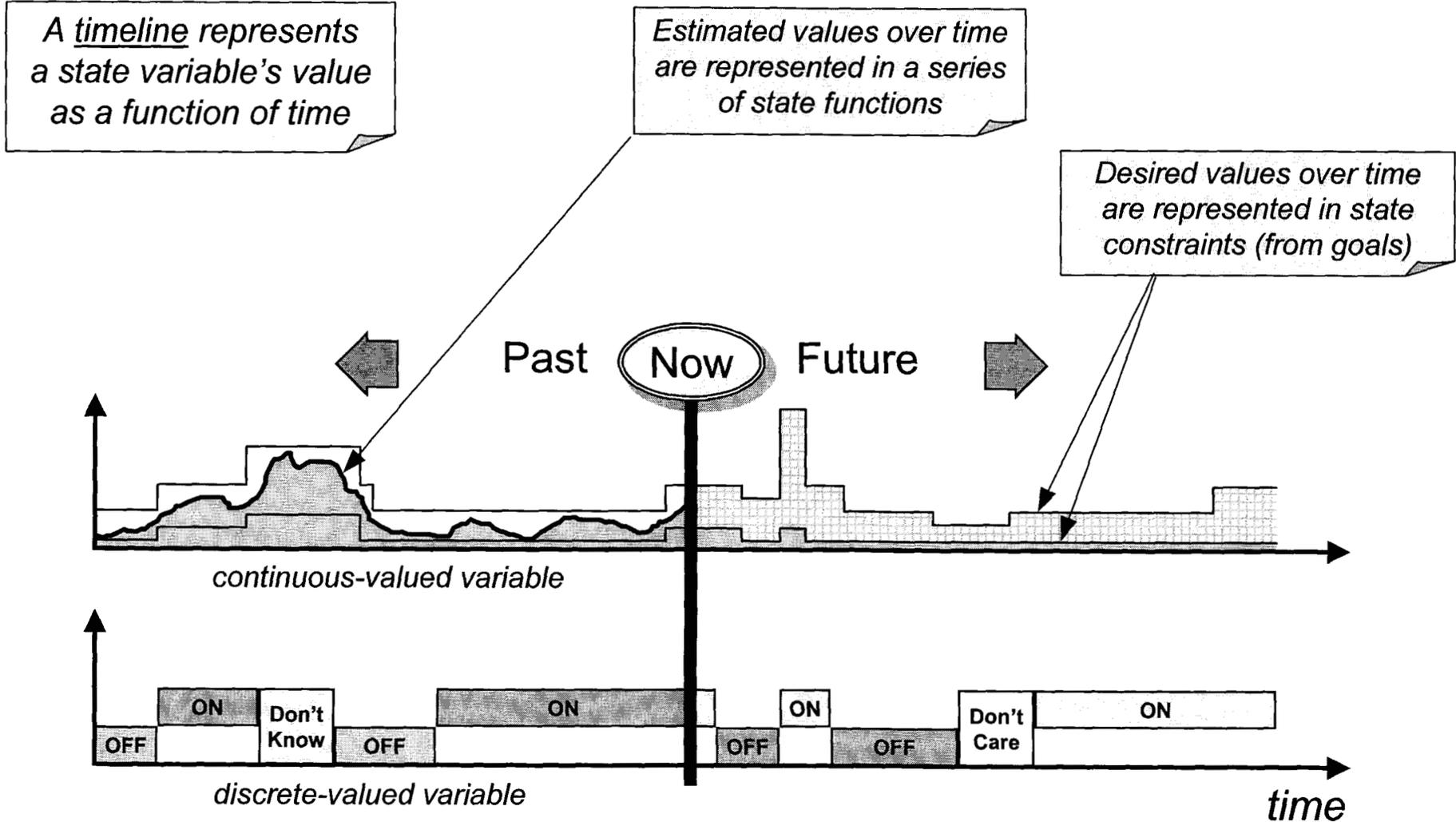






# A State Knowledge Timeline





# Value History



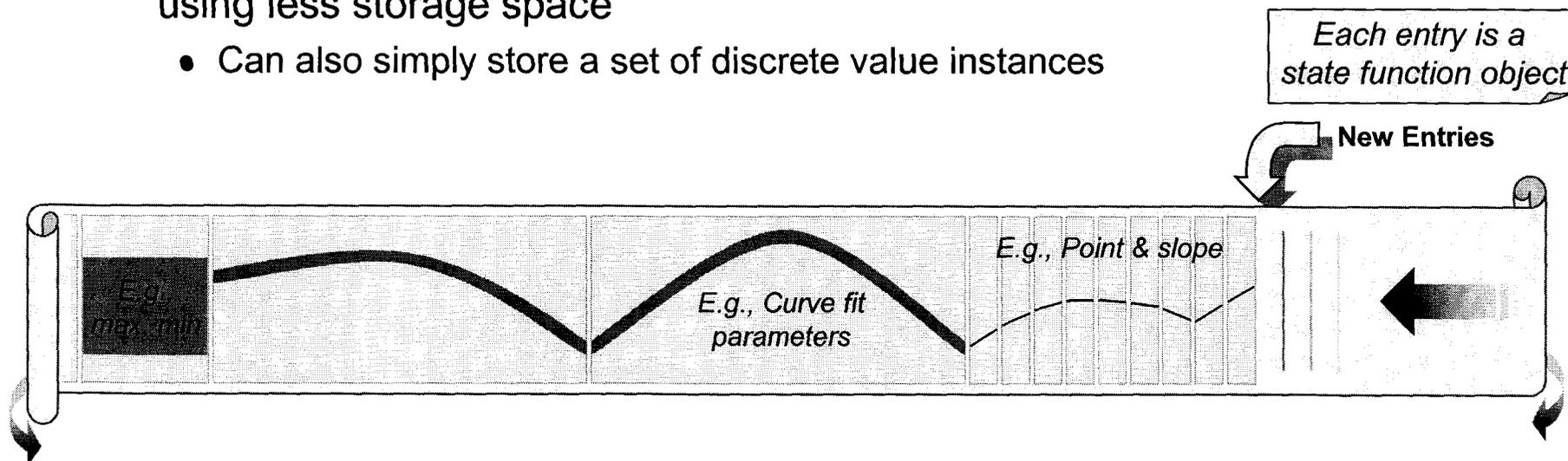
- **Every state represented by a State Variable uses the Value History service for storing its data**
- **Value History provides:**
  - Access to storage resources and telemetry
  - Rule-based ability to merge & compress value functions, checkpoint data, initialize at startup, generate telemetry
  - Consistent set of data interfaces on ground and flight
- **Also used in other components that generate data**
  - Sensors -> Measurements
  - Controllers -> Command Histories
- **Value histories group values and value functions into products for persistent storage and transport via the catalog service**



# Data Compression / Summarization



- “Intervallic” value histories
  - A container mechanism supporting functions that produce values over time
  - Encapsulate a “back-end” interface to data management persistent storage and data transport.
    - Can be stored and transported as data products
  - Leverage the use of models to preserve continuous information using less storage space
    - Can also simply store a set of discrete value instances



Entries are combined and compressed as they age, and are eventually deleted



# Catalog Service



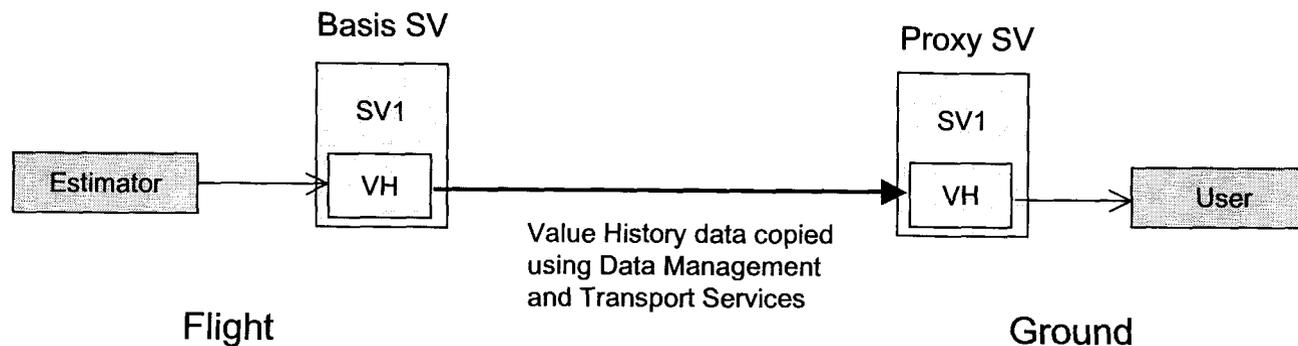
- 
- **Data Products are effectively files with descriptive metadata**
  - **The Catalog Service keeps track of these files and their metadata and provides a lookup service allowing applications to find products with matching attributes**
  - **The Value History Service is the primary user of the Catalog Service**
  
  - **Catalog is also the main user of persistent storage resources**



# Basis Histories and their Proxies



- **Public data interfaces are provided by State Architecture components:**
  - State Variables
  - HardwareAdapters
- **These components use (contain) one or more Value Histories to support the storage, retrieval, and management of their data.**
- **Basis components contain original data directly from hardware or estimators**
- **Proxy components contain copies of value history data**

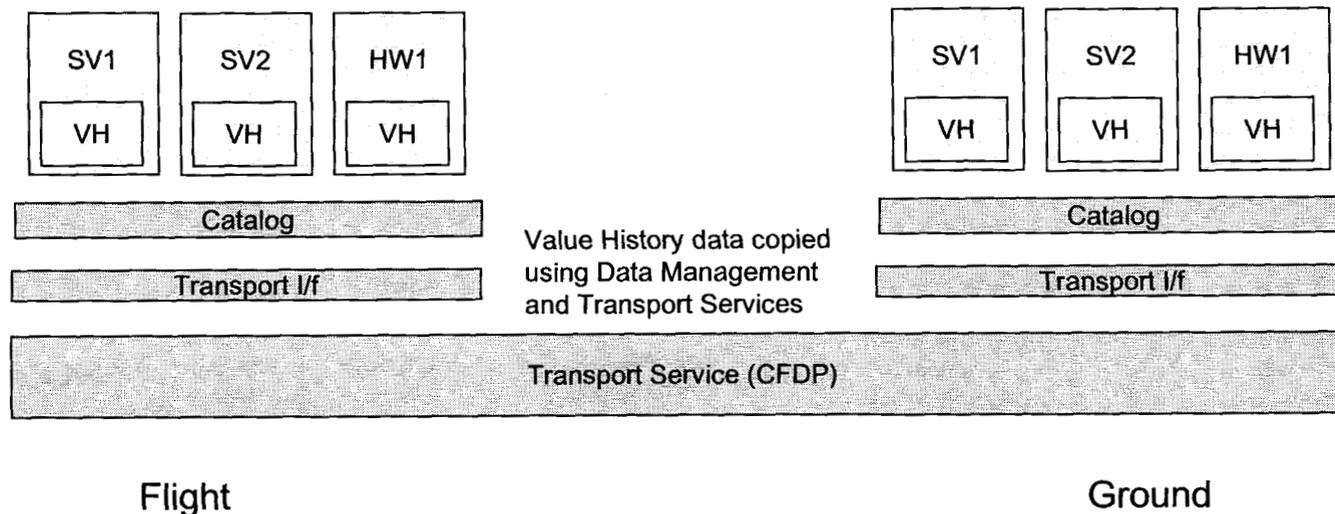




# Product-Oriented Transport

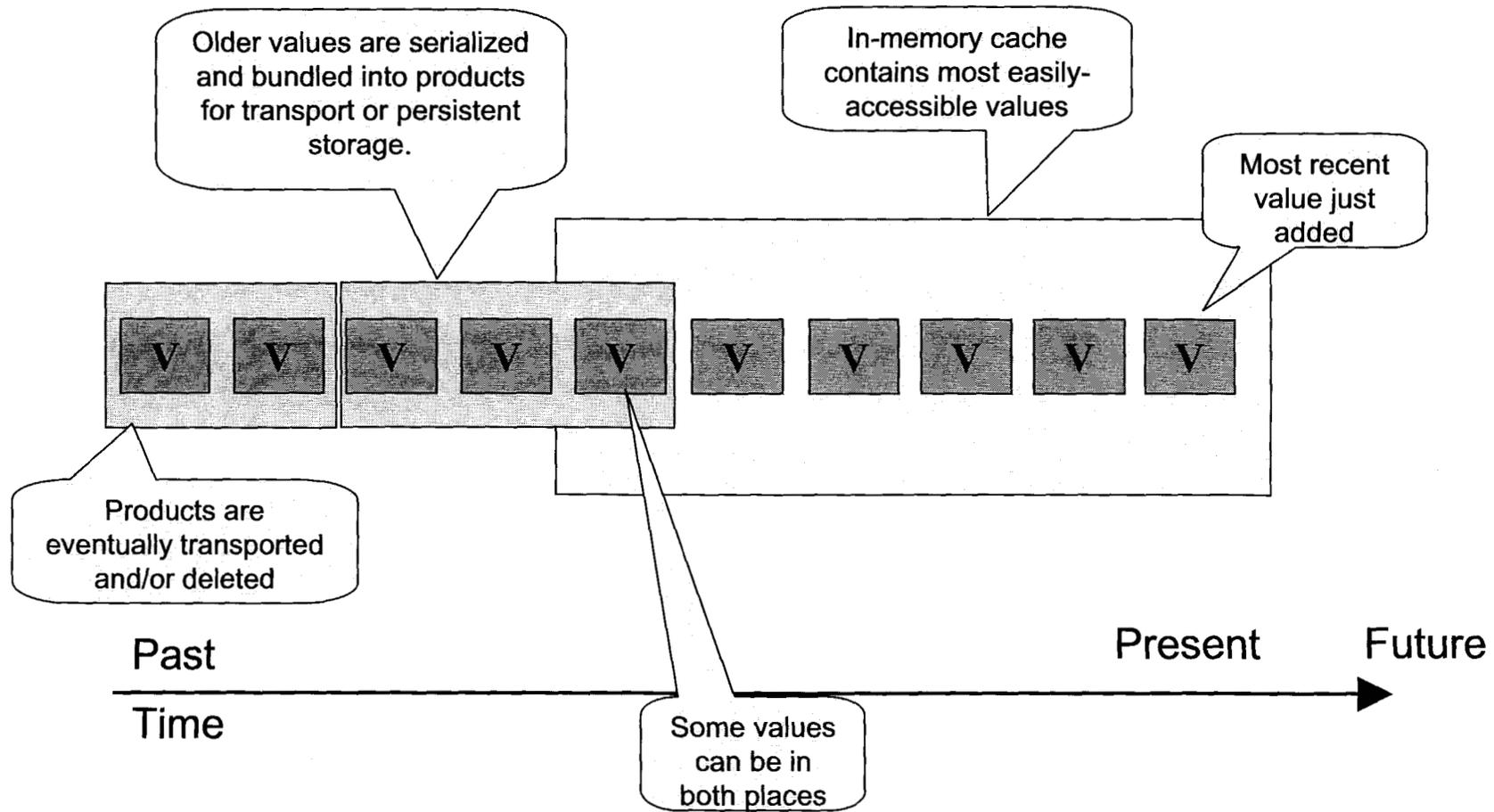


- **Transport service copies data products between catalog deployments**
  - Symmetrical protocols & interfaces
- **Designed with CCSDS File Delivery Protocol (CFDP) in mind**





# Value History Structure





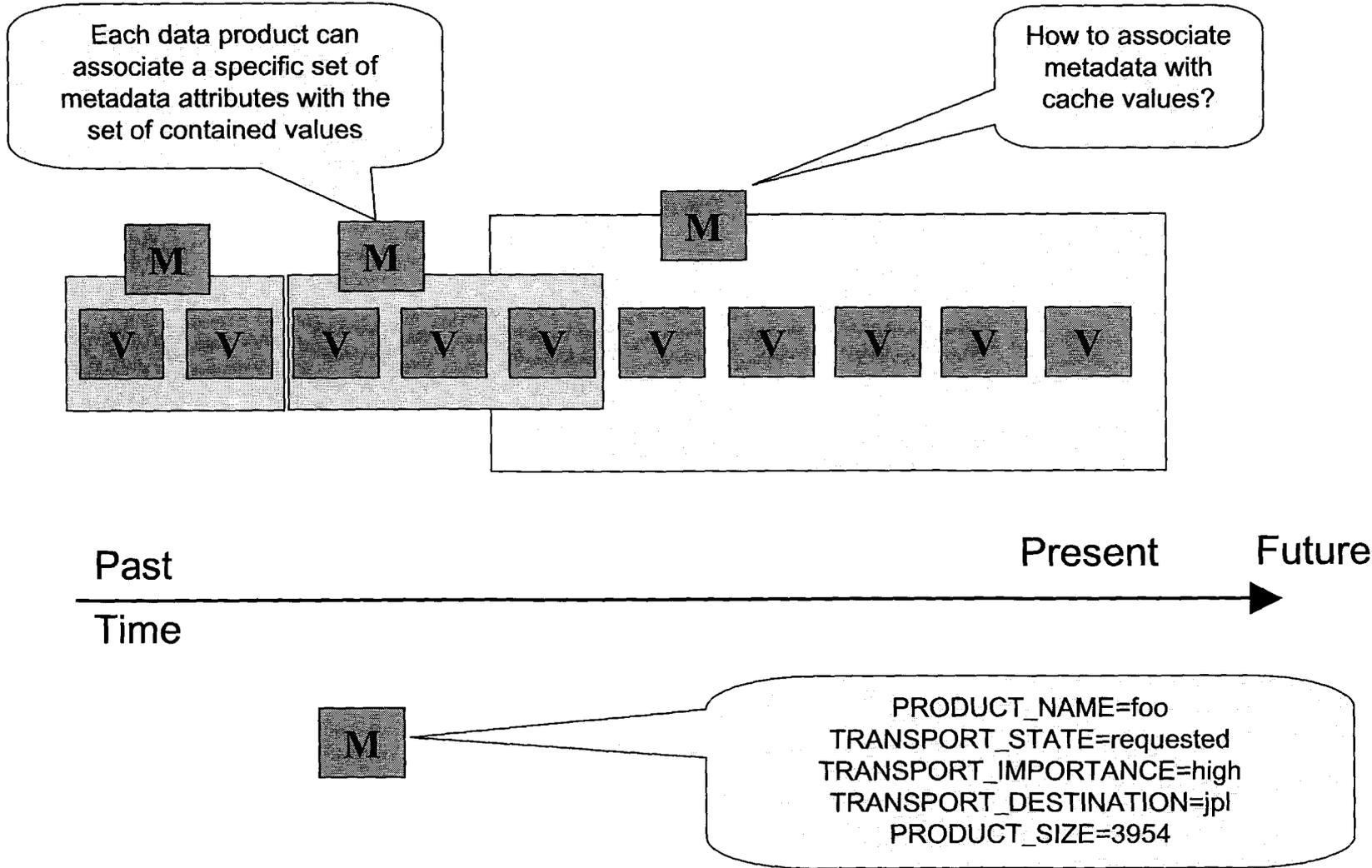
# Value History Metadata



- **Metadata** – refers generically to data about the data
- **The difference between data and metadata depends on scope of interest** – assume data management scope
- **Intrinsic attributes are essentially part of the measurement:**
  - Time tag: required for use in value history, so implemented in value base class
  - Individual value class can implement any attributes it needs internally, but the value history never looks inside the value class beyond the time tag, so any internal attributes are seen as part of the data.
- **Transient attributes are attributes associated with sets of data within a value history**
  - Used by DM to keep track of the values
  - Not part of the value class
  - Transient attribute values can change without affecting the data values they associate to



# Value History Metadata





## Controlling Data



- **Use goals to control significant states**
- **Use policies (mechanistic rules) to accomplish lower-level transactions**
- **Goals can change policies**
- **Goals can be used to effect policies on State Variables (their value history databases) to cause telemetry products to be created and marked for transport**
- **Data goals are usually specific (have a particular image, have values for a particular time period with a given resolution) but can be more abstract (have some data). When the outcome is more abstract and the goal period is long-term prefer to use simple policies. Use goals where there might be an alternate elaboration if it fails.**
- **Resource recovery can be automatic (resource management by goal). Based on product attributes.**



## MDS Design Features



- **Explicit units on all measurement and state values end-to-end**
- **Explicit representation of uncertainty and “unknown” values**
- **Explicit representation of value continuity (value functions) for states represented by state variables**
- **Flight-Ground data consistency through the use of common software classes to represent values and value functions**
- **Explicit reporting (via the goalnet) whether intentions (goals) were met (not just expectations)**
- **Value history framework enables semantic data compression**