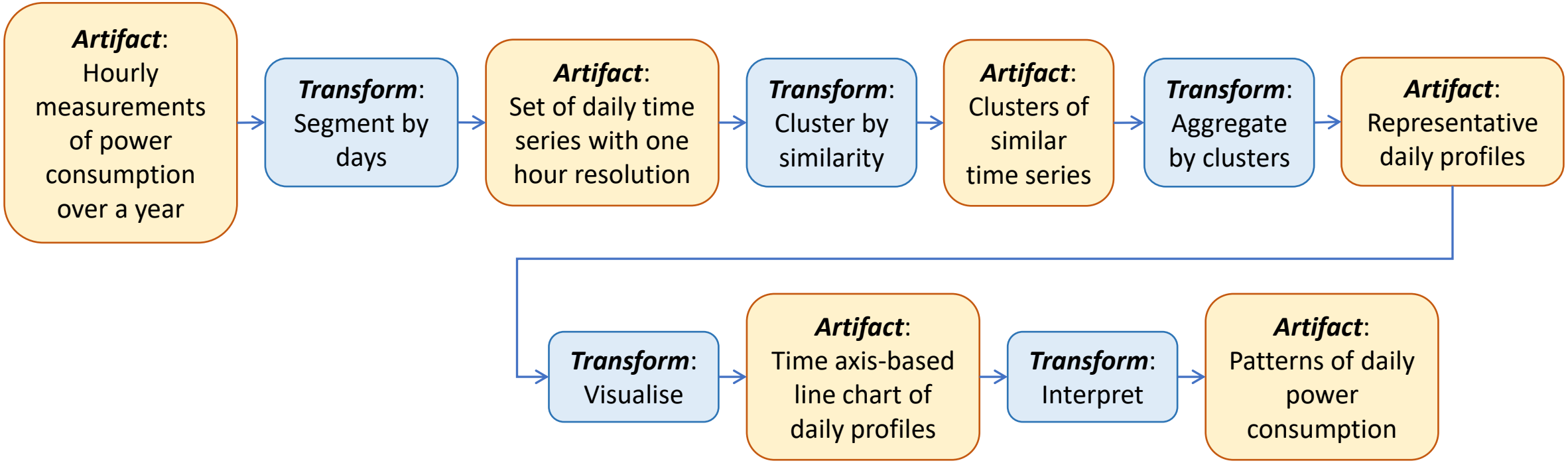
ATWL in a Nutshell

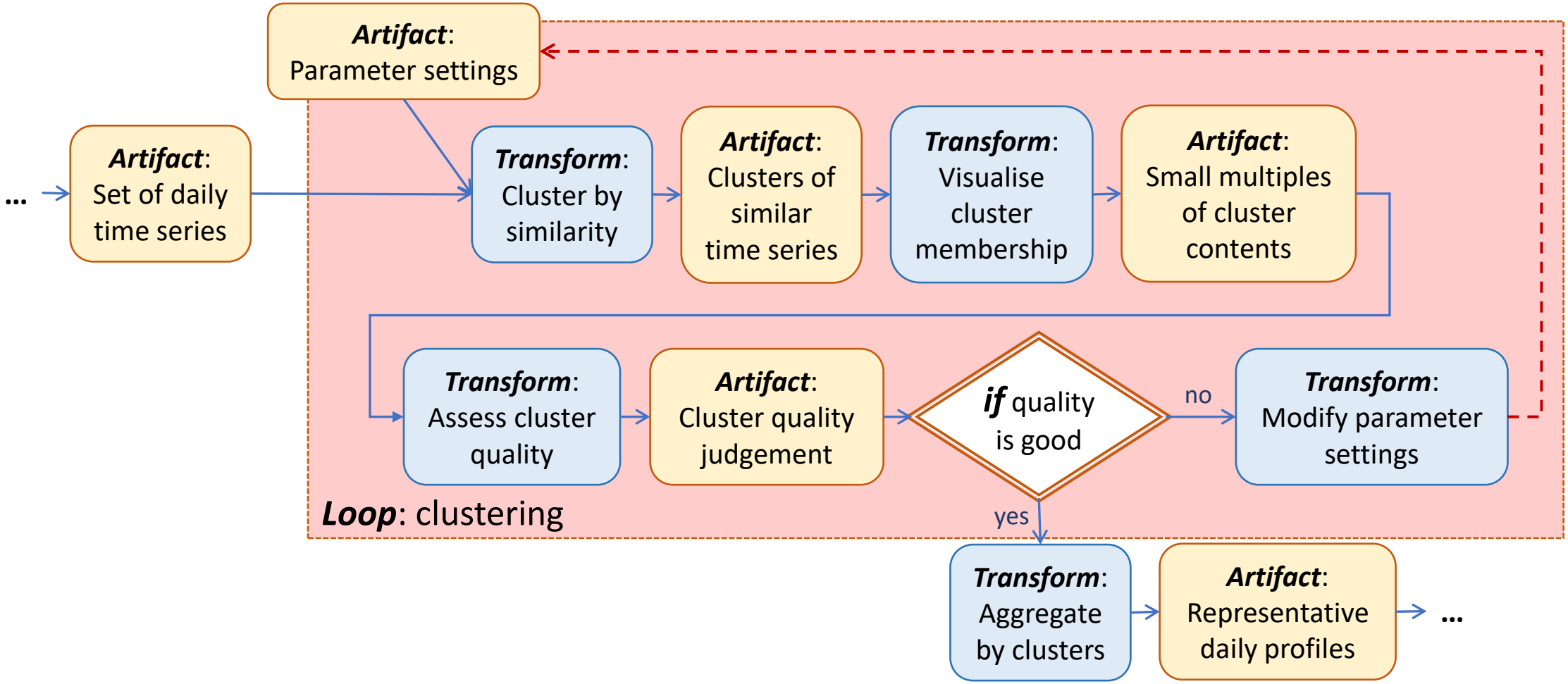
Artifact-Transform Workflow Language

Purpose: standardised representation of analytical workflows

Main idea 1: a workflow is a sequence of ***transforms*** applied to ***artifacts*** and producing new artifacts



Workflows typically contain iteration.
ATWL represents it through "*if*" and "*loop*" constructs.



Main idea 2: each transform is done by an actor: *human*, *machine*, or *hybrid*

Transform:
Segment by days
Actor: **machine**

Transform:
Aggregate by clusters
Actor: **machine**

Transform:
Visualise
Actor: **machine**

Transform:
Interpret
Actor: **human**

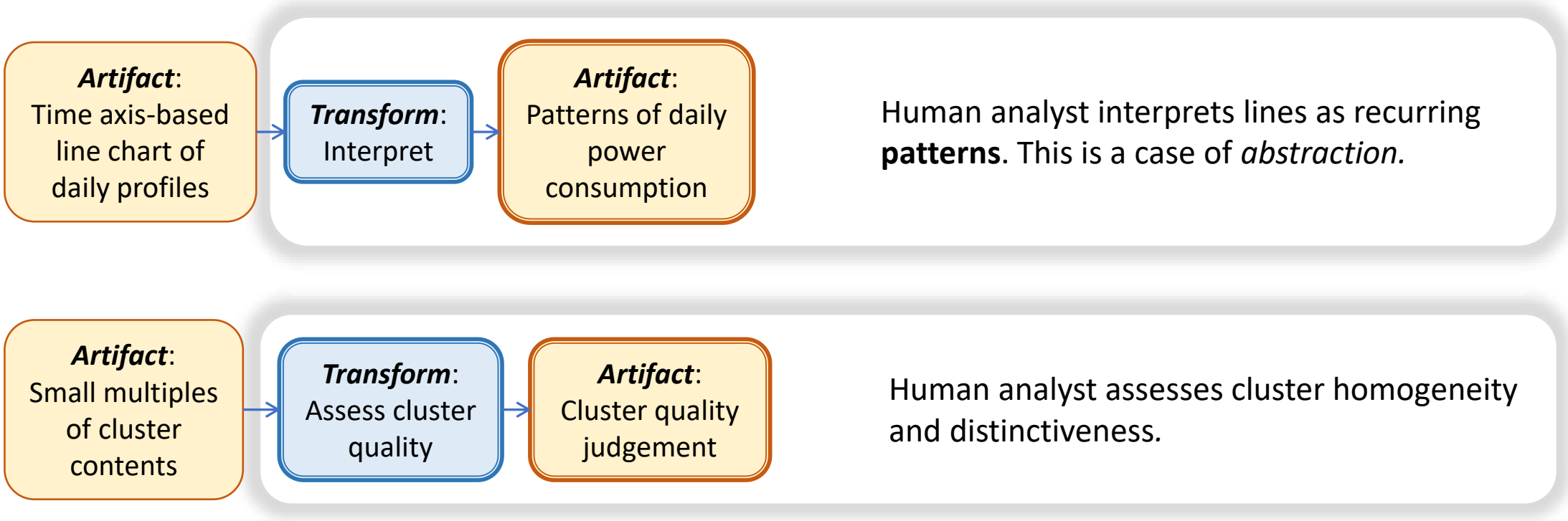
Transform:
Assess cluster quality
Actor: **human**

Transform:
Modify parameter settings
Actor: **human**

Transform:
Cluster by similarity
Actor: **hybrid**

For example, the human selects dendrogram cut in hierarchical clustering.

Main idea 3: human cognitive activities are represented by *explicit* transforms and resulting *explicit* artifacts



Important: ATWL isn't meant to represent what happens inside human's brain. It represents intermediate or final analysis results that are generated by a human and have *or could have* an explicit form: statement, judgement, method specification, label, rule, …

Main idea 4: ***abstraction*** from concrete methods, data types, and operational details

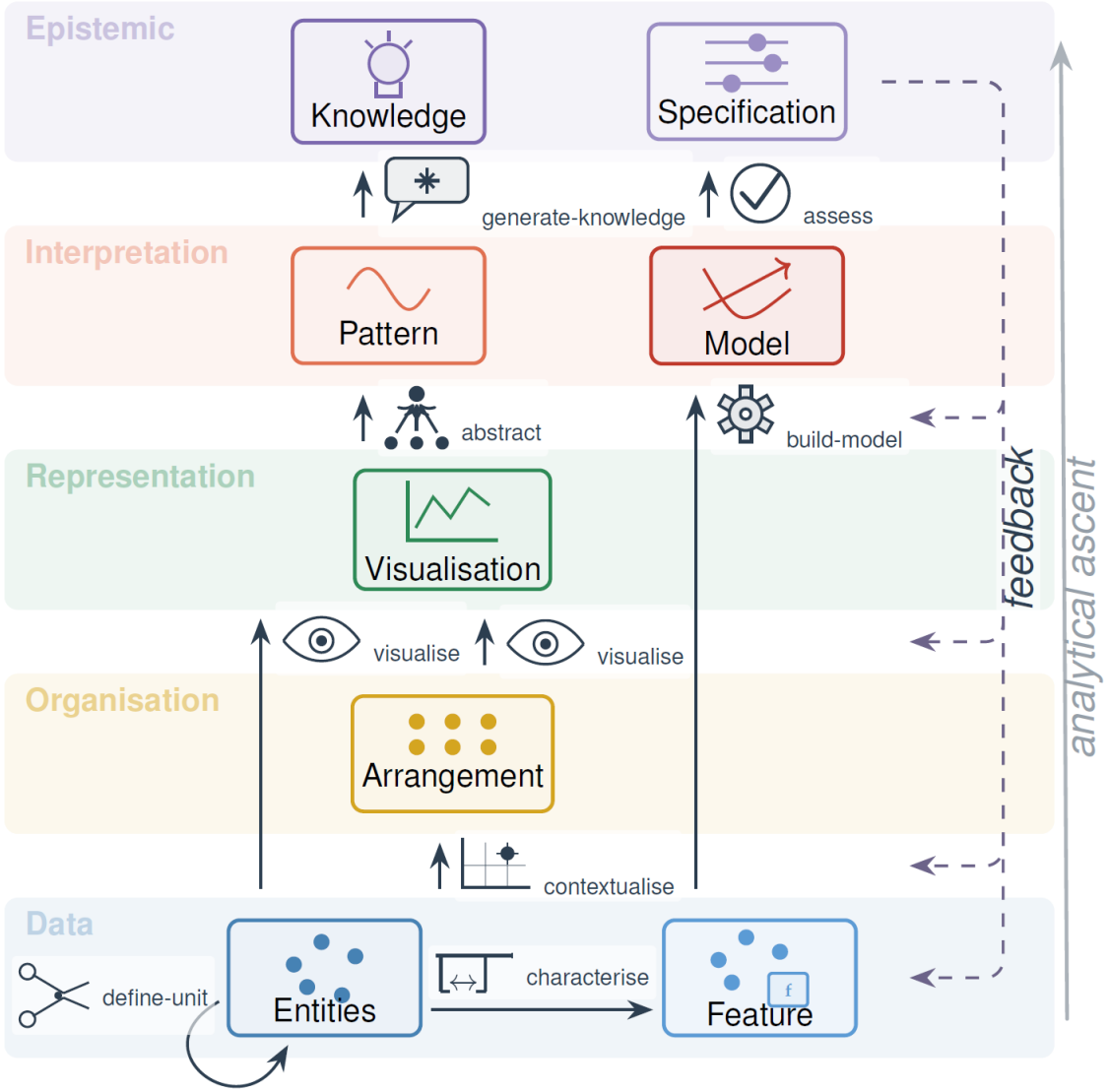
Artifact **types**:

1. ***Entities*** (what is analysed)
2. ***Feature*** (property or relationship)
3. ***Arrangement*** (positioning in a reference context)
4. ***Visualisation***
5. ***Pattern*** (observed regularity, trend, or structure)
6. ***Model*** (formal or computational representation for prediction or explanation)
7. ***Knowledge*** (explicitly formulated insights, judgements, assertions, rules)
8. ***Specification*** (control knowledge: parameters, constraints, method choices, …)

Transform **intents**:

1. ***Define-unit*** (create entities as units of analysis)
2. ***Characterise*** (compute features of entities)
3. ***Contextualise*** (position entities in a reference context)
4. ***Visualise***
5. ***Abstract*** (derive patterns or concepts)
6. ***Build-model***
7. ***Generate-knowledge*** (formulate explicit knowledge or specification)
8. ***Assess*** (evaluate quality or appropriateness of artifacts)

Conceptual architecture of ATWL



The eight artifact types are grouped into five layers (from bottom to top) according to distinct analytical roles:

- **Data** (entities, features),
- **Organisation** (arrangements),
- **Representation** (visualisations),
- **Interpretation** (patterns, models), and
- **Epistemic** (knowledge, specifications).

Transforms move artifacts upward through these layers, from raw data toward understanding.

Human-formulated knowledge and specifications feed back to earlier layers, guiding subsequent iterations.

*The layers and intents together give a shared **vocabulary** for describing what any analytical workflow does, independently of domain or method.*

Artifact types and **transform intents** are fixed and mandatory.

More specific information can be included as free texts.

```
artifact D_hour : entities
  origin: given
  ...
  description: "Time series
measurements at regular intervals
over extended period"
```

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

```
artifact D_day : entities
  internal structure: episode
  ...
  description: "Daily episodes
consisting of all measurements
within each 24-hour period"
```

Mandatory for transforms is also specification of input and output artifacts.

Intent versus *method*

```
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"
```

What does this step achieve for the analysis?

How is it done? *(free text)*

creates groups to serve as units for further analysis

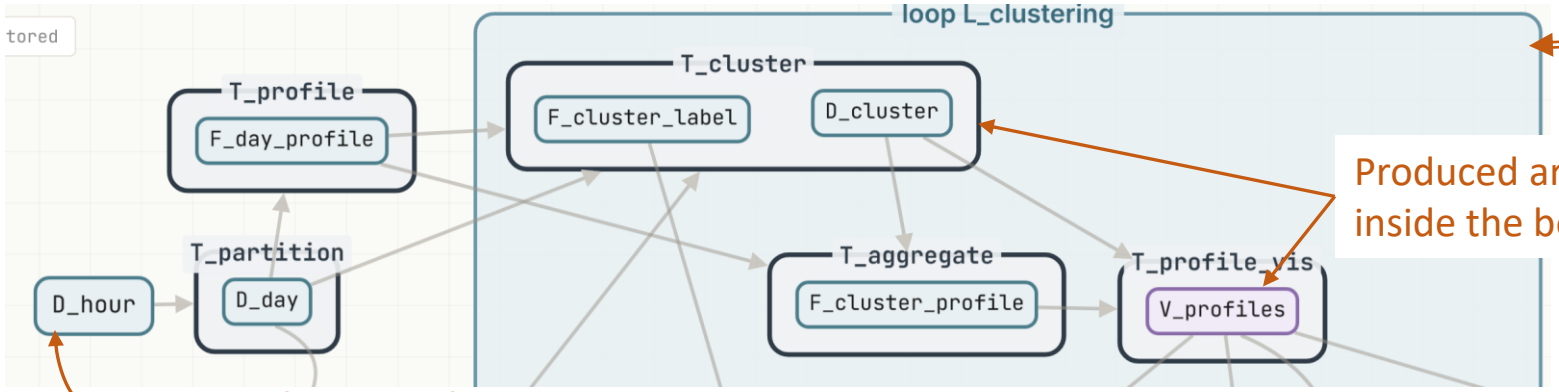
Same intent, different method:
k-means, DBSCAN, interactive grouping by hand …

Same method, different intent

```
transform T_structure :
      intent: contextualise
      manner: "hierarchical clustering by similarity"
      input: D_day, F_day_profile, S_clustering
      output: A_dendrogram, F_day_position
      actor: machine
      description: "Apply hierarchical clustering to
arrange days in a similarity hierarchy"
```

arranges days along a similarity hierarchy that serves as a reference structure for exploration

ATWL codes can be directly translated to graphical representation for more convenient reading

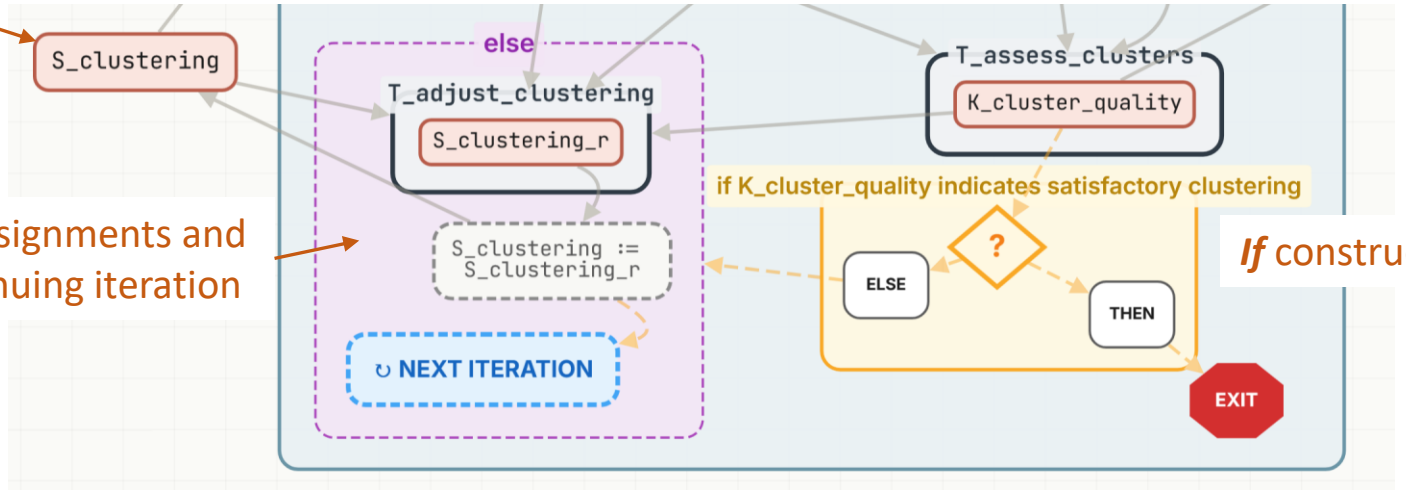


Loop

Produced artifacts are shown as internal blocks inside the boxes of the transforms generating them.

Given artifacts – defined externally

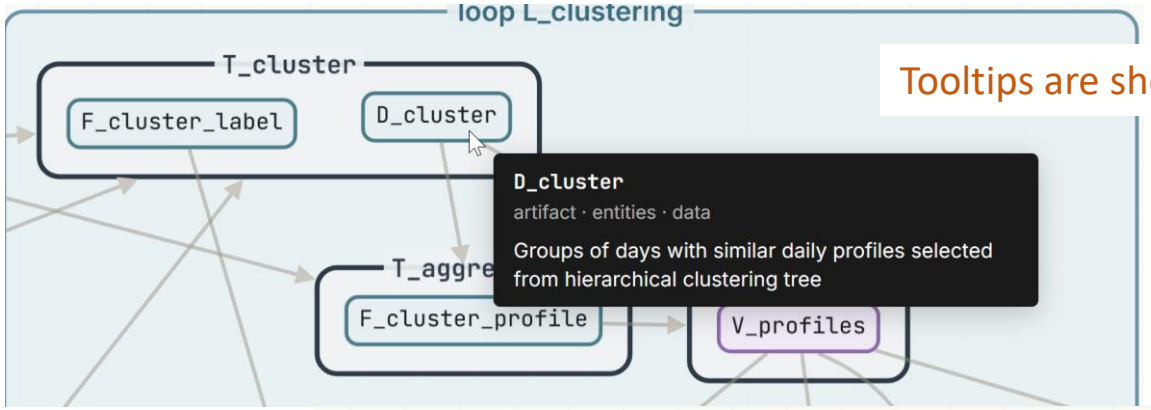
Nodes can be moved by dragging



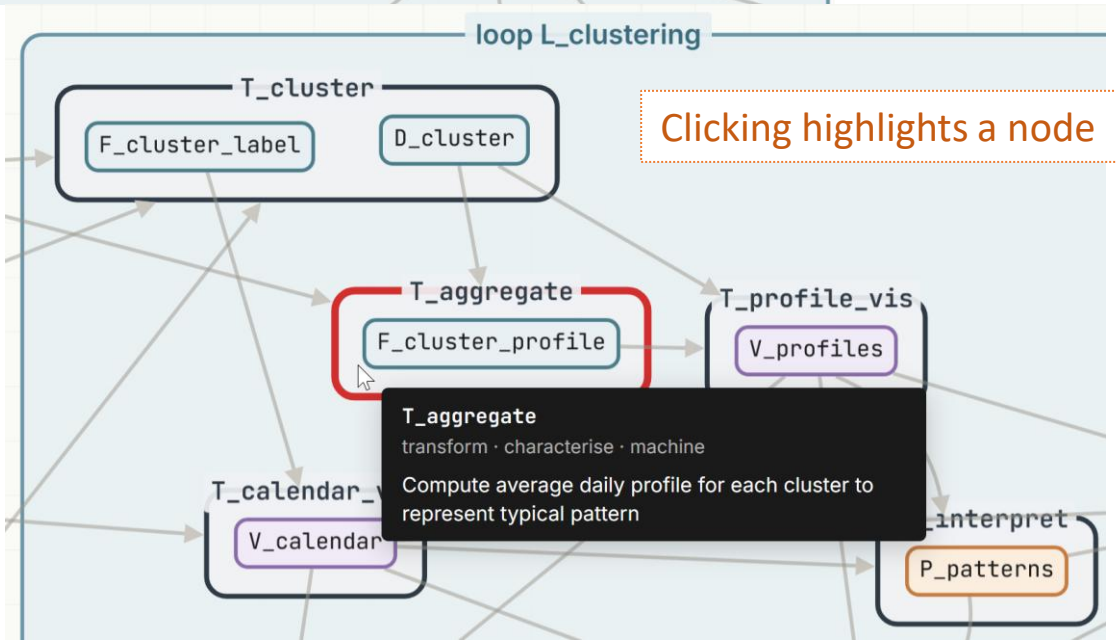
Re-assignments and continuing iteration

If construct: condition checking

Graphical viewer: details through interaction



Tooltips are shown on mouse hovering



Clicking highlights a node

TRANSFORM · CHARACTERISE

T_aggregate

characterise

ACTOR

machine

MANNER

aggregate profiles per cluster

INPUTS

D_cluster

F_day_profile

OUTPUTS (NESTED)

F_cluster_profile

DESCRIPTION

Compute average daily profile for each cluster to represent typical pattern

Panel on the right shows information about the highlighted node, which persists regardless of mouse movements.

Reference: Artifact types

Type	What it is	Examples	Produced by
entities	units of analysis	things, individuals, time moments/intervals, places, events, documents, trajectories, network snapshots	**define-unit** (or given)
feature	value or descriptor of a property or relationship	amount, quality, position, temporal profile, topic weights, distance matrix, residual	**characterise**
arrangement	positioning of entities in a reference structure	time cycles (clock, calendar), 2D projection, geographic space, hierarchy, ordering	**contextualise**
visualisation	visual representation for human perception	scatterplot, line chart, heatmap, node-link diagram, geographic map, storyline	**visualise**
pattern[1]	regularity, trend, or structure identified in data	coherence, discrepancy, distinction, trend, motif, hot spot, concentration, sparsity, alignment	**abstract**
model	formal representation for prediction, simulation, or explanation	classifier, regression model, topic model, causal network	**build-model**
knowledge[2]	explicitly formulated understanding: insights, judgements, rules	summary statement, quality judgement, domain rules, established relationships	**generate-knowledge, assess,** or given
specification	control directives: parameters, constraints, method choices	number of clusters, threshold value, method selection	**generate-knowledge** or given

[1,2] **Pattern** vs. **knowledge**: A pattern describes an observed regularity within the data; it is descriptive, bounded by what is observed. Knowledge goes beyond observation to generalisations, inferred statements, judgements, conclusions, decisions.

Reference: Transform intents

Intent	What it does	Examples (manner)	Typical outputs
define-unit	creates or redefines units of analysis	partitioning, filtering, clustering, extraction, segmentation, sampling, synthetic data generation	**entities** (possibly + **feature**)
characterise	computes features of entities or relations	aggregation, summarisation, residual computation, feature engineering, distance computation, keyword detection	**feature**
contextualise	positions entities in a reference structure	temporal arrangement (time of day, weeks, calendar), projection (PCA, UMAP), geographic mapping	**arrangement**
visualise	creates a visual representation	visual encoding, rendering, layout, creation of UI with interaction controls	**visualisation**
abstract	derives patterns or conceptual structures	interpretation of visualisation or computational artifacts: clusters, arrangements, motifs, frequent combinations, topics	**pattern** (possibly + **feature**)
build-model	constructs or refines a formal model	training a classifier, fitting a topic model, calibration	**model**
generate-knowledge	formulates explicit knowledge or specifications	writing a summary, deriving rules, setting parameters, assigning semantic labels, defining concepts, annotating	**Knowledge** (substantive), **specification**
assess	evaluates quality or appropriateness	judging cluster homogeneity, assessing model fit, checking coverage, deciding about parameter adjustment	**knowledge** (evaluative), **specification**