Timeline as Unifying Concept for Spacecraft Operations

Kirk Reinholtz
Principal Engineer
Jet Propulsion Laboratory, California Institute Of Technology
Why Now, Why Timelines?

• The scene in 1990
  – O(10) CPU cycles telemetry bit
  – O(hour) of telemetry per PC disk
  – … And those CPU’s and disks were expensive
  – Internet? Sort of …
Why Now, Why Timelines?

• 1990’s architecture (still ubiquitous today)
  – Storage and processing not inline with display
    • Otherwise, no chance of timely display
  – Analysis “offline”
  – Weak information model
    • Lots of work, limited practical value in 90’s
  – Files and ad-hoc interfaces everywhere
    • Put in a database? Not feasible in 90’s
    • No room to store processed telemetry anyway
  – Limited use of databases
Why Now, Why Timelines?

• The scene today
  – O(10,000) CPU cycles per telemetry bit
  – O(10,000) hours of telemetry on PC disk
  – Disk and CPU way, way cheaper
  – Internet? Everywhere. Most tested protocols ever, probably
Why Now, Why Timelines?

- **Timelines** have always been there.
  - We just didn’t exploit that fact much till now
Fundamental Changes

• Formalize timelines
  – Catches the vast majority of data volume
  – Greatly reduces software costs

• Put them in databases (relational or other)
  – No more ad-hoc interfaces to manage
  – No more custom storage systems to develop and manage
  – Basic execution model is read-compute-write
Fundamental Changes

• Name and preserve each version
  – The name is then as good as the bits
  – Stashing and passing files not necessary
  – Eliminates A LOT of incidental machinery

• Naming Service
  – Maps name to current location
  – Supports migration, DB splitting/merging
  – Supports all repository types (timeline, triplestore, filestore, others)
Fundamental Changes

- Computation model
  - Massively scalable, auditable, repeatable computations on timelines
  - Highly regularized
    - Manipulate computations qua computations
  - Store a result, or rederive it, becomes an un-scary routine tactical choice
  - Scales to the Cloud
What's in the Paper

- Timeline Types
  - State, Measurement, Event, Activity…

- The SCN
  - Architectural commitment that EVERY mutation of EVERY timeline is in principle named and retained, immutable, forever. The SCN is how we do so with practical efficiency.
  - There are admin operations and associated theory for physical deletion of course.
Whats in the Paper

• Naming
  – Architectural commitment that all potential referents have a name (URI. Reference.)
    • The name is not the place. Place can be changed (split/merge/relocate repositories, etc)
    • The name itself is SCN’d so it can be changed without breaking older references

• Immutability Principle
  – Architectural commitment that referent is immutable

• Repository
Where We Are Today

• Programmatic
  – MGSS Program has committed to timelines for the new AMMOS architecture and implementation

• The Repository
  – Easy 10K samples/sec, continuous
    • Stress tested to 60K/s
  – … concurrent with Timebox queries 20K/sec
  – … concurrent with 20 Realtime Displays
    • deltaSCN query
  – Will soon demonstrate cloud version
  – Uses Oracle. Massive COTS leverage
Where We Are Today

• The Compute System
  – Demonstrated on “simplesat” use cases
  – Provides RESTful API to most capabilities
Where We Are Today

- Implemented elements of Info Model
  - Note these are also in the physical Repository
  - Measurement timeline
  - State timeline
  - Event timeline
  - Activity timeline
  - Triplestore (for metadata and relationships)
  - Filestore (for legacy and other unstructured data. Greatly eases migration to new architecture)
Where We Are Today

• Naming service
The End

Questions?

Thank you for Listening!