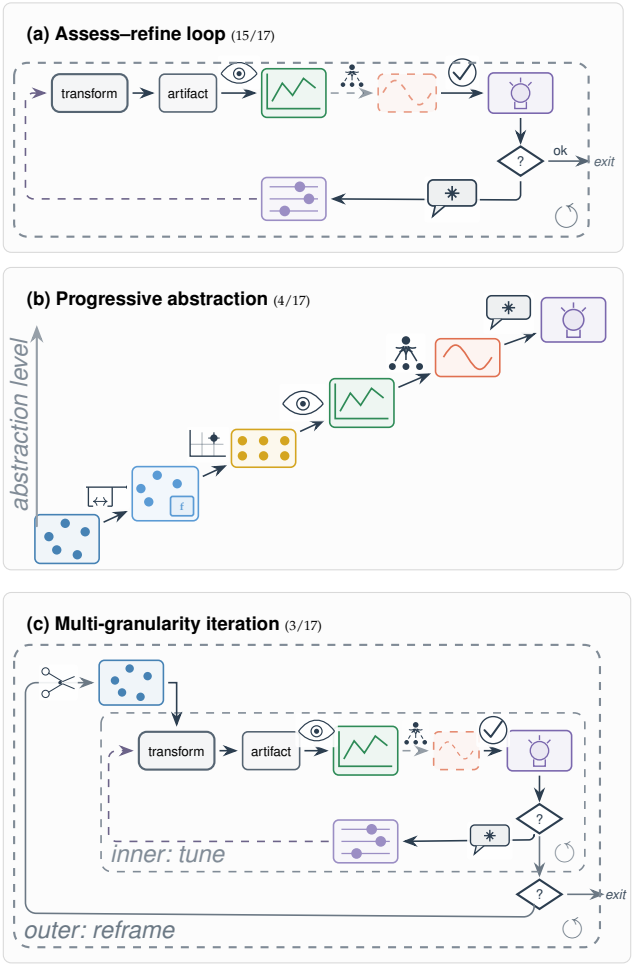


Fig. 3: Three recurring structural motifs. (a) Assess–refine loop (15/17). (b) Progressive abstraction (4/17). (c) Multi-granularity iteration (3/17). Dashed semi-transparent nodes indicate typically arising but non-definitional artifacts.

6.2 Recurring Structural Motifs

Within the global meta-structure, the workflows share recognisable structural motifs. Three motifs are especially prominent (Fig. 3).

***Assess–refine loop* (15/17 workflows).** A computational transform produces an artifact; a `visualise` transform renders it for inspection; an `assess` transform (human-dominated) evaluates quality; and a conditional branch either exits or triggers refinement. This motif is the primary mechanism through which human judgment steers machine computation in visual analytics. Its near-universality suggests it constitutes a structural signature of the discipline.

In the majority of cases (13/17 workflows), the feedback mechanism operates through specification artifacts: rather than manipulating data or parameters directly, the analyst formulates/updates a specification artifact (`generate-knowledge`(spec)) that subsequently parameterises the next computational transform. This separation of judgment from execution means the human contribution is captured as a first-class, typed artifact that is inspectable, revisable, and formally traceable. The specification serves as the typed interface between human cognition and machine computation.

***Progressive abstraction* (workflows 1, 3, 9, 12).** These workflows traverse the full artifact hierarchy (Fig. 1) in sequence: `entities` → `feature` → `arrangement` → `visualisation` → `pattern` → `knowledge`. Each transform lifts the representation to a higher level of analytical abstraction, culminating in explicit knowledge synthesis. This motif characterises workflows whose analytical strategy is systematic bottom-up construction from raw data to understanding.

***Multi-granularity iteration* (workflows 9, 11, 14).** Complex workflows layer loops at different analytical granularities: fine-grained parameter tuning nested within coarse-grained conceptual refinement. The outer loop may restructure the analytical framing while the inner loop optimises within a fixed framing. This motif reveals that some analytical problems require hierarchically organised iteration, not merely repeated adjustment.

These motifs are not individually surprising to experienced practitioners. An assess–refine cycle, for instance, is intuitively understood as central to VA. The contribution of formal representation is to make them *precisely identifiable* and *structurally comparable*: one can now ask whether a given workflow contains the assess–refine motif, how many instances it contains, and whether those instances share the same internal structure as instances in other workflows. Such questions are unanswerable from prose.

6.3 Reusable Methodological Building Blocks

Beyond the motifs of Section 6.2, the library contains more structured sub-workflows that recur across domains with consistent input and output artifact types. We call these *methodological building blocks*: named, composable units with typed interfaces that can be extracted from one workflow and transferred to another. While a motif is identified by its recurring *presence* as a signature of analytical organisation, a building block carries typed interfaces that specify what artifact types it consumes and produces, making it a unit of principled composition and cross-domain transfer. Some motifs correspond directly to building blocks (the assess–refine motif is formalised as SP-1 with explicit interface types); others describe structural organisation that is not directly composable (e.g., progressive abstraction, multi-granularity iteration).

Table 9 lists six building blocks identified in the library, organised into two categories. *Universal mechanisms* (SP-1 and SP-2) characterise visual analytics as a methodology: they represent the fundamental ways in which human judgment, prior knowledge, and iterative feedback enter analytical workflows. Their high frequency (appearing in 11–16 of 17 workflows) is a consequence of their methodological centrality. *Transferable building blocks* (SP-3 through SP-6) are paradigm-specific analytical strategies—concrete approaches to clustering, dimensionality reduction, model diagnosis, or multi-level exploration—that recur across unrelated domains. For example, SP-3 (Feature-then-Cluster) connects temporal pattern analysis, topic modelling, and movement analytics—workflows that use entirely different terminology but share identical intent-level structure.

Building block	n	Core sequence
Universal mechanisms		
SP-1: Assessment-Driven Refinement	16	`visualise` → `assess` → [*exit* | `generate-knowledge`(spec) → *machine transform*]
SP-2: Knowledge Injection	11	Explicit knowledge/specification as transform input
Transferable building blocks		
SP-3: Feature-then-Cluster	7	`characterise` → `define-unit`(cluster)
SP-4: Project-and-Explore	4	`contextualise`(DR) → `visualise` → `abstract`
SP-5: Residual-Based Refinement	3	`build-model` → `characterise`(residuals) → `visualise` → `assess`
SP-6: Multi-Level Exploration	4	`visualise`(hybrid) → `abstract` → `assess` with level change

TABLE 9: Methodological building blocks identified across the library. Column n: number of workflows containing the pattern. Details in Appendix D.3.

SP-2 (Knowledge Injection) differs from the other building blocks in that it is less a multi-step sub-workflow than a structural principle: prior knowledge or specifications from outside the current workflow enter as first-class typed artifacts rather than implicit assumptions. We include it among building blocks because it carries a typed interface requirement: the consuming transform must accept a `knowledge` or `specification` input. This makes its presence or absence a formally checkable property of any workflow.

The value of expressing building blocks formally lies in three capabilities that informal familiarity cannot provide. First, intent-level classification reveals structural equivalences that domain-specific terminology obscures. For example, dimensionality reduction (SP-4) is classified as a `contextualise` transform constructing a reference space for interpretation, placing it in the same analytical category as calendar grids, geographic maps, and hierarchical layouts. Similarly, residual computation (SP-5) is classified as `characterise` applied to model fit, revealing that model diagnosis and data exploration share identical analytical structure. Clustering (SP-3), while familiar as a technique, is classified as a `define-unit` transform: it constructs new analytical entities (groups). This places it in the same intent category as trajectory segmentation, event parsing, or temporal partitioning - all operations that establish the units on which subsequent analysis operates. Second, typed artifact interfaces enable composition: any upstream sub-workflow producing characterised entities can feed into SP-3 or SP-4 regardless of domain. Third, machine-readable specifications enable computational search and matching over the library.

Detailed descriptions and domain examples for each building block appear in Appendix D.3.

6.4 Iterative Strategy Types

The motifs, meta-structure, and building blocks described above characterise *what* analytical moves workflows perform. A complementary question is *how* they organise repetition: what kinds of iterative strategies appear in the library, and how do they differ in the demands they place on the analyst?

Loop type	n	Workflows
Computational refinement (15 loops)		
Parameter-tuning	5	[21], [23]×2, [24], [31]
Feature/encoding	3	[26]×2, [32]
Model fitting	4	[30]–[32], [34]
Specification-guided	3	[28], [29], [35]
Exploratory investigation (8 loops)		
Diagnostic exploration	3	[34], [36], [37]
Selection-navigation	2	[25], [35]
Strategy exploration	2	[33], [37]
Distribution exploration	1	[26]
Data restructuring (2 loops)		
Simplification	1	[27]
Progressive exclusion	1	[25]
Multi-step analysis cycle (1 loop)		
Nested outer cycle	1	[31]

TABLE 10: Iterative strategy types observed in the library, organised by depth of human intellectual engagement.

Iteration is pervasive: 16 of 17 workflows contain at least one loop, with 26 loops in total across the library. Table 10 organises these into eleven types within three broad categories, distinguished by the depth of human intellectual engagement they require.

The three categories reflect progressively deeper levels of human intellectual engagement. In *computational refinement* loops, the analyst adjusts parameters or method choices while the problem framing and data representation remain fixed; the human role is evaluative (assessing whether results are satisfactory). In *exploratory investigation* loops, the analyst navigates and interprets data interactively, building understanding through successive views; the human role is interpretive (constructing meaning from visual evidence). In *data restructuring* loops, the analyst changes the analytical representation itself—simplifying event sequences, excluding data subsets, or redefining analytical units; the human role is constitutive (reshaping what the workflow operates on).

This gradation has design implications: computational refinement loops require responsive parameter controls and convergence feedback; exploratory loops require flexible navigation and view-coordination facilities; data restructuring loops require tools for the analyst to reformulate the analytical representation—a qualitatively different and more demanding design challenge. Detailed characterisation of each loop type appears in Appendix D.4.

6.5 Summary

Formal representation has revealed structural regularities at four levels of granularity: a shared meta-structure (global workflow organisation), recurring motifs (signatures of analytical organisation), reusable building blocks (composable sub-workflows with typed interfaces), and diverse iterative strategies (loop organisations reflecting different depths of human engagement). While these specific findings are preliminary hypotheses from a purposively selected library, the analysis demonstrates that formal workflow representation creates an analytical infrastructure within which such patterns can be systematically discovered, compared, and refined. The formal language is the enabler; the current findings are its first fruits.

7 Recommendation Support: Comparing Prose and Formal Libraries

The cross-workflow analysis in Section 6 demonstrated what becomes visible once workflows are formally represented. A different question is whether formal representation also makes a practical difference for using a workflow library to design workflows for new problems. To examine this, we conducted a comparative experiment in which the same LLM (Claude Opus 4.6R), addressing the same problems, was supplied with the library in two alternative forms: as the seventeen original research papers (PDFs) and as the seventeen ATWL workflow representations preceded by the language definition.

Our prior expectation was that recommendations of comparable substance would not be obtainable from the original papers. The experiment refuted this expectation: in both formats the agent produced coherent, library-grounded recommendations. The question is therefore not *whether* a formal library is needed but *what specifically the formal representation contributes* and *when each format is preferable*. The remainder of this section reports the experiment and our answers; a detailed comparison appears in Appendix E.

7.1 Experimental Setup

Four fresh agent sessions were run, one per cell of a 2×2 design (library format $\times$ problem). Each session was independent: no agent saw the other sessions' inputs or outputs.

Library formats. In the *prose* condition the agent received an archive of the 17 source papers as PDFs. In the *formal* condition the agent received first a document defining ATWL and then a document containing the 17 ATWL workflow representations. Before being given the user's problem, each agent was asked to analyse the library and identify reusable components.

Problems. *Problem A (bike-sharing)*: design a workflow for analysing public bike-sharing data with spatio-temporal patterns of shortages and overcrowding related to temporal cycles, and develop a relocation strategy. The problem is described conceptually; we did not provide data. *Problem B (research-topic evolution)*: design a workflow for revealing major research topics in IEEE VIS publications (1990–2024) and their evolution, emphasising long-term trends over short fluctuations. Real data is available, so both agents were additionally asked to produce a Jupyter notebook implementing the recommended workflow.

The complete agent inputs, outputs, and our side-by-side analysis are available at https://geoanalytics.net/VAworkflows/experiment. A formal-library agent's natural-language description of the recommended workflow for Problem B is in Appendix G. A more elaborately structured illustrative recommendation for Problem A, produced in an earlier session in which an ATWL-equipped agent was asked to deliver its output as a LaTeX document with explicit phase-to-library cross-references, is in Appendix F (with its flow diagram in Fig. 5).

7.2 Findings

Both formats produced usable recommendations. In all four sessions the agent identified relevant library workflows, decomposed the user's problem into phases, and proposed a coherent end-to-end pipeline. On Problem B the two notebooks converged on essentially the same analytical content: text vectorisation, topic modelling (NMF or BERTopic), document–topic assignment, year-wise aggregation, temporal smoothing, stacked-area visualisation, and a final classification of topics into rising, declining, and stable. The selection of library sources also overlapped substantially across formats. We therefore cannot claim that formal representation is *necessary* for LLM-based recommendation against a moderately sized library.

The two formats differ along four clear dimensions that emerged consistently across both problems.

Explicitness of iteration structure. The formal-library recommendations carried iteration as a first-class object: each loop was named, the specification artifact controlling it was declared (e.g., `clustering_spec`, `threshold_spec`, `model_spec` for Problem A), and the assess–refine condition was stated. The prose-library recommendations described iterative refinement narratively ("adjust thresholds and re-run", "iterate until topic set is stable") without enumerating the loops or naming the parameters updated. This difference propagated into the generated notebooks for Problem B: the formal-library notebook included dedicated assessment cells with steerable Boolean flags (`topics_satisfactory`, `smoothing_satisfactory`) and pre-filled refinement suggestions; the prose-library notebook contained only one comparable checkpoint (a topic-merging map).

Provenance traceability. Both agents cited their sources. The formal-library agent additionally produced *adaptation tables* that linked each phase of the new workflow to one or more library workflows and itemised what was reused and what was changed (e.g., "Phase 1 from [24]: spatial-clustering loop with visual assessment; adapted from trajectory stops to station locations; added capacity aggregation"). The prose-library agent cited papers inline by author and year but did not produce a comparable fragment-level mapping.

Typed data flow between fragments. The formal-library recommendations typed every intermediate artifact (e.g., `place_time_series : entities, internal structure: sequence, embedment: {space: place, time: regular slots}; flow_matrices : feature(places), value structure: matrix`). The prose-library recommendations described the same intermediates in domain language without committing to types. The typed version exposes the data shape of the workflow and makes the composition of fragments from different sources structurally checkable.

Methodological detail. The asymmetry runs in the opposite direction here. Prose papers carry method-level material that ATWL abstracts away by design — parameter ranges, algorithmic variants, validation tricks, exceptions. This material was visible both in the prose-library agent's preparatory analysis (which was longer and more methodologically detailed than the formal-library agent's) and in the prose-library recommendations themselves (which included more specific algorithmic suggestions, e.g., walking-distance thresholds for DBSCAN, vocabulary parameters for TF–IDF). The formal-library agent recovered comparable method choices from its pre-training but had no library-grounded warrant for the specific defaults it picked.

Dimension	Prose-papers library	Formal ATWL library
Form of deliverable	Methodological narrative with inline citations	Formal specification with named transforms, typed artifacts, and declared loops; natural-language version on request
Iteration structure	Described in prose; rarely surfaces as discrete pause points in generated code	First-class: named loops with declared specifications and assess–refine conditions; propagates into generated code as steerable checkpoints
Provenance	Inline citation of source papers by author/year	Fragment-level adaptation tables linking each phase to library workflows and itemising changes
Composition of fragments	Implicit; type compatibility is left to the reader	Typed artifact interfaces make structural compatibility explicit and checkable
Methodological detail	Rich: parameter ranges, algorithmic variants, validation tricks, exceptions	Abstracted away by design; agent supplies method defaults from its pre-training
Scalability with library size	Already at the edge of context capacity at 17 workflows	Roughly an order of magnitude smaller per workflow; remains feasible at much larger scales

TABLE 11: Qualitative contrasts between the two library formats as inputs to LLM-based workflow recommendation, summarising the findings of the four-session experiment.

Table 11 summarises the contrasts qualitatively.

Context-window pressure. The pressure is not hypothetical even at 17 workflows. The archive of 17 PDFs (about 94 MB) already exceeded what could be supplied to the agent intact: the interface reported context utilisation at 149 % and warned that the attachment had been truncated, with possible loss of image content. The session ran successfully, but the agent's view of the library was incomplete. The ATWL library, by contrast, is roughly an order of magnitude smaller per workflow and dense in workflow-relevant content, so it fits comfortably alongside the language definition, the user's problem, and the agent's reasoning. The 17-workflow case is therefore already at the edge of feasibility for the prose format; at 50 or 100 workflows the prose approach would no longer be viable in any current context window, whereas the formal library would remain manageable.

Curation as a persistent artifact. A second consideration concerns the library as an object. Our seventeen-workflow ATWL library is not yet community-curated; we hope it will become so. Independently of curation, the formal representation has the property that its abstractions — intent vocabulary, typed artifacts, loop conventions — are externalised once and reused across all workflows and all sessions. A prose library does not have this property: each agent rederives whatever abstractions it uses from raw text every time. Community curation, if it materialises, would amplify the externalised abstractions; it is not required for them to hold.

7.3 What Formal Representation Contributes

Read together, the findings support a narrower and more defensible claim than the one we initially expected to make.

Strengths of the formal-library approach. ATWL representations gave the agent a shared analytical vocabulary that surfaced consistently in its outputs as named transforms, typed artifacts, explicit loops, and itemised provenance. These elements make recommendations more easily auditable, more readily composable across domain boundaries, and more directly translatable into structured implementations (notebooks with assessment checkpoints, diagrams, or executable pipelines). The compactness of the representation also supports scaling to larger libraries.

Strengths of the prose-papers approach. The original papers carry a depth of methodological detail that no abstraction preserves. When method choices, parameter defaults, or implementation rationale are central to the user's needs, a library of papers is an asset that ATWL alone cannot replace. The prose representation also requires no formalisation effort, removing the bootstrap barrier of an ATWL library.

When each format is preferable. A prose-papers library is preferable when methodological detail is decisive, when the library is small enough to fit in context, when no formalisation effort has been invested, and when the user wants a literature-grounded narrative rather than a machine-checkable specification. A formal ATWL library is preferable when the library is large or expected to grow, when cross-domain structural matching is desired, when provenance must be fine-grained, and when the recommendation will be implemented programmatically or composed with other workflows.

Combining the two formats. The two representations are not exclusive and their complementary strengths suggest a two-stage process that captures both. Stage 1: query the ATWL library to obtain a structured recommendation — a workflow specification with named loops, typed artifacts, and an adaptation table identifying which library workflows informed which phases. The recommendation is library-grounded and structurally complete but methodologically thin. Stage 2: in a separate, focused session, consult only the papers identified in the adaptation table to enrich the recommendation with method-level detail — specific algorithms, parameter defaults, validation procedures. Because the adaptation table typically points to a handful of papers rather than the full library, the second stage stays within feasible context budget even when the prose library as a whole would not. ATWL thereby acts as both the structural skeleton of the recommendation and the index that selects which papers warrant detailed reading.

This recasts the practical contribution of ATWL in the terms our evidence actually supports: not as a precondition for LLM-based workflow design support, which the experiment shows it is not, but as a representation that adds explicit iteration structure, typed composition, fragment-level provenance, and scalable curation to what the prose baseline already enables and that, used as the first stage of a two-stage process, also serves as the index for targeted consultation of the source papers when methodological detail is needed.

8 Discussion

We discuss our primary contribution, its practical value, limitations, and how it could provide a basis for a joint community effort to better characterise the field of VA.

8.1 The Primary Contribution

Sections 6–7 have demonstrated that ATWL and workflow library create an analytical infrastructure (shared vocabulary, comparable structures, machine-readable specifications) that enables turning VA practice into a subject of formal study.

The language embodies ontological choices whose adequacy can be evaluated independently: typing artifacts by analytical role rather than computational format; classifying transforms by intent rather than method; treating human knowledge and specifications as first-class artifacts; and restricting the language to observable, externalised products. Different decisions would yield a different language with different affordances. The evidence that the current design is adequate comes from the fact that all seventeen workflows in the library could be represented without requiring constructs beyond the defined ontology. This is necessary but not sufficient evidence of generality; ATWL is designed to be extended when new workflows motivate additional constructs (Section 8.4).

8.2 Practical Value

The work has practical implications at different levels of readiness.

Available now. For researchers publishing VA workflows, ATWL provides a disciplined documentation format that makes explicit what prose descriptions often leave implicit: the sequence of operations, the types of information produced and consumed, the points of human judgment, and the structure of iteration. Our experience formalising seventeen workflows consistently revealed ambiguities in the original descriptions, suggesting that formalisation itself has value as an analytical discipline. For teaching, the library and cross-workflow analysis offer a structured resource for discussing how the same strategies recur across domains and how design decisions have structural consequences.

Demonstrated at proof-of-concept level. The comparative experiment (Section 7) shows that an LLM can produce usable workflow recommendations from a moderately sized library in either prose or ATWL form; formal representation is therefore not a precondition for LLM-based design support. What the formal library systematically adds is structure that the prose baseline delivers only partially: explicit iteration with declared specifications, typed data flow between fragments that makes their composition structurally checkable, fragment-level adaptation provenance linking each phase of a recommendation to its library source and the changes made, and compactness, which already at seventeen workflows the prose archive strained and which becomes decisive at larger library sizes. The two formats are complementary: ATWL provides the structural skeleton and traceable provenance, and the source papers retain the methodological detail that ATWL abstracts away. We suggested in Section 7.3 a two-stage use that combines them. However, neither these claims about formal representation nor the learnability of ATWL have been evaluated with independent users.

Adoption barriers. Practical adoption faces challenges: learning the intent-vs-method distinction requires initial effort; formalisation benefits from access to an advanced LLM; no dedicated tool support exists beyond the LLM-based agents; and the library's usefulness grows with size, creating a bootstrapping problem for early adopters. We believe these barriers are manageable: stored session logs shorten the learning curve, and formalising a workflow typically requires about an hour of supervised interaction.

8.3 Limitations

Sample size and selection bias. The library contains seventeen workflows chosen for breadth rather than selected randomly. Approximately half of the library draws from the authors' own prior work, which provided the most readily available source of complete workflow descriptions. This concentration may influence the observed regularities; independent contributions to the library would provide a stronger test of generality. Since all workflows come from published VA research, some regularities may reflect shared academic conventions rather than properties of analytical reasoning. Findings may not generalise to industrial practice, unrepresented domains, or paradigms beyond those included.

Language expressiveness. ATWL's initial design was developed and iteratively refined using a small number of workflows (three to four) as test cases; the majority of the seventeen library workflows were formalised after the language design had stabilised. Nevertheless, all workflows were selected and formalised by the same team that designed the language, so an element of circularity remains: the authors' familiarity with the language may have influenced which workflows were selected or how ambiguous cases were resolved. Workflows requiring constructs outside the current ontology, e.g., collaborative multi-analyst processes, real-time streaming, or extensively automated decision-making, may exist but did not appear in this library. We welcome extension proposals motivated by concrete workflows that the current constructs do not adequately capture.

No independent user evaluation. Neither learnability nor recommendation quality has been evaluated with independent users. Formalisations were produced and validated by the authors; recommendation experiments were also assessed by the authors. Proper evaluation would test whether unfamiliar users can produce (with LLM assistance) consistent formalisations and whether recommendations are perceived as useful.

Subjectivity in formalisation. Formalisation requires interpretive judgment: deciding workflow boundaries, assigning intent labels, distinguishing `assess` from `abstract`, and setting loop termination criteria. The multi-agent review process and fresh-agent validation reduce but do not eliminate this subjectivity. Inter-annotator agreement studies would be needed to quantify variability.

Potential predisposition in cross-workflow analysis. The cross-workflow analysis was conducted by the same

team that designed the language and formalised the workflows. Although the patterns found (meta-structure, subpatterns, equivalences) are not consequences of the language definition, we cannot fully exclude that the team's shared analytical perspective influenced both how workflows were formalised and which regularities were sought. A stronger test would analyse workflows formalised independently by researchers not involved in ATWL's development.

LLM dependence. The methodology was validated with Claude Opus and GPT-5, but reproducibility with other models, future versions, or smaller open-source models is uncertain. The stored session logs partially mitigate this by encoding accumulated expertise independently of any specific model.

8.4 Community Adoption and Library Growth

The true potential of formal workflow representation can only be realised through sustained community engagement. We invite VA researchers to adopt ATWL for documenting workflows in their publications and to contribute representations to a growing public library.

For individual researchers, formalisation imposes disciplined reflection on analytical choices and may reveal design decisions not fully articulated in prose. A formal representation can accompany a publication as supplementary material, paired with an automatically generated flow diagram. For the community, a growing library would allow the hypotheses from Section 6 to be tested and refined, and would substantially improve recommendation quality by expanding the space of available building blocks and archetypes.

Open resources. To lower the adoption barrier, we provide a public repository (https://geoanalytics.net/VAworkflows/index.html) containing the ATWL language definition, the workflow library, and stored LLM session logs that function as pre-trained agents for four tasks: `extractor` (workflow extraction and formalisation), `reviewer` (systematic review of representations), `diagrammer` (flow diagram generation), and `recommender` (workflow recommendations for new problems). These logs encode accumulated expertise and, when loaded into an advanced LLM, prime the model with ATWL knowledge and conventions.

The recommended workflow for formalising a new paper or technical report is: (1) load the extractor session and source document into an LLM to obtain an initial representation; (2) load the reviewer session with the paper and extractor output for critical assessment; (3) return feedback to the extractor for correction; (4) optionally generate a flow diagram via the diagrammer. Users may also query the recommender by describing an analytical task and receiving grounded suggestions from the library.

Note on diagram generation. LLM agents reliably produce correct diagram *structure* (nodes, edges, groupings, loop boundaries) from ATWL specifications. However, spatial layout and aesthetic quality depend on the rendering approach. We experimented with two strategies: generating Mermaid code for browser-based rendering (faster, but with limited layout control) and generating LaTeX specifications using the `tikz` package (more laborious, but with full control over positioning). Both approaches may require human post-editing for complex workflows. Users should expect the diagrammer to produce a structurally correct starting point that may need manual layout refinement rather than a publication-ready figure.

We welcome contributions of new workflow representations, feedback on the language, and proposals for extensions motivated by workflows that the current constructs do not adequately capture.

9 Conclusion

We have presented ATWL, a formal language for representing visual analytics workflows, together with a library of seventeen formalised workflows. The language provides the VA community with a shared vocabulary in which analytical processes become comparable, decomposable, and machine-processable regardless of application domains. Cross-workflow analysis of the library reveals structural regularities - a common meta-structure, recurring motifs, reusable building blocks with typed interfaces, diverse iterative strategies, and cross-domain equivalences - that remain invisible when the same workflows are read as prose. A controlled experiment shows that the same library, supplied to an LLM either as research papers or as ATWL representations, supports useful workflow design recommendations in both forms, but that the formal representation systematically adds explicit iteration structure, typed composition, fragment-level provenance, and compactness that supports larger libraries than prose can fit in context. The true potential of this approach, however, lies not in what seventeen workflows reveal but in what a community-maintained library could enable: a cumulative formal science of human–computer analytical processes in which workflow structures are catalogued, compared, composed, and progressively refined across the field.

References

[1] M. Sedlmair, M. Meyer, and T. Munzner, "Design study methodology: Reflections from the trenches and the stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.

[2] J. van Wijk, "The value of visualization," in *VIS 05. IEEE Visualization, 2005.*, 2005, pp. 79–86.

[3] N. Andrienko, T. Lammarsch, G. Andrienko, G. Fuchs, D. Keim, S. Miksch, and A. Rind, "Viewing visual analytics as model building," *Computer Graphics Forum*, vol. 37, no. 6, pp. 275–299, 2018.

[4] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim, "Knowledge generation model for visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1604–1613, 2014.

[5] C. He, N. Elmqvist, A. Bellucci, M. Munshi, and G. Jacucci, "Systemization of knowledge (sok): Visualization insight – two decades of research, practice, and future directions," *ACM Comput. Surv.*, vol. 58, no. 9, Feb. 2026.

[6] M. Meyer, M. Sedlmair, P. S. Quinan, and T. Munzner, "The nested blocks and guidelines model," *Information Visualization*, vol. 12, no. 3–4, pp. 262–275, 2013.

[7] S. McKenna, D. Mazur, J. Agutter, and M. Meyer, "Design activity framework for visualization design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2191–2200, 2014.

[8] M. Chen and D. S. Ebert, "An ontological framework for supporting the design and evaluation of visual analytics systems," *Computer Graphics Forum*, vol. 38, no. 3, pp. 131–144, 2019.

[9] Y. Wu, S. Gao, S. Zhang, X. Dou, X. Wang, and Q. Li, "From requirement to solution: Unveiling problem-driven design patterns in visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, pp. 1–18, 02 2025.

[10] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, and T. C. Community, "Methods included: standardizing computational reuse and portability with the common workflow language," *Commun. ACM*, vol. 65, no. 6, p. 54–63, May 2022.

[11] Object Management Group, *Business Process Model and Notation (BPMN), Version 2.0*, Object Management Group Standard, 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/

[12] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo, "VisTrails: visualization meets data management," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, 2006, p. 745–747.

[13] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[14] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME – the Konstanz information miner: version 2.0 and beyond," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, p. 26–31, Nov. 2009.

[15] D. Sacha, M. Kraus, D. A. Keim, and M. Chen, "Vis4ml: An ontology for visual analytics assisted machine learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, p. 385–395, Jan. 2019.

[16] P. Beaucamp, H. Maylor, and M. Chen, "Information-theoretic cost–benefit analysis of hybrid decision workflows in finance," *Entropy*, vol. 27, no. 8, 2025.

[17] T. Munzner, "A nested model for visualization design and validation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 921–928, 2009.

[18] L. Moreau and P. Missier, "PROV-DM: The PROV data model," W3C, W3C Recommendation, Apr. 2013, https://www.w3.org/TR/2013/REC-prov-dm-20130430/.

[19] S. B. Davidson and J. Freire, "Provenance and scientific workflows: challenges and opportunities," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008, p. 1345–1350.

[20] M. Chen and A. Golan, "What may visualization processes optimize?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 12, p. 2619–2632, Dec. 2016.

[21] J. J. van Wijk and E. R. van Selow, "Cluster and calendar based visualization of time series data," in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '99)*, 1999, pp. 4–9.

[22] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk, "Reducing snapshots to points: A visual analytics approach to dynamic network exploration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, p. 1–10, Jan. 2016.

[23] T. von Landesberger, F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, and A. Kerren, "Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 11–20, 2016.

[24] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. 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)*, 2011, pp. 161–170.

[25] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko, "Visually-driven analysis of movement data by progressive clustering," *Information Visualization*, vol. 7, pp. 225–239, 07 2008.

[26] N. Andrienko, G. Andrienko, and G. Shirato, "Episodes and topics in multivariate temporal data," *Computer Graphics Forum*, vol. 42, no. 6, p. e14926, 2023.

[27] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman, "Temporal event sequence simplification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2227–2236, 2013.

[28] F. Du, C. Plaisant, N. Spring, and B. Shneiderman, "Eventaction: Visual analytics for temporal event sequence recommendation," in *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2016, pp. 61–70.

[29] J. Choo, C. Lee, C. K. Reddy, and H. Park, "UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 1992–2001, 2013.

[30] 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, 2013.

[31] N. Andrienko and G. Andrienko, "A visual analytics framework for spatio-temporal analysis and modelling," *Data Mining and Knowledge Discovery*, vol. 27, no. 1, pp. 55–83, Jul 2013.

[32] 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," *IEEE Computer Graphics and Applications*, vol. 44, no. 3, pp. 14–29, 2024.

[33] D. Cashman, S. R. Humayoun, F. Heimerl, K. Park, S. Das, J. Thompson, B. Saket, A. Mosca, J. Stasko, A. Endert, M. Gleicher, and R. Chang, "A user-based visual analytics workflow for exploratory model analysis," *Computer Graphics Forum*, vol. 38, no. 3, pp. 185–199, 2019.

[34] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini, "A workflow for visual diagnostics of binary classifiers using instance-level explanations," in *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017, pp. 162–172.

[35] 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*, vol. 41, no. 6, pp. 302–315, 2022.

[36] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Vigéas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in TensorFlow," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 1–12, 2018.

[37] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. B. Viégas, and J. Wilson, "The what-if tool: Interactive probing of machine learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 56–65, 2020.

[38] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual analytics: Definition, process, and challenges," *Lecture Notes in Computer Science*, 03 2008.

Appendix to:

ATWL: A Formal Language for Representing, Comparing, and Reusing Visual Analytics Workflows

Natalia Andrienko, Gennady Andrienko, Jürgen Bernard, and Michael Sedlmair

APPENDIX A
ATWL SYNTAX REFERENCE

This section provides a concise syntax reference for the **Artifact–Transform Workflow Language (ATWL)**, a declarative language for representing visual analytics workflows as structured transformations of artifacts. Angle brackets (< >) denote placeholders; the # character introduces comments.

A.1 Workflow Declaration

```
workflow <workflow-ID>
  template: <intent> → <intent> → ...          #
     optional
  description: "<text>"
      # optional
```

- `template` summarises the main analytic stages using the same intent vocabulary as transforms (Section A.3.1). Parenthetical qualifiers (e.g. `(similarity-based)`) are informal annotations, not new intents.
- Iteration may be indicated with `loop(<intent> -> <intent> -> ...)`. Loop notation in templates is optional and used only when the iterative structure is a key characteristic of the analytical method.
- Artifacts and transforms may be declared in any order that aids readability.

A.2 Artifacts

ATWL defines eight artifact types. Every artifact is declared with a unique identifier and its type. The field `origin: given` marks *exogenous* artifacts not produced by any transform in the workflow; it is omitted for artifacts that appear as transform outputs. Artifact type declarations may include references to other artifacts in parentheses, e.g. `feature(D1)`.

A.2.1 Entities

A collection of identifiable analytical objects of the same kind, each treated as a single (possibly structured) piece of data for the purposes of analysis.

```
artifact <ID> : entities
  origin: given                                #
    only for exogenous artifacts
  internal structure: <structure-type>
  embedment: <embedment-type or {...}>    #
    omitted for single entity
  features:
    - id: <feature_id>

      value structure: <structure>
      value type: <type>                       #
    optional
      description: "<text>"
  description: "<text>"
```

Internal structure. Describes how components inside each entity are organised.

Embedment. Describes the shared environment(s) in which the entities reside and how they are related to each other. Omitted when the artifact represents a single entity. Values may be single or combined in set notation (e.g. `{set, time}`).

Features (inside entities). Each feature declares a **value structure** and, optionally, a **value type**. The same scheme applies to standalone feature artifacts (Section A.2.2).

When atomic components are of mixed types, set notation is used (e.g. `{numeric, temporal}`). The value type may be omitted when the mixture is complex or the types are evident from context.

A.2.2 Feature

Explicit descriptors of properties of entities or relationships between them. Typically produced by `characterise` transforms.

```
artifact <ID> : feature(<artifact-ID>)
  value structure: <structure>
  value type: <type>                           #
    optional
  representation form: "<text>"          #
    optional
  description: "<text>"
```

`<artifact-ID>` refers to the artifact described by the feature. `representation form` clarifies encoding for complex features (e.g. `"similarity matrix"`, `"k-NN graph"`).

A.2.3 Arrangement

An arrangement organises entities in a context, defining how they are positioned within a reference structure. It does not create new entities or values.

Type	Description	Category
`elementary`	Indivisible unit in workflow	Atomic
`group/cluster`	Unordered collection	Container
`episode`	Bounded time interval; temporal units	Container
`region`	Bounded spatial extent; spatial units	Container
`sequence`	Components in linear order	Relational
`formation`	General relational structure (network)	Relational

TABLE 12: Types of internal structure for entities.

Embedment	Description
`set`	Unordered collection
`sequence`	Ordered positions (no metric)
`time`	Temporal axis (abs./rel.)
`space`	Spatial reference (coords/geom)
`relational`	Network or hierarchical structure

TABLE 13: Embedment types for entities.

Value structure — overall organisation	
`atomic`	Single value per entity
`list`	Enumeration of values
`vector`	Fixed-length array
`matrix`	2D array of values
`relational`	Irregular structure (graph/tree)
Value type (optional) — atomic components	
`numeric`	Quantitative counts/measures
`ordinal`	Ordered values, no metric
`categorical`	Unordered discrete categories
`temporal`	Time points or durations
`spatial`	Coordinates or geometries
`text`	Textual content
`reference`	ID pointing to other artifacts

TABLE 14: Value structure and value type for features.

```
artifact <ID> : arrangement(<entities-ID>)
  context: <entities-ID>
  principle: "<text>"
  description: "<text>"
```

- `<entities-ID>`: the entities whose members are arranged.
- `context`: an entities artifact acting as the reference structure (e.g. calendar days, map regions, a projection layout).
- `principle`: how positions are determined (e.g. `"calendar(year, month, weekday)"`, `"2D projection based on similarity"`).

A.2.4 Visualisation

An external visual representation for human perception.

```
artifact <ID> : visualisation(<artifact-IDs>)
  layout: "<text>"
  form: "<text>"
  encoding: "<text>"                    #
    optional but recommended
  description: "<text>"
```

- `<artifact-IDs>`: one or more artifacts of any type.
- `layout`: structure of the visual space (e.g. `"calendar grid"`, `"2D projection"`).
- `form`: visual mark type (e.g. `"coloured marks"`, `"node-link diagram"`).
- `encoding`: how features and arrangements map to visual attributes.

A.2.5 Pattern

An abstract regularity or structure identified in data, features, arrangements, visualisations, or models.

```
artifact <ID> : pattern(<artifact-IDs>)
  representation form: "<text>"
  description: "<text>"
```

`representation form` describes how the pattern is represented (e.g. `"textual descriptions"`, `"ranked list of motifs"`, `"rules"`).

A.2.6 Model

A formal or computational representation used for prediction, simulation, or explanation.

```
artifact <ID> : model(<artifact-IDs>)
  model type: "<text>"
  representation form: "<text>"         #
    optional
  description: "<text>"
```

- `model type`: high-level category (e.g. `"classifier"`, `"topic model"`, `"simulation model"`).
- `representation form`: internal parametric or structural form (e.g. `"decision tree"`, `"neural network weights"`).

A.2.7 Knowledge

Explicitly formulated knowledge, either *derived* in the analysis (insights, explanations, rules) or *injected* during the analysis (domain expertise, constraints, feedback).

```
artifact <ID> : knowledge(<artifact-IDs>)
  origin: given                                  # for
    injected; omitted for derived
  representation form: "<text>"
  description: "<text>"
```

`<artifact-IDs>` reference the artifacts this knowledge is based on or applies to. `representation form` examples: `"statements"`, `"if-then rules"`, `"labels"`, `"quality judgment"`.

Note: `assess` transforms produce *evaluative* knowledge (quality judgments, adequacy decisions), while `generate-knowledge` transforms produce *substantive* knowledge (domain insights, conclusions).

A.2.8 Specification

An explicit description of parameters, settings, constraints, or method choices that determine how transforms are executed.

```
artifact <ID> : specification
  origin: given                                  # for
    injected; omitted for derived
  representation form: "<text>"
  description: "<text>"
```

`representation form` examples: `"parameter settings"`, `"method choice"`, `"constraints"`, `"configuration schema"`.

Specifications are typically consumed by transforms whose behaviour depends on configurable choices. They may be given (exogenous) or derived—most commonly by `generate-knowledge` or `assess` transforms.

A.3 Transforms

Each transform consumes one or more input artifacts and produces one or more output artifacts.

```
transform <ID> :
  intent: <generic-intent>
  manner: "<text>"                    #
    optional specialisation
  input: <artifact-IDs>
  output: <artifact-IDs>
  actor: human | machine | hybrid
  description: "<text>"
```

- `intent`: one of the generic intents in Table 15.
- `manner`: optional free-text specialisation of intent (see Table 16).
- `input`/`output`: comma-separated artifact IDs.
- `actor`: `human` (analyst only), `machine` (computation only), or `hybrid` (human–machine collaboration).

A.3.1 Generic Intents

Intent	Purpose	Typical outputs
`define-unit`	Create/redefine units (extract, group, aggregate)	entities (+ feature)
`characterise`	Compute/transform features	feature
`contextualise`	Place entities in context	arrangement
`visualise`	Create visual reps.	visualisation
`abstract`	Derive pattern-s/structures	pattern (+ feature)
`build-model`	Construct/refine models	model
`generate-knowledge`	Formulate knowledge	knowl./spec.
`assess`	Evaluate quality/adequacy	knowl. (+ spec.)

TABLE 15: Generic transform intents.

A.3.2 Specialisations via Manner

The `manner` field is not part of the core type system but is recommended for consistency and tool interoperability. Table 16 gives illustrative (extensible) values.

Intent	Example manner values
`define-unit`	extract, filter, partitioning, cluster, group, merge
`characterise`	summarise, aggregate, profile, project, encode, relate
`contextualise`	calendar-, map-, or projection-based; time-alignment
`visualise`	line-graph, coloured-marks, node-link, matrix-plot
`abstract`	pattern-mining, salient-groups, interpretation
`build-model`	train-classifier, fit-topic-model, calibrate
`generate-knowledge`	formulate-statements, derive-rules, write-summary
`assess`	evaluate-quality, assess-performance, judge-adequacy

TABLE 16: Illustrative `manner` values by intent.

A.4 Control Structures

Control structures describe analytic logic, not executable control.

A.4.1 Loop

```
loop <ID>:
  purpose: "<text>"
  until: "<qualitative stopping condition>"
  body:
    <transforms, artifact declarations,
    conditionals, assignments>
end loop <ID>
```

The `until` field states a qualitative stopping condition. Two styles of termination are supported:

- **Explicit assessment.** The loop body contains an `assess` transform followed by a conditional with `exit loop <ID>` in one branch.
- **Implicit termination.** No explicit assessment or conditional exit appears; the `until` condition is monitored informally by the analyst.

A.4.2 Conditional

```
if <condition>:
  then:
    <transforms and artifacts>
  else:
    <transforms and artifacts>
```

Conditions refer to artifact properties or human judgements. Use `exit loop <ID>` in a branch to leave an enclosing loop.

A.4.3 Assignment

```
assign:
  <artifact-ID>  := <artifact-ID'>
  <artifact-ID2> := <artifact-ID2'>
```

Assignments bind an artifact identifier to a new version. They appear in two contexts:

- **Before a loop**, to initialise an identifier that will be reassigned during iteration.
- **Inside a loop body**, to express iterative update of an artifact.

A.5 Workflow Validity and Conventions

A well-formed ATWL workflow should satisfy:

1) All artifact identifiers are unique within the workflow.
2) Every transform input references a declared artifact.
3) Artifacts with `origin: given` are never transform outputs.
4) The artifact dependency graph is acyclic (except for explicit `assign` statements in loops).

Additional conventions:

- Exogenous artifacts (`origin: given`) are typically declared near the top or near their first use.
- Lists of artifact IDs are comma-separated.
- Artifact references appear in parentheses after the type keyword, e.g. `feature(D1)`, `arrangement(D_events)`.
- Each transform carries exactly one `intent`, even when it produces multiple artifact types; the intent reflects the primary analytic purpose.

APPENDIX B
EXAMPLE: CLUSTER-CALENDAR WORKFLOW REPRESENTED IN ATWL

Source [21]: 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.

A flow diagram representing the Cluster-Calendar workflow is shown in Fig.4.

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"
```

APPENDIX C
WORKFLOW LIBRARY CONSTRUCTION

The library was built through a collaborative process combining LLM agents and human expert oversight. For each published paper, a *Formalising Agent* extracted the analytical workflow from the paper's prose, figures, and algorithmic descriptions, producing an ATWL specification. A *Reviewing Agent* then verified syntactic correctness, ontological consistency, and logical completeness of the result, generating structured feedback that was passed back to the Formaliser for revision. This two-agent cycle was repeated until the specification satisfied all validity constraints (Section 4.4), with a human expert providing corrective prompts when the agents' own iteration did not resolve an issue. For transparency and replication, a detailed description of the process is available at https://geoanalytics.net/VAworkflows/LLM_experiments_extract_review.pdf.

After all individual workflows had been extracted, a systematic audit of the full library was conducted collaboratively between the human expert and an LLM assistant (a new instance of an LLM-based agent). For each workflow, the LLM identified remaining issues and proposed corrections; the expert evaluated each proposal, accepting, modifying, or rejecting it. In several cases, disagreements between the expert and the LLM led to productive discussions among co-authors that resulted in refinements of the ATWL language definition itself; disagreements were resolved through group discussion until consensus was reached. The full audit report is available at https://geoanalytics.net/VAworkflows/review_16_workflows.pdf. The patterns of errors observed across the library informed the development of comprehensive procedural instructions for both the extraction and review agents, encoding the expertise accumulated during the process (see https://geoanalytics.net/VAworkflows/extraction_instructions.pdf and https://geoanalytics.net/VAworkflows/reviewing_instructions.pdf). These instructions were validated in a fresh extraction experiment using two new LLM instances that had not participated in any preceding work, confirming that the documented procedures are sufficient to guide new agents to produce correct ATWL representations with minimal human intervention. We invite readers to conduct their own experiments using the materials and instructions provided at the URL https://geoanalytics.net/VAworkflows/.

All seventeen workflows were successfully formalised through this process, demonstrating that ATWL's vocabulary is sufficiently structured for automated reasoning yet close enough to natural analytical descriptions to be derived from them by current language models, provided that human expertise guides the overall process, validates the results, and feeds corrective knowledge back into both the language definition and the agent instructions.

Importantly, although human involvement remains essential, the overall formalisation effort is substantially reduced compared to purely manual work: the expert's role shifts from writing ATWL specifications from scratch to evaluating and correcting machine-generated proposals. The latter task is considerably faster and less cognitively demanding, particularly for complex multi-loop workflows.

APPENDIX D
DETAILED CROSS-WORKFLOW ANALYSIS

This appendix provides detailed elaboration of the cross-workflow analysis findings summarised in Section 6.

D.1 Recurrent and Paradigm-Specific Analytical Patterns

A first question is which types of transforms and artifact types recur broadly across VA workflows and which are specific to particular analytical paradigms.

Transforms common to all library workflows. Three transform intents appear in every workflow in the library: creating visual representations (`visualise`), identifying regularities through abstraction (`abstract`), and formulating explicit knowledge (`generate-knowledge`). Almost all workflows also include computing features that describe analytical entities (`characterise`) and assessing results of preceding transforms (`assess`). This consistent co-occurrence across seventeen heterogeneous workflows provides empirical support for the central tenet of established VA process models [3], [4], [38]: that the interplay of visual encoding, pattern perception, quality assessment, and knowledge generation constitutes the core of human–machine analytical reasoning. What the formal analysis adds is precision: these are not merely conceptually present

but identifiable as specific typed transforms with defined inputs, outputs, and actor assignments across all seventeen workflows.

Paradigm-discriminating transforms. Three less frequent transform intents discriminate between different analytical paradigms. First, `define-unit` (creation or redefinition of analytical entities) is absent from workflows that examine pre-existing models (e.g., [33], [37]) rather than constructing objects from raw data. This observation points to a paradigm in which the purpose is *investigation* of pre-structured objects rather than *construction* of new structures. Second, explicit contextualisation (`contextualise`), i.e., constructing a reference space such as a calendar grid [21], [23], a 2D projection ([22], [29]), or a relative time axis [27], appears in roughly half the workflows. The remainder operate in an implicitly given context (a geographic map, a feature space, a model interface). This division reflects a methodological choice: whether the reference context for interpretation must itself be designed as part of the analysis. Third, the construction of formal predictive or explanatory models (`build-model`) appears in a few workflows [30]–[34], suggesting a boundary between pattern identification and model building that only a subset of VA workflows cross.

Artifact types. Five artifact types - `entities`, `features`, `visualisations`, `patterns`, and `knowledge` - appear in all seventeen library workflows. Two further types play important but paradigm-dependent roles. Explicit `specifications` (parameters, constraints, method choices) are used in most workflows. Their occasional absence highlights alternative mechanisms by which human judgment can steer computation: through direct knowledge injection rather than parameterisation [32], or through interactive navigation rather than configured computation [36]. Formal `models` appear both as constructed outputs [30]–[34] and as given inputs subject to exploration [35]–[37]. This duality reflects the growing importance of model understanding as an analytical activity.

Entity structures. The diversity of entity structures across the library, including indivisible units, groups and clusters, time-bounded episodes, ordered sequences, spatially bounded regions, and entities with internal relational structure (networks, hierarchies, dataflow graphs), demonstrates that the VA workflows we examined operate on structured, composite analytical objects, not merely on flat records or simple measurements. This structural richness of analytical entities is a characteristic of VA that any formal representation must accommodate.

D.2 Detailed Meta-Structure Analysis

This subsection elaborates the five-stage meta-structure presented in Section 6.1.

Two distinct entry modes. Stage 1 (*Representation Construction*) takes two forms not differentiated by existing conceptual models: in *data-centric* workflows (12/17), raw inputs are converted into suitable analytical units with computed features; in *model-understanding* workflows (e.g., [33]–[37]), summary statistics are computed for existing objects and presented for exploration without data restructuring. These forms reflect two different entry points into analytical reasoning: building the problem representation versus examining a pre-existing one. This structural distinction has design implications for tool support.

Contextualisation as an identifiable design decision. Optional Stage 2 (*Contextualisation*) constructs a reference space that situates analytical units for investigation. Eight workflows that include this stage span a wide range of strategies: calendar grids [21], 2D projections [22], [29], [32], [35], relative time axes [27], contextual arrangements [26], and hierarchical layouts [36]. The workflows that omit explicit contextualisation operate in an implicitly given context. Existing conceptual models do not isolate contextualisation as a distinct analytical stage; its identification here highlights that the design of an appropriate *analytical space* is itself a significant methodological decision, present in roughly half of the workflows studied.

Specific transform–actor assignments across stages. The meta-structure reveals a systematic shift in agency across stages: Stage 1 is predominantly machine-executed, Stage 3 is hybrid, and Stages 4–5 are predominantly human-driven. This progression from computational construction through collaborative refinement to interpretive synthesis operationalises the general principles of human–machine collaboration. The specific mapping of typed transforms to actor roles at each stage is new and could inform the design of VA systems.

The iterative core and its exceptions. Stage 3 (*Iterative Analysis*) is the analytical core, present in sixteen workflows. It is the stage at which human judgment is most intensively engaged, steering computational processes through visual assessment until results meet relevant criteria. The single workflow without iteration [22] compensates with a richer linear sequence of transforms, demonstrating that iterative human–machine dialogue is a dominant, though not the only possible, mode of VA reasoning within this library.

Knowledge generation within and beyond loops. Stages 4 and 5 (*Pattern Recognition* and *Knowledge Synthesis*) appear in all seventeen workflows. In 12 workflows, they appear as terminal stages. In five workflows [26], [27], [31], [35], [37], pattern recognition or knowledge generation also occurs *within* loops, blurring the boundary between iterative refinement and interpretive synthesis. This is consistent with the progressive nature of insight formation described by Sacha et al. [4], but formal representation makes it possible to identify precisely *which* workflows exhibit this feature and *where* in their structure knowledge generation takes place.

D.3 Detailed Building Block Descriptions and Cross-Domain Examples

This subsection provides detailed descriptions and domain examples for the six building blocks summarised in Table 9, followed by an analysis of cross-domain structural equivalences that these building blocks reveal.

Universal mechanisms

SP-1: Assessment-Driven Refinement. All sixteen workflows containing a loop share the core sequence `visualise` → `assess` → [*exit* | `generate-knowledge`(spec) → *machine transform*]. In every case the exit condition involves a *qualitative* human judgment rather than an automated convergence

criterion. The consistency of this pattern across the library supports the view that visual assessment by a human analyst is a central feedback mechanism in VA, one in which computational output is evaluated against interpretive criteria that cannot be fully formalised. In the majority of cases (13/17 workflows), the feedback channel operates through specification artifacts: the analyst's evaluative judgment is externalised as a typed `specification` that parameterises the next computational transform, separating the act of judgment from its computational execution.

SP-2: Knowledge Injection. Eleven workflows [21], [23]–[28], [31], [34], [35], [37] treat analyst-provided inputs—such as domain knowledge, study questions, initial parameter settings, or pre-trained models—as explicit typed inputs to transforms. SP-2 is less a multi-step sub-workflow than a structural principle: prior knowledge or specifications from outside the current workflow enter as first-class artifacts rather than implicit assumptions. We include it among building blocks because it carries a typed interface requirement—the consuming transform must accept a `knowledge` or `specification` input—making its presence or absence a formally checkable property of any workflow. This finding highlights that the VA workflows in our library are rarely assumption-free: prior expertise shapes the analytical process from the beginning, and making these assumptions explicit is essential for reproducibility and critical evaluation.

Transferable building blocks

SP-3: Feature-then-Cluster. Seven workflows [21], [23], [24], [29], [31], [32], [35] follow the sequence `characterise` →`define-unit` (cluster). That clustering requires prior feature definition is well known; what the formal representation makes explicit is that clustering serves as a `define-unit` transform: it constructs new analytical entities (groups). This classification places clustering in the same intent category as trajectory segmentation, event parsing, or temporal partitioning - all transforms that establish the units on which subsequent analysis operates. The typed interface (`entities, features` → `entities`(grouped)) makes the dependency explicit and the pattern composable with any downstream sub-workflow that consumes grouped entities.

SP-4: Project-and-Explore. Four workflows [22], [29], [32], [35] employ the sequence `contextualise`(dimensionality reduction) → `visualise`(scatterplot) → `abstract`, spanning dynamic network analysis, text mining, movement analysis, and ML model interpretation. The cross-domain recurrence of DR projection is itself unsurprising—it is a ubiquitous exploration technique. What the formal representation reveals is a conceptual equivalence: dimensionality reduction serves the same analytical role as constructing a calendar grid [21], a relative time axis [27], or a hierarchical layout [36] — namely, establishing a reference space (`contextualise`) within which entities become visually interpretable. This classification makes substitution questions well-formed: if a workflow contextualises entities through projection, could an alternative contextualisation strategy (e.g., a domain-specific spatial arrangement) serve the same analytical purpose? The typed interface makes such questions answerable.

SP-5: Residual-Based Refinement. Three model-building workflows [30]–[32] share the sequence `build-model` → `characterise`(residuals) → `visualise`(residual displays) → `assess`. That model-building workflows include residual analysis is expected: it is a standard diagnostic practice. What the formal representation makes explicit is the analytical mechanism: computing residuals is classified as a `characterise` transform that converts model inadequacy into observable features of analytical entities, which then enter the same `visualise` → `assess` cycle used throughout non-modelling workflows. Model diagnosis and data exploration thus share identical analytical structure in ATWL; they differ only in what is being characterised. This structural equivalence suggests that any visual assessment strategy effective for feature exploration could, in principle, be adapted for model diagnosis, and vice versa—a design implication that becomes visible only when both activities are expressed in the same typed vocabulary.

SP-6: Multi-Level Exploration. Four model-understanding workflows [34]–[37] employ loops in which the analyst navigates between linked views at different abstraction levels (e.g., outcome → feature → instance in [34]; group → component → node in [35], [36]). This shared strategy of *progressive deepening* through interactive view reconfiguration defines a distinctive mode of analytical reasoning in which data remain fixed and understanding is built by systematically varying the analyst's viewpoint. Despite targeting binary classifiers, deep learning architectures, and general ML models, all four employ the same structural pattern.

D.4 Detailed Loop Analysis and Multi-Loop Structures

This subsection elaborates the iterative strategy types summarised in Section 6.4 and describes multi-loop progressive structures observed across the library.

Computational refinement (15 loops). The majority of iterative strategies involve human-guided adjustment of a computational process while differing in the depth of human engagement. *Parameter-tuning* loops(5) involve the least demanding form of engagement: the method is fixed and only its configuration varies [21], [23], [24], [31]. *Feature/encoding refinement* loops (3) are qualitatively more demanding: the analyst restructures the problem representation itself by refining symbolic encodings [26] or engineering new features [32]. *Model-fitting* loops (4) iterate between model construction and quality assessment, typically through residual analysis [30], [31] or error examination [32], [34]. *Specification-guided* loops (3) represent the most direct form of human-to-machine knowledge transfer: the analyst's decision is externalised as a specification artifact that directly controls the next computation. Examples are semi-supervised matrix factorisation [29], refining an action plan [28], or adjusting decision boundaries [35].

This gradation from parameter adjustment through problem restructuring to knowledge externalisation suggests progressively deeper levels of human intellectual engagement with the analytical process. If this gradation holds across a larger corpus, it has implications for system design: parameter-tuning loops require responsive controls

and convergence feedback, while specification-guided loops require facilities for the analyst to articulate complex analytical decisions as structured artifacts.

Exploratory investigation (8 loops). A second category comprises loops in which the data remain fixed and the analyst builds understanding through interactive navigation. *Diagnostic exploration* loops [34], [36], [37] involve navigating between linked views at different abstraction levels. *Selection-navigation* loops [25], [35] involve drilling down through candidate spaces. *Strategy exploration* loops [33], [37] test alternative analytical approaches. The *distribution exploration* loop in [26] investigates how patterns distribute across contextual dimensions. These loops are characteristic of analytical processes in which the goal is *understanding* rather than *optimisation*.

Data restructuring (2 loops). Two distinctive loops involve reorganisation of analytical entities. The *simplification* loop in EventFlow [27] iteratively reduces event sequence complexity until a display of acceptable visual density is achieved. The *progressive exclusion* loop in [25] removes dense clusters at each iteration and re-clusters the remainder at lower sensitivity. Both represent a strategy of iteratively *simplifying* the problem rather than refining a solution, demonstrating a different analytical logic than in the earlier discussed categories.

Multi-step analysis cycle. In the spatio-temporal workflow [31], an outer loop encompasses an entire analysis cycle consisting of grouping, model fitting, and residual evaluation with inner refinement loops at each stage. This represents the most complex form of analytical organisation observed in the library, requiring the analyst to manage multiple interdependent decisions.

Multi-loop progressive structures. Eight workflows employ multiple loops, and comparison of their architectures reveals a recurring pattern: *progressive advancement of abstraction level* (Table 17).

Workflow	Loop progression
[23]	Spatial aggregation → Temporal clustering
[25]	Destination clustering → Route analysis
[26]	Symbol encoding → Topic discovery → Distribution exploration
[32]	Feature engineering → Model validation
[35]	Component selection → Boundary refinement
[37]	Hypothetical probing → Fairness strategy
[31]†	Analysis cycle ⊃ {Grouping, Model fitting}
[34]†	Model improvement ⊃ Diagnostic exploration

TABLE 17: Progressive abstraction in multi-loop workflows. † = nested (non-sequential) structure.

In six workflows, successive loops raise the abstraction level of representation. The most elaborate is [26]: three loops transform raw temporal values into symbolic patterns (Loop1), combine patterns into multi-attribute topics (Loop2), and explore topic distributions across contextual dimensions (Loop3). Patterns produced at one level serve as inputs at the next, creating a chain of progressive abstraction. In [32], the two loops differ in purpose: the first refines the *problem representation* (feature space) while the second validates the *solution* (classifier). Such distinctions are easily obscured in prose descriptions that label conceptually different processes as "iterative refinement"; the formal representation makes them explicit.

In [31], an outer analysis loop contains two inner loops (grouping refinement, model fitting) and a residual evaluation step that determines whether to exit or re-enter with adjusted parameters. In [34], an outer model-improvement loop contains an inner diagnostic exploration loop. These nested structures represent *multi-granularity reasoning*: fine-grained tuning or exploration within coarse-grained conceptual refinement, each with its own assessment criteria and termination conditions.

D.5 Division of Cognitive and Computational Labour

A foundational principle of visual analytics is the effective division of labour between human and machine, combining "the best of both sides" [38]. Keim et al. emphasised that effectively switching between tasks for the computer and tasks for the human requires understanding what each contributes. However, this principle has remained largely at the level of general guidance. The formal workflow library provides an opportunity to examine how this division manifests concretely in published VA workflows (Table18). The pattern reported below is not prescribed by ATWL but emerges empirically from the design choices of the workflow authors.

Actor	Characteristic transform	Rationale
Machine	`define-unit`, `characterise`, `contextualise`, `build-model`	Scalable, deterministic computation
Human	`assess`, `abstract`, `generate-knowledge`	Qualitative judgment; semantic interpretation
Hybrid	`define-unit` (interactive), `visualise` (exploratory), `abstract` (partial)	Computation steered by human judgment

TABLE 18: Division of cognitive and computational labour observed in library workflows.

The pattern observed in the library can be summarised as: *machines handle scale, humans handle meaning, and hybrid transforms mediate between them*. Machine actors consistently handle data partitioning, feature extraction, dimensionality reduction, clustering, model training, and static visualisation rendering. Human actors consistently fulfil the transforms requiring semantic interpretation: evaluating whether results meet interpretive criteria (`assess`), assigning meaning to visual patterns (`abstract`), and articulating insights (`generate-knowledge`).

Hybrid actors appear in two distinct roles that correspond to the two entry modes described in Section 6.1. In data-centric workflows, hybrid transforms are predominantly `define-unit` where the method is computational but its parameterisation is interactively determined. Examples are selecting a dendrogram cut [21], choosing a similarity threshold [28], specifying query conditions [24], or labelling exemplars [32]. In model-understanding workflows, hybrid transforms are predominantly `visualise` where the analyst interactively controls what is displayed [34], [36], [37], meaning that constructing a visualisation is by itself an analytical act.

A notable difference in the treatment of visualisation is observed between the two entry modes. In data-centric

	Problem A		Problem B	
Indicator	Prose	Formal	Prose	Formal
Words in preparatory analysis	1184	794	1377	633
Words in workflow recommendation	1069	2712	834	997
Library-source references	15	62	15	12
ATWL intent-vocabulary terms	0	39	7	42
Artifact-type terms	22	91	23	35
Named iteration loops	0	5	0	2
Adaptation-table rows	0	6	0	6
Notebook cells with assess flag	—	—	1	2

TABLE 19: Surface indicators across the four sessions. Counts are of explicit occurrences (e.g., a named "Loop L1" with stated condition).

workflows, `visualise` is almost exclusively machine-executed: the system renders a predetermined encoding; the human explores and interprets the resulting display. In model-understanding workflows, it is frequently hybrid: the analyst interactively configures the view. Whether this distinction reflects a deep structural difference or a historical tendency in how tools for each paradigm have been designed is worth investigating with a broader sample.

APPENDIX E
COMPARATIVE RECOMMENDATION EXPERIMENT: DETAILS

This appendix gives the detailed evidence behind the comparative experiment summarised in Section 7. The four agent sessions, the user problems, and the agent outputs are referenced at https://geoanalytics.net/VAworkflows/experiment. The new bike-sharing agent's ATWL specification and adaptation table inform Tables 21 and 22 below. The Problem B agent's natural-language description appears in Appendix G. An earlier, more elaborately structured illustrative recommendation for Problem A, produced in a separate session with an ATWL-equipped agent that was asked to deliver its output as a LaTeX document with explicit phase-to-library cross-references, is given in Appendix F as a complementary worked example.

E.1 Experimental Design

We used four independent sessions of the same model (Claude Opus 4.6R) in a 2×2 design:

Prose / Problem A. Library = 17 source papers as PDFs; user task = bike-sharing analysis and relocation strategy (conceptual, no data).

Formal / Problem A. Library = ATWL definition + 17 ATWL workflow representations; user task = bike-sharing (same wording).

Prose / Problem B. Library = 17 PDFs; user task = topic evolution in IEEE VIS publications 1990–2024, with a request for an implementing Jupyter notebook.

Formal / Problem B. Library = ATWL definition + 17 ATWL representations; user task = same as above.

Each session began with a library-preparation step in which the agent was asked to identify reusable components without knowing the upcoming problem, followed by the problem statement and, for Problem B, a follow-up requesting the notebook. The two problems were chosen to differ along two dimensions: complexity (B simpler, A more open-ended) and groundedness (B has data, allowing an executable artifact; A is conceptual).

E.2 Quantitative Indicators

Table 19 reports surface indicators from the four outputs. The pattern is consistent across both problems: prose-library agents produce slightly longer preparatory analyses, while formal-library agents produce markedly more structured recommendations. The contrast in named loops and adaptation-table rows is the clearest quantitative signature of the difference.

E.3 Form of the Recommendations

The recommendations from the prose-library agent are organised as a methodological narrative: section headings name the analytical step ("Defining places by grouping spatially close stations", "Topic extraction"), tables enumerate concrete methods and parameters, and source papers are cited inline. The recommendations from the formal-library agent are organised as a formal specification: artifact declarations with types and embedment, transform blocks with intent and actor designations, and explicit loop blocks. On request, the formal-library agent also produced a natural-language description of the recommended workflow (Appendix F for Problem A; Appendix G for Problem B), comparable in length and clarity to the prose-library narrative.

The two formats therefore generate recommendations of *different shape* from broadly equivalent content. The choice between them is in part a choice about what the analyst will do next: read a narrative to inform manual implementation, or hand a structured specification to a downstream tool (diagram renderer, code generator, structural validator).

E.4 Reusable Components Identified by the Agents

Before being shown a problem, each agent was asked to analyse the library and identify reusable components. The prose-condition prompt was "*use this as a workflow library for recommending workflows addressing new problems; analyse the library and extract reusable patterns*"; the formal-condition prompt was equivalent ("*perform initial analysis for revealing reusable and adaptable parts*"). The two prompts were open-ended in the same way: any difference in the resulting catalogues comes from the input format, not from prompt steering. The four catalogues differ in instructive ways. We focus on three properties: the analytical content of the identified components and how it overlaps across sessions; the level of abstraction at which components are described; and the kinds of structure the agent identifies in addition to sub-workflows.

Substantive overlap. The four catalogues identify a largely overlapping set of analytical ideas. Table 20 maps the components reported in each session to a common set of analytical themes. All four sessions identify clustering

Analytical theme	Prose / Problem A	Formal / Problem A	Prose / Problem B	Formal / Problem B
Iterative clustering with visual assessment	Pattern 3 (Cluster-then-Explore)	Module A	Pattern 4 (Cluster-Label-Distribute)	Motif A
Iterative model building with residual feedback	Pattern 1	Module C	Pattern 3	Motif B & E
Multi-level model diagnostics drill-down	Pattern 6	Module H	Pattern 10	Segment 6
Feature engineering loop	Pattern 8	Module E	Pattern 7	Segment 3
Temporal pattern discovery via clustering + temporal layout	Pattern 9	Module G	Pattern 11	Segment 2
Progressive abstraction / simplification	Pattern 2	Module D	Pattern 1 & Pattern 5	Segment 7
Projection-based exploration (vectorise + DR + scatter)	—	Module B	implicit in Pattern 11	Segment 1 & Motif D
Human-in-the-loop / interactive model steering	Pattern 4	Module I	Pattern 7 & Pattern 12	Segment 5
Similarity-based recommendation / prescription	Pattern 5	Module F	Pattern 6	—
Spatio-temporal event aggregation	Pattern 7	in adaptability map	Pattern 9	Segment 4
Exploratory model analysis (problem–model exploration)	Pattern 10	in adaptability map	Pattern 8	—
Total components named	10 patterns	9 modules	12 patterns	6 motifs + 7 segments

TABLE 20: Reusable components identified by the agent in each of the four preparatory analyses, mapped to a common set of analytical themes. The four catalogues largely overlap in substantive content; differences are in naming, form, and coverage rather than in which analytical ideas are recognised as reusable.

with visual assessment, model-building with residual feedback, multi-level model diagnostics, feature engineering, temporal pattern discovery, and human-in-the-loop steering as recurring components. Differences are in coverage and naming rather than in what the agent found.

Form of description: a worked comparison. The clearest difference between the two formats is visible when matched components are placed side by side. Both the Problem A prose and the Problem A formal catalogues identify "iterative model building with residual feedback" as a reusable component. The prose catalogue presents it as Pattern 1 with the core loop stated as

> *Build model → Visualize errors/residuals → Identify weaknesses → Refine → Repeat,*

followed by six numbered "Generic Steps" that interleave structural moves with method choices — e.g., step 2 "Build initial model on top-ranked feature(s)", step 5 "Refit model; evaluate improvement (e.g., RMSE)". The formal catalogue presents the same component as Module C in ATWL intent vocabulary:

```
loop:
 generate-knowledge (specify model
config) →
 build-model (fit) →
 characterise (compute residuals) →
 visualise (residual distributions) →
 assess (random vs. systematic) →
 [refine model | exit]
```

The two representations of the same component are not interchangeable. The prose version is concrete and method-aware (RMSE, top-ranked features) and immediately actionable for an analyst who recognises the methods. The formal version is method-agnostic, makes the controlling specification artifact explicit (`generate-knowledge (specify model config)`), and states the loop exit as an alternative (`refine model | exit`). Six of the nine modules in the Problem A formal preparation follow this same loop-with-exit format. None of the ten patterns in the Problem A prose preparation does. The prose form lists steps and notes "Repeat" or "iterate until" as a final step, but does not separate the loop from its body or name the artifact that controls it.

Loop-bearing vs. linear components. The formal catalogues consistently distinguish loop-bearing components from linear ones. Seven of the nine modules in the Problem A formal preparation are explicit loops (Modules A, C, D, E, F, H, I); the remaining two (Modules B and G) are linear intent sequences without an exit condition. Prose catalogues describe iteration narratively but use the same numbered-list form for iterative patterns (e.g., Pattern 1, step 6: "Stop when gains are negligible or overfitting appears") and linear ones (e.g., Pattern 7, which is a six-step pipeline with no repetition). Whether a component is iterative is therefore immediately visible in the formal catalogues and recoverable only by reading the steps in the prose catalogues. This asymmetry is the upstream cause of the iteration-structure differences seen in the recommendations themselves (Section E.6).

Domain anchoring. The prose catalogues occasionally name components in domain-specific terms. Pattern 7 in the Problem A prose preparation is called "Spatio-temporal Event Aggregation Pipeline" and its core idea is stated as "*Extract events from trajectories, cluster them into places, aggregate by space×time, then analyse temporal profiles of places*" — a description specific to movement data. Pattern 11 in the Problem B prose preparation is called "Symbolic Encoding + Topic Mining" and is anchored to multivariate time-series data. The corresponding formal components — Module G "Temporal Partitioning → Profiling → Grouping" and Segment 2 "Calendar/context-based temporal distribution" — are framed at the level of analytical intent and apply equally to non-movement, non-temporal data wherever the same intent sequence is appropriate. This is consistent with ATWL's design choice to classify components by intent rather than by domain or method.

Phase	Formal-library agent: named loop & specification	Prose-library agent: equivalent passage
Define places	Loop with `clustering_spec`; exit when places are geographically coherent	"Validate clusters visually on a map. Adjust thresholds interactively until..."
Critical-state detection	Loop with `threshold_spec`; exit when events correspond to real disruptions	"Define events as threshold crossings" (no iteration described)
Pattern discovery	Loop with `pattern_clustering_spec`; exit when clusters are distinct	"Cluster similar daily profiles..." (no iteration described)
Predictive model	Loop with `model_spec`; exit when residuals show no systematic patterns	"Validate using residual analysis with partition-based visualisations..."
Allocation plan	Loop with `allocation_spec`; exit when plan reduces critical events at feasible cost	"Compute expected deficit/surplus... formulate a redistribution plan" (no iteration described)

TABLE 21: Loop structure in the Problem A recommendations. The formal-library agent names five loops and their controlling specifications; the prose-library agent describes iteration narratively in one of these five phases and not at all in the others.

What else is identified. Beyond named sub-workflows, the two formats surface different kinds of additional structure. Both prose catalogues end with a table of *cross-cutting design principles*, attributed to source papers: linked multi-view coordination (all papers), ranking with small multiples (Partition-Based, RfX, EMA), derived quantities as exploration targets (Partition-Based, MobilityGraphs), adjustable level of detail, colour-coded categorical/temporal membership, semantic interaction / direct manipulation (UTOPIAN, HITL, What-If), provenance / undo, and model-agnostic explanations (explainMLVis, What-If). These are UI- and interaction-design patterns; ATWL by design does not capture them. The formal catalogues instead produce two distinctively formal artefacts. One is a table of *common artifact templates* — typical structural shapes such as "Specification controlling a loop: `specification, representation form: "parameter settings"`", "Arrangement via projection: `arrangement (entities), context: projection_space, principle: "dimensionality reduction"`", and "Cluster entities: `entities, internal structure: group/cluster, embedment: set`", each with the workflows in which they recur. The other is a *cross-workflow adaptability map* that indexes goals to library templates: "*Find patterns in time series* → workflows 1.1, 1.9 with Modules G and A"; "*Build a predictive model iteratively* → workflows 1.10, 1.13 with Modules C and F"; and eight further rows of the same kind. The two kinds of supplementary structure are complementary rather than competing: the prose catalogues contribute UI-design vocabulary the formal catalogues cannot express, while the formal catalogues contribute data-flow vocabulary and a goal-to-template index that the prose catalogues do not articulate.

Summary. The four preparatory catalogues largely agree on which analytical ideas are reusable but disagree on how to name, group, and present them. The formal catalogues are more abstract, more disciplined about separating loop-bearing components from linear ones, more uniformly cross-domain in framing, and supplement the component list with structural templates and goal-to-template indexing. The prose catalogues are more concrete, more method-aware, and supplement the component list with UI-design principles that the formal vocabulary does not capture. The differences in component representation are upstream of, and consistent with, the differences observed in the recommendations themselves.

E.5 Convergence and Divergence on Workflow Content

On Problem B (research-topic evolution), the two recommendations converged on essentially the same workflow: combine title and abstract per paper, vectorise, fit a topic model (NMF in the prose case; BERTopic initially in the formal case, switched to NMF on request), assign documents to topics, aggregate counts per year, apply temporal smoothing, visualise as stacked area or streamgraph, and classify topics as rising, declining, or stable. The selected library sources also overlapped (UTOPIAN [29] for interactive topic refinement, Episodes-and-Topics [26] for the multi-attribute view, Cluster-Calendar [21] or MobilityGraphs [23] for the temporal display).

On Problem A (bike-sharing) the two recommendations were structurally similar but the formal-library version was substantially more detailed in terms of fragment-level reuse. The formal-library agent organised the workflow into five analytical phases (with phase 5 split into a predictive sub-phase 5a and a prescriptive sub-phase 5b) and produced an explicit adaptation table whose six rows each cite the library workflows the phase drew on, identify the reused element at module granularity, and itemise the adaptations (e.g. "Phase 1, source 1.6: spatial-clustering loop with visual assessment; changed from trajectory stops to station locations; added capacity aggregation"). The prose-library agent organised the workflow into six numbered sections (five analytical sections plus a summary pipeline diagram) and cited about seven library papers in the running text, but produced neither a phase-to-source mapping nor an itemised list of adaptations.

We did not find cases where one format led the agent to a substantively different workflow that the other could not have produced. The divergences are about *how* the workflow is expressed and *how its provenance is recorded*, not about *what* workflow is recommended.

E.6 Iteration Structure in Recommendations and Notebooks

The clearest behavioural difference between the two formats concerns iteration. In Problem A the formal-library agent declared five loops with named specifications and stated termination conditions (Table 21); the prose-library agent described iterative refinement narratively without enumerating loops or naming the parameters updated.

The downstream consequence is visible in the Problem B notebooks. The formal-library notebook contained two dedicated assessment-cell pairs (T6 / Loop L1 for

Ph.	Goal	Source	Adapted element
1	Group stations into places	[24]	Spatial-clustering loop with visual assessment
2	Place-level time series and occupancy	[31]	Spatio-temporal aggregation
3	Critical states (empty/full)	[32]	Feature-engineering loop with threshold tuning
4	Daily-profile clustering and flow patterns	[21], [23]	Calendar-based pattern view; flow-graph view
5a	Predictive model	[30], [31]	Residual-based model refinement
5b	Allocation plan	[28]	Prescriptive plan-tuning loop

TABLE 22: Adaptation table from the formal-library agent for Problem A, linking each phase to a library source and an adapted element.

topic quality; T12 / Loop L2 for smoothing) with markdown text stating the assessment questions, a Boolean flag (`topics_satisfactory`, `smoothing_satisfactory`) the analyst can set to `False` to trigger refinement, and prefilled suggestions for what to change (specific values of `n_topics`, `loess_frac`, etc.) and where to re-run from. The prose-library notebook contained one comparable steerable section (the `MERGE_MAP` for topic merging); the other iterations mentioned in its recommendation ("iterate until topic set is stable"; "adjust smoothing parameters") did not surface in the notebook as explicit pause points.

This pattern reflects, plausibly, a property of code generation by LLMs: when the source specification names a loop and an updateable specification artifact, the model emits the loop as a discrete object in the notebook; when the source specification describes iteration in prose, the model often realises it as a single configurable parameter at the top of a cell.

E.7 Provenance Traceability

Both formats yielded library-grounded recommendations, but the granularity of provenance differed. The formal-library agent produced explicit adaptation tables (Table 22) that linked each phase of the new workflow to one or more library workflows, identified the reused element (typically a building block named at the level of intent sequence), and itemised the adaptations made.

The prose-library agent cited the same papers in running prose but did not produce a comparable fragment-level table; the relation between any specific paragraph of the recommendation and the cited paper is left for the reader to verify. Both forms of citation are honest; the formal one is more easily auditable.

E.8 Methodological Detail: Where Prose Wins

ATWL by design classifies transforms by analytical intent, not by computational method. Method-level material — specific algorithms, parameter values, validation procedures, exceptions — is therefore absent from the formal representation, even though it is present in the source papers. The prose-library agent could draw on this material directly.

The contrast is visible in the preparatory analyses (Section E.2, Table 19): the prose-library agent's preparatory analysis was longer than the formal-library agent's in both problems, and inspection shows the additional length is concentrated in method-specific content. For example, in the Problem A preparatory analysis the prose-library agent named "density-based spatial clustering (e.g., DBSCAN or OPTICS) with geographic distance thresholds informed by walking distance (e.g., 200–400 m)" as the recommended technique for spatial grouping. The formal-library agent named the same intent (`define-unit` via clustering) but provided no comparable methodological commitment from the library — it later recovered a similar recommendation when producing the workflow, but from its general training rather than from the library.

The asymmetry is structural, not contingent on the specific library: ATWL deliberately omits method-level material to support cross-method comparison and intent-level transfer, and so a formal library cannot match a prose library for method-level detail. The trade-off is intrinsic to the design choice.

E.9 Considerations Beyond the Four Sessions

Two considerations are not directly examined by varying parameters across the four sessions but bear on the comparison in practice. The first is partially evidenced by what happened during our experiment.

Scaling and context capacity. Even at 17 workflows the prose format strained context capacity. The archive of 17 PDFs (about 94 MB) was reported by the interface as utilising 149 % of context and was delivered to the agent in truncated form, with a warning that image content might be incomplete. The session ran to completion and the recommendations are usable, but the agent's view of the library was demonstrably partial. At 50 or 100 workflows the prose library would exceed any current context window even before the user's problem and the agent's reasoning are accounted for, whereas the ATWL representations — roughly an order of magnitude smaller per workflow — would remain manageable. A scaling experiment across library sizes is beyond the scope of this paper, but the 17-workflow case already provides one data point at the boundary of feasibility.

The library as a persistent artifact. A prose library is the union of the original papers; its conceptual organisation must be re-derived by every agent in every session. The ATWL library has a different property: the abstractions used to formalise each workflow — the intent vocabulary, the typed artifacts, the loop conventions — are externalised once and shared across all 17 representations. They are also independent of the LLM session: the same abstractions are available to any agent or human consulting the library. Our seventeen-workflow library is not yet community-curated; we hope it will become so (Section 8.4). The persistence of abstractions is, however, a property of the formal representation, not of the curation process. Community curation, if it materialises, would amplify this property but is not a precondition for it.

Combined use. The two considerations above, together with the asymmetric strengths identified in Sections E.6–E.8,

suggest that the two formats are most useful in combination rather than as alternatives. In a two-stage process, the ATWL library is queried first to obtain a structured, library-grounded recommendation with a fragment-level adaptation table; in a second, separate session the small subset of papers identified by the adaptation table is consulted for method-level detail. The second stage requires only a few PDFs rather than the whole library, so it stays within feasible context budget even when the prose library as a whole would not. The adaptation table produced by ATWL thereby plays the additional role of an index, focusing the prose consultation on the papers most relevant to the recommendation rather than the entire corpus.

E.10 Threats to Validity and Caveats

Sample size. Four sessions are sufficient to refute the strong claim of an earlier draft (that prose-based retrieval cannot produce coherent workflow recommendations) and to identify consistent differences, but they do not support quantitative claims about which format produces better recommendations on average. A larger study with multiple problems, multiple runs per condition, and independent assessors would be required.

Single model. All four sessions used Claude Opus 4.6R. Smaller or earlier models would likely struggle more with the prose condition (context window, sustained reasoning over many PDFs), shifting the comparison in favour of the formal representation. Other current frontier models would need separate evaluation.

Library composition. The 17 library workflows were not selected for this experiment; they are the same library used in Section 6. About half were drawn from the present authors' prior work. This may make the library easier to recommend from than an arbitrary VA corpus, but it does so equally for both conditions.

No independent users. Recommendations were assessed by the authors. Whether the differences identified here (named loops, typed artifacts, adaptation tables) translate into measurably better outcomes for the analyst remains an open empirical question.

Single setting per problem. Problem A was conceptual without data; Problem B was grounded with data and a notebook deliverable. The contrast in iteration structure between formats was observable in both, but the difference in notebook scaffolding (Section E.6) rests on a single notebook-producing comparison per format. The pattern is consistent with how LLMs typically translate structured specifications into code, but is itself a single observation.

E.11 Summary

The comparative experiment supports four claims and refutes one. It refutes the claim that formal representation is a precondition for coherent LLM-based workflow recommendation: both formats produced usable recommendations. It supports the narrower claims that formal representation gives recommendations (i) explicit, named iteration structure that propagates into generated code, (ii) fragment-level adaptation provenance, (iii) typed data flow between fragments enabling structural composition, and (iv) compactness that supports scaling. The prose representation conversely retains method-level detail that the formal representation, by design, abstracts away. These complementary strengths argue for using both representations together rather than treating either as a replacement for the other.

Appendix F
Recommended Workflow for Analysing Bike-Sharing Spatio-Temporal Patterns

This appendix presents an illustrative natural-language workflow for the bike-sharing problem (Problem A of Section 7). It was produced in an earlier session in which an LLM agent equipped with the ATWL workflow library was asked to draft a recommended workflow as a structured LaTeX document with phase-by-phase pointers to library workflows. We include it as a worked example of how a formal-library agent can articulate its recommendation in human-readable form, with explicit cross-references from each phase to the library sources informing it. The four-session comparative experiment reported in Section 7 used separate, freshly initialised agents; the new bike-sharing agent's ATWL specification and adaptation table are the basis for Tables 21 and 22 in Appendix E.

F.1 Overview

The analysis proceeds through a place-definition step (Phase 0) followed by six analytical phases, each building on results from the previous one. The overall logic is:

0) **Define "places"** by grouping nearby docking stations.
1) **Aggregate trip data** to the place level and compute availability time series.
2) **Discover temporal demand-supply patterns** and relate them to the weekly, seasonal, and holiday calendar.
3) **Identify and analyse critical events**—shortages and overcrowding.
4) **Build predictive models** of bike demand and supply for each place.
5) **Develop and validate an allocation/relocation strategy.**
6) **Synthesise findings** into a knowledge report.

Below, each phase is described in detail, along with pointers to published workflows that illustrate similar analytical steps.

F.2 Phase 0: Defining Places from Docking Stations

F.2.1 What You Do

You start with the known locations and capacities of all docking stations. Because many popular areas have multiple stations clustered close together, you group spatially nearby stations into *places*—your primary spatial units for the entire analysis.

Use a spatial clustering algorithm (for example, density-based clustering or distance-threshold grouping) to assign each station to a place. For each place, compute the **total docking capacity** (sum of member station capacities) and a **centroid location**.

Display the result on a **geographic map**: stations coloured by place membership, with place boundaries drawn around each group. Visually assess whether the grouping makes sense—are popular areas with multiple

stations properly unified? Is the granularity appropriate (not too coarse, not too fine)? If not, adjust the distance threshold or minimum group size and repeat.

This is an **iterative loop**: cluster → visualise on map → assess → adjust parameters → re-cluster, until you are satisfied with the places.

F.2.2 Where to Find Relevant Examples

- **Workflow 1.6** [24] demonstrates how to delineate meaningful places from data through iterative spatial clustering with visual assessment. In that paper, places are derived from movement events using density-based clustering with a custom distance function, and the analyst iteratively adjusts clustering parameters while inspecting results on a map and in a space-time cube. *Relevant aspects*: the iterative loop of clustering → map visualisation → quality assessment → parameter adjustment. Your case is simpler because station locations are already given (you do not need to extract events first), but the iterative refinement logic is the same.
- **Workflow 1.3** [23] uses graph-based spatial clustering to aggregate nearby places with strong flows into regions, also with iterative parameter adjustment guided by a quality heatmap. *Relevant aspect*: the idea of using a quality metric display (heatmap over parameter combinations) to guide the choice of spatial aggregation parameters.

F.3 Phase 1: Aggregating Trip Data and Computing Availability

F.3.1 What You Do

With places defined, you now transform the raw trip records into place-level time series. For each place and each time interval (e.g., one hour):

- Count the number of **bikes taken** (trip origins at member stations).
- Count the number of **bikes returned** (trip destinations at member stations).
- Compute the **net flow** (returns minus takes).
- Compute **directed flows between place pairs** (how many bikes moved from place A to place B).

From the net flow, compute a **running estimate of bike availability** at each place over time and the **occupancy rate** (fraction of capacity occupied). This requires either knowing the initial bike distribution or estimating it from the data and domain knowledge.

The result is a set of time series per place: takes, returns, net flow, availability, and occupancy rate.

F.3.2 Where to Find Relevant Examples

- **Workflow 1.11** [31] begins with precisely this step: transforming raw spatio-temporal records into spatial time series by dividing territory into spatial compartments and aggregating attribute values by location and time interval. *Relevant aspect*: the spatio-temporal aggregation step producing one time series per spatial unit.
- **Workflow 1.6** [24] includes a spatio-temporal aggregation step after place delineation, where events and trajectories are aggregated by places and time intervals, producing time series of counts and statistics per place, as well as directed flows between place pairs. *Relevant aspects*: aggregating both local statistics (counts per place) and relational statistics (flows between places)—exactly what you need for understanding bike redistribution patterns.

F.4 Phase 2: Discovering Temporal Patterns through Day Clustering

F.4.1 What You Do

This is the core pattern-discovery phase. The idea is to **treat each day as an analytical unit**, characterise it by its hourly demand-supply profile across all places, then **cluster days with similar profiles** and visualise the clusters on a **calendar** to reveal weekly, seasonal, and holiday patterns.

Step 2a—Partition into daily episodes. Cut all place-level time series into daily segments. Each day is now represented by a matrix of hourly values (places × hours) for takes, returns, net flow, and availability.

Step 2b—Compute daily profiles. Summarise each day by a feature vector that captures the system-wide hourly shape of demand and supply—for instance, the total takes and returns per hour across all places, plus indicators like total volume and peak-hour timing.

Step 2c—Cluster days iteratively. Apply hierarchical clustering (or another method) to group days with similar profiles. Adjust the number of clusters and distance measure interactively until you get a clear, interpretable decomposition.

Step 2d—Visualise and interpret. Use three coordinated views:

1) **Calendar view**: A grid where each cell is one day, coloured by cluster membership. Months run along one axis, days of the week along the other. This immediately reveals whether clusters correspond to weekdays vs. weekends, holidays, seasons, etc.
2) **Profile line graphs**: For each cluster, show the average hourly take and return curves (with variability bands). These show the characteristic diurnal shape for each day type—e.g., "regular weekday" might show morning peaks at residential places and evening peaks at business areas.
3) **Place-level heatmap**: A matrix (places as rows, hours as columns, one panel per cluster) where colour intensity shows the net flow at each place and hour. Blue might indicate places gaining bikes, red places losing bikes. This reveals *where* demand imbalances occur for each day type.

Optionally, also show **flow map thumbnails** for selected hours (e.g., morning peak, evening peak) with arrows between places sized by flow volume, to visualise the dominant movement patterns.

Examine these views together. Assign interpretive labels to clusters (e.g., "regular weekday," "summer weekend," "public holiday"). If clusters are not yet clear or too many/-too few, adjust parameters and re-cluster.

F.4.2 Where to Find Relevant Examples

- **Workflow 1.1** [21] is the **primary reference** for this phase. It defines exactly this approach: partition time series into daily episodes, characterise each day by its temporal profile, hierarchically cluster days by profile similarity, and display results through a calendar view (colour-coded by cluster) coordinated with line graphs of cluster-average profiles. The analyst iteratively adjusts the number of clusters, distance measure, and time interval focus until meaningful patterns emerge. *Relevant aspects*: the entire iterative loop of clustering $\rightarrow$ calendar + profile visualisation $\rightarrow$ interpretation $\rightarrow$ assessment $\rightarrow$ parameter adjustment. Your workflow extends this by making the daily profiles multi-place (matrices instead of single vectors) and adding the place-level heatmap, but the core logic is directly from this paper.
- **Workflow 1.3** [23] applies temporal clustering to flow data (grouping time steps with similar spatial flow patterns) and displays results on a calendar with flow graph thumbnails per cluster. *Relevant aspects*: the calendar view for temporal cluster distribution combined with small-multiple flow graph thumbnails showing representative spatial patterns per cluster—a visualisation design directly applicable to your flow map thumbnails.
- **Workflow 1.9** [26] provides a model for progressive abstraction of multivariate temporal data. It shows how symbolic encoding of temporal patterns within episodes, followed by topic modelling to discover co-occurring patterns, can reveal multi-attribute behaviours. *Relevant aspects*: the general strategy of encoding temporal variation within episodes and then discovering higher-level patterns from the encoded representations, especially useful if you want to go beyond simple day clustering and discover more nuanced combinations of place-level patterns.

F.5 Phase 3: Identifying and Analysing Critical Events

F.5.1 What You Do

Define **critical events** as time intervals when a place's occupancy rate crosses a threshold—either too low (shortage: few bikes available, users cannot take a bike) or too high (overcrowding: few free docks, users cannot return a bike). Set thresholds based on domain knowledge (e.g., below 10% of capacity = shortage, above 90% = overcrowding).

Scan the availability time series to **extract all critical episodes**: for each, record the place, type (shortage or overcrowding), start and end time, duration, and severity (e.g., estimated number of unserved users).

Then **aggregate** critical events by place, day type (from Phase 2), and hour of day to build a profile of *where* and *when* shortages and overcrowding concentrate.

Visualise the results with:

- A **map** where each place has a glyph sized by total critical event frequency, coloured by type (red for shortage, blue for overcrowding), with an embedded hourly bar diagram showing at which hours critical events occur.
- A **calendar heatmap** where cell colour intensity reflects the number of critical events per day.

Interpret these views together with the Phase 2 results. Look for **spatial complementarity**—e.g., morning shortages at residential places co-occurring with overcrowding at business-area places. These complementary patterns are the key to designing effective relocation strategies.

F.5.2 Where to Find Relevant Examples

- **Workflow 1.6** [24] provides the model for **event extraction and characterisation**: identifying relevant events from data using attribute-based criteria, then aggregating and visualising events by places and time intervals to discover spatio-temporal patterns. *Relevant aspects*: the event extraction step (applying threshold conditions to identify events), spatio-temporal aggregation of events per place and time interval, and exploration through temporal diagrams positioned on a map.
- **Workflow 1.1** [21] and **Workflow 1.3** [23]—the calendar-based visualisation approach from Phase 2 can be reused here to show the temporal distribution of critical events across the calendar. *Relevant aspect*: the calendar grid as a tool for revealing weekly and seasonal concentration patterns.

F.6 Phase 4: Building Predictive Demand-Supply Models

F.6.1 What You Do

The goal is to build a model that, given any date and time of day, predicts the expected takes, returns, and net flow at each place. This model is the foundation for the allocation strategy in Phase 5.

Step 4a—Group places by demand similarity. Cluster places whose demand-supply time series have similar temporal shapes (even if different magnitudes). This allows you to fit a shared model structure per group, with per-place scaling parameters, rather than building a separate model for each place.

Step 4b—Identify temporal components. For each place group, use the cluster profiles from Phase 2 to identify the temporal variation components: the diurnal cycle shape, weekly modulation (weekday vs. weekend), seasonal variation, and holiday effects.

Step 4c—Fit a time series model. Derive a representative time series for each place group. Select and configure a modelling method that can handle multiple seasonal components (e.g., a multiplicative seasonal model with 24-hour and 168-hour cycles, plus seasonal and holiday adjustment factors). Fit the model and overlay the model curve on the actual data.

Step 4d—Iteratively refine. Assess the model visually: does the model curve capture the characteristic diurnal shape? The weekday/weekend distinction? The seasonal variation? If not, adjust parameters and refit.

Step 4e—Evaluate through residual analysis. Compute residuals (actual minus predicted) for all places and examine their distribution over time and space. Residuals should look random; if they show systematic patterns (e.g., consistently underestimating demand at certain hours or places), this indicates the model needs refinement—perhaps additional components, different grouping, or special handling of holidays.

This is a **nested loop**: an outer loop over the grouping–modelling–residual cycle, and inner loops for cluster refinement and model parameter tuning.

F.6.2 Where to Find Relevant Examples

- **Workflow 1.11** [31] is the **primary reference** for the entire modelling phase. It describes precisely this approach: cluster spatial time series by temporal similarity, visually identify temporal variation characteristics, derive representative series, configure and fit statistical time series models, iteratively refine model parameters while comparing model curves to data, and evaluate model quality by examining residual distributions over time and space. If residuals show systematic patterns, the analyst decides whether to subdivide clusters, adjust the modelling approach, or both. *Relevant aspects*: essentially the entire modelling workflow—grouping → representative derivation → model configuration → fitting → visual comparison → residual-based evaluation → refinement decision. Your workflow adds holiday/seasonal exogenous variables and multi-scale seasonality, but the structural logic comes directly from this paper.
- **Workflow 1.10** [30] provides a complementary approach to model refinement through residual analysis. After building an initial model, residuals become the analytical target: features are re-ranked by their relevance to the residuals, revealing effects not yet captured by the model, and the analyst decides which features or interactions to add. *Relevant aspect*: the residual-based discovery loop—compute residuals → re-rank features by residual relevance → visualise conditional residual distributions → discover unexplained effects → refine model. This approach is useful if you want a more structured way to decide what to add to your model when residuals are non-random.

F.7 Phase 5: Developing and Validating the Allocation Strategy

F.7.1 What You Do

With a predictive model in hand, you now build an **allocation/relocation model**: given a date and time, use the demand-supply predictions to compute a recommended distribution of bikes across places that minimises expected shortages and overcrowding.

Step 5a—Define objectives and constraints. Specify what you are optimising: minimise total expected shortage and overcrowding events, subject to constraints—total fleet size, place capacities, and logistical limits (e.g., maximum bikes a redistribution vehicle can move per trip, operating hours of redistribution crews).

Step 5b—Formulate the optimisation model. Build a model that takes the predicted demand curves for a given date and time, and computes the optimal initial bike allocation and (optionally) a schedule of relocations during the day.

Step 5c—Validate by simulation. Test the model on historical dates: for a representative sample of dates (sampling from each day-type cluster identified in Phase 2), compute the recommended allocation, simulate the resulting availability through the day using actual demand data, and count how many critical events (shortages and overcrowding) would have been avoided compared to the actual historical situation.

Step 5d—Visualise and assess. Display a side-by-side comparison: for each place, show the critical event frequency under the actual historical allocation versus the recommended allocation. Use a map with paired bars (actual vs. recommended) and a summary table by day type showing the total reduction in critical events. Assess whether the improvement is sufficient. If certain places or day types still show unacceptable critical event rates, refine the allocation criteria—for example, adjust the relative weighting of shortage vs. overcrowding, introduce priority places, or add time-of-day relocation windows.

This is again an **iterative loop**: specify criteria → build allocation model → simulate on historical data → visualise comparison → assess → refine criteria → rebuild.

F.7.2 Where to Find Relevant Examples

- **Workflow 1.5** [28] provides the model for the **prescriptive recommendation loop**. In that paper, the analyst specifies an action plan, the system estimates the plan's impact by recomputing outcome probabilities, the analyst assesses the result, and if unsatisfied, refines the plan based on outcome feedback. This cycle continues until the analyst is satisfied. *Relevant aspects*: the iterative plan-specification → impact-estimation → visualisation → assessment → refinement cycle. In your case, the "action plan" is the bike allocation, the "outcome estimation" is the simulated critical event count, and the "refinement" is adjusting allocation criteria. The structural logic is the same.

F.8 Phase 6: Synthesising Knowledge

F.8.1 What You Do

At the end, bring together all findings into a comprehensive understanding:

1) **Demand-supply pattern types**: The named day types (from Phase 2) with their characteristic diurnal shapes, place-level variations, and calendar distribution.
2) **Critical event patterns**: Which places are most affected, when, and why—including spatial complementarity patterns (paired shortages and overcrowding) explained by directional flows.
3) **Predictive models**: For any given date and time, the expected demand and supply at each place, with the model's accuracy and limitations documented.
4) **Allocation recommendations**: The validated relocation strategy with its expected reduction in critical events, along with operational guidance for redistribution logistics.

All 17 workflows in the library end with a knowledge synthesis step. This case combines pattern-based understanding (as in Workflows 1.1, 1.3, 1.6) with model-based prediction (as in Workflow 1.11) and prescriptive recommendation (as in Workflow 1.5).

Ph.	Activity	References & Key Focal Points
0	Grouping	**1.6**: Iterative spatial clustering with visual map assessment.
1	Aggregation	**1.11, 1.6**: Spatio-temporal aggregation of local counts and inter-place flows.
2	Clustering	**1.1, 1.3**: Day clustering with calendar views, profile line graphs (1.1), and flow thumbnails (1.3).
3	Events	**1.6**: Event extraction via thresholds; spatio-temporal aggregation of events on maps.
4	Modeling	**1.11, 1.10**: Grouping → model fitting → residual evaluation (1.11); residual-based feature discovery (1.10).
5	Strategy	**1.5**: Iterative loop for plan specification, impact estimation, and refinement.

TABLE 23: Primary ATWL references for each workflow phase.

F.9 Summary of Key References

Table 23 summarises, for each phase, the primary reference workflows and what to look for in the corresponding papers.

Figure 5 contains a schematic graphical representation of the workflow. The ATWL representation of the workflow is provided in online supplementary materials.

APPENDIX G
RECOMMENDED WORKFLOW FOR ANALYSING RESEARCH-TOPIC EVOLUTION IN IEEE VIS

This appendix reproduces the natural-language version of the workflow recommended by the formal-library agent for Problem B of the comparative experiment (Section 7 and Appendix E). The agent first produced an ATWL specification of the workflow, then, on request, the natural-language description below.

G.1 Goal

The workflow takes a dataset of IEEE VIS publications (each described by year, title, and abstract) and reveals the major research topics within the community and how their prominence has changed over 35 years, emphasising long-term trends rather than year-to-year noise.

G.2 Phase 1: Topic Discovery

The workflow begins by preparing the text data. Each paper's title and abstract are concatenated into a single document and transformed into a numerical vector using a pre-trained sentence embedding model. These vectors capture semantic similarity — papers about related subjects end up with similar vectors.

Next, a topic model (BERTopic) is fitted to the document vectors. The model clusters semantically similar papers together and identifies representative keywords for each cluster. Each cluster corresponds to a research topic. Every paper is then assigned to its most probable topic, and the top terms characterising each topic are extracted.

To allow the analyst to visually verify topic quality, the high-dimensional vectors are projected into two dimensions using UMAP. The resulting document map is displayed as a scatterplot where each dot is a paper, coloured by its assigned topic. A companion bar chart shows the size of each topic. Together, these views let the analyst judge whether the topics are coherent, well-separated, and at an appropriate level of granularity.

The analyst then assesses the result. If topics are too fine-grained, too coarse, or contain incoherent mixtures, the analyst adjusts the model parameters — for example, changing the minimum topic size, merging similar topics, or constraining the total number of topics — and the model is rebuilt. This refinement loop repeats until the analyst is satisfied with topic quality.

G.3 Phase 2: Temporal Profiling

Once stable topics are established, the workflow shifts to temporal analysis. For each year in the dataset, the number (or proportion) of papers belonging to each topic is counted, producing a topic-by-year matrix.

Because raw yearly counts are noisy (especially in earlier years with fewer publications), a smoothing function is applied. By default, LOESS regression gently smooths each topic's time series so that multi-year trends become visible while short-lived fluctuations are suppressed. The smoothing window is chosen to reveal patterns spanning roughly five or more years.

The smoothed trends are then visualised in multiple coordinated views: a stacked area chart (streamgraph) showing how the overall topic composition evolves, a set of individual line charts (small multiples) showing each topic's trajectory in isolation, and a heatmap showing topic intensity across years.

The analyst assesses whether the smoothing level is appropriate. If the curves are still too jagged, the smoothing fraction is increased; if important transitions are being blurred out, it is decreased. This adjustment loop continues until the temporal visualisation clearly communicates long-term trends.

G.4 Phase 3: Interpretation and Knowledge Generation

With validated topic trends in hand, the analyst interprets the patterns. Each topic is automatically classified as *rising* (gaining prominence over time), *declining* (fading), *stable* (consistently present), or *peaked* (rose and then fell). This gives a quick structural summary of the field's evolution.

The workflow then segments the 35-year timeline into distinct research eras — contiguous periods during which the topic composition remains relatively stable, separated by years of notable compositional shift. For each era, the dominant topics are identified.

An annotated visualisation overlays era boundaries on the streamgraph, making it easy to see when the community transitioned from one set of dominant concerns to another.

Finally, the analyst synthesises all observations into a structured knowledge report: which topics have emerged, which have faded, when major shifts occurred, and what characterises each era of IEEE VIS research.

G.5 Iterative Structure

The workflow contains two human-in-the-loop feedback cycles:

- **Topic refinement loop** — ensures the discovered topics are meaningful before any temporal analysis begins.
- **Smoothing adjustment loop** — ensures the temporal visualisation reveals genuine trends at the right level of abstraction.

Both loops follow the same principle: the machine computes, the human assesses, and if the result is unsatisfactory, the human updates a specification that controls the machine's next iteration.

G.6 Roles

Actor	Responsibilities
Machine	Text vectorisation, topic modelling, dimensionality reduction, smoothing, aggregation, change-point detection, rendering visualisations
Human	Assessing topic coherence, judging smoothing adequacy, interpreting trends, identifying eras, synthesising narrative findings

G.7 Output

The workflow produces three main outputs:

1) A validated set of research topics with representative keywords and a document map.
2) Smoothed temporal trend charts showing each topic's trajectory from 1990 to 2024.
3) A structured narrative summarising rising, declining, and stable themes, major transition points, and distinct research eras in the IEEE VIS community.

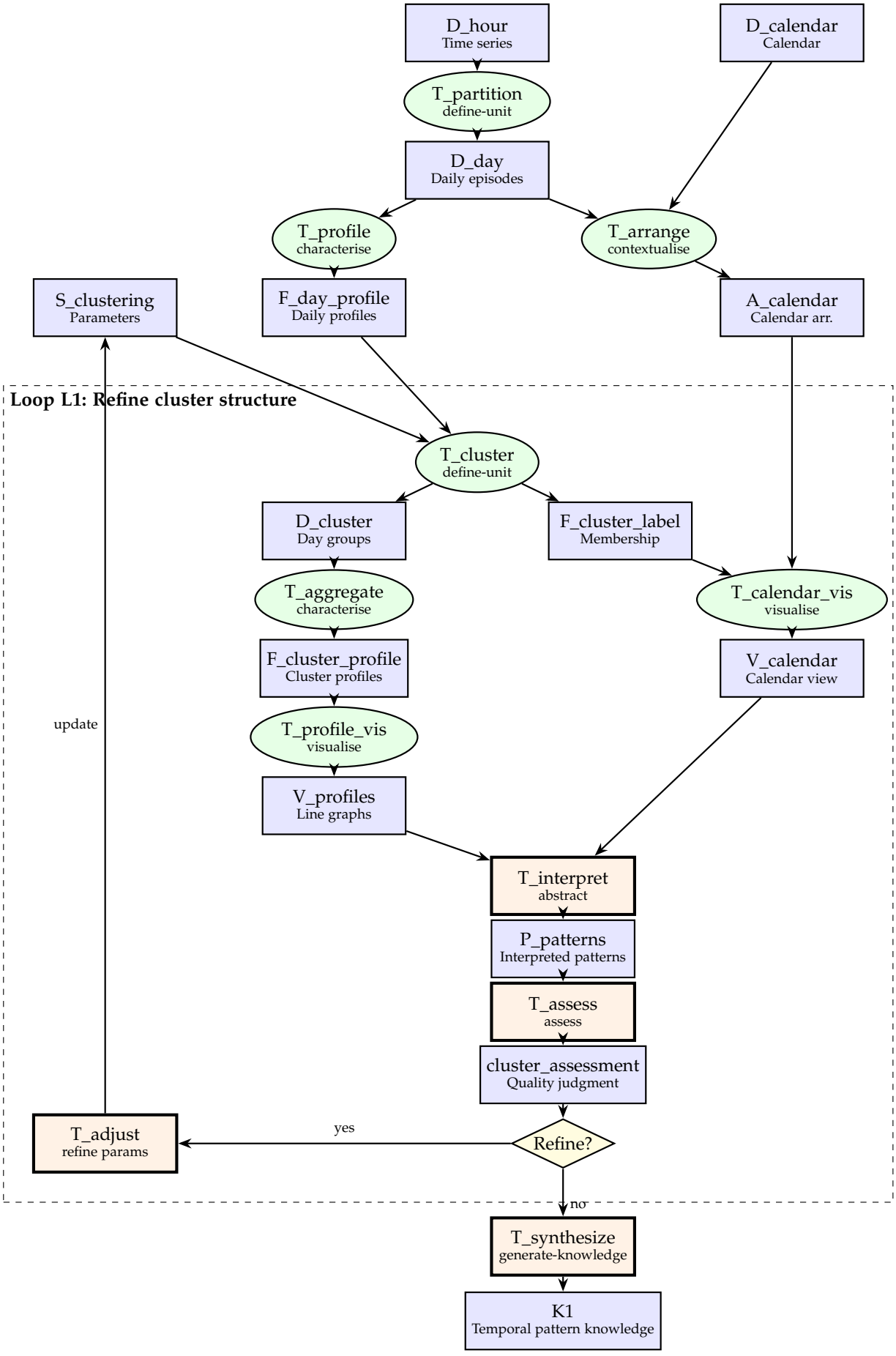


Fig. 4: Diagrammatic representation of the Cluster-Calendar workflow

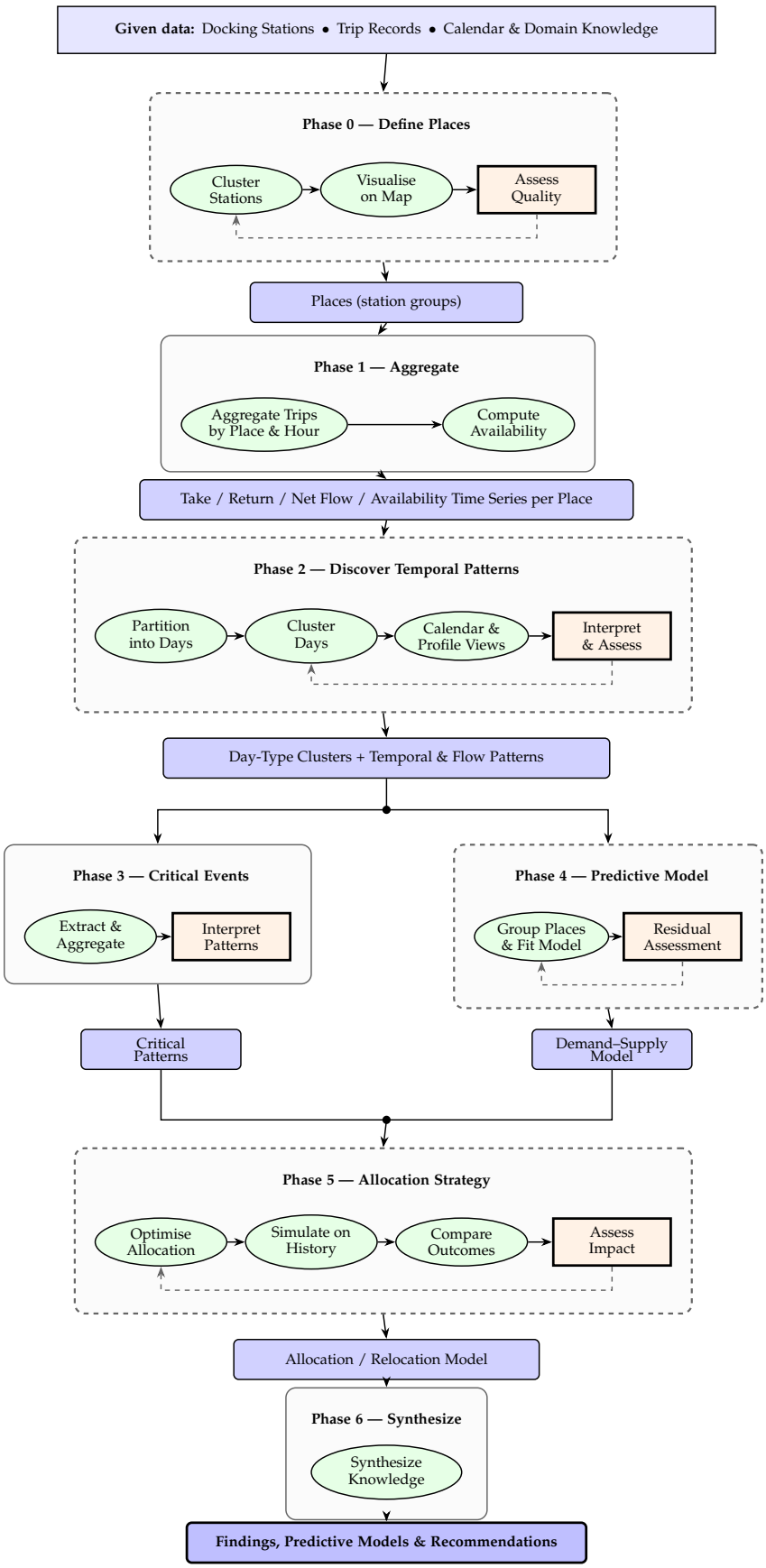


Fig. 5: High-level workflow for analysing bike-sharing spatio-temporal patterns. Green ellipses represent computational transforms; bold-bordered orange rectangles represent human assessment and decision steps; blue rounded rectangles represent data artifacts passed between phases. Dashed phase borders and dashed feedback arrows indicate iterative loops with human-in-the-loop refinement. Phases 3 and 4 are independent and may proceed in parallel; their outputs converge into Phase 5. Black dots mark branch and merge junctions.