



# **Timeline and the Timeline Exchange Infrastructure: A Framework for Exchanging Temporal Information**

**Kenneth Donahue and Seung Chung**

**Mitch Ingham, John Day, Luke Walker, Allen Kummer, Ethan Post**

**Jet Propulsion Laboratory, California Institute of Technology**

**2013 IEEE Aerospace Conference  
Big Sky, Montana**



- **The Behavior Engineering Problem**
- **Motivation and Goals**
- **Timeline and the Timeline eXchange Infrastructure (TXI)**
- **SMAP Power Analysis Application**
- **Future Work**



- **Behavior Engineering is...**
  - The discipline that works to specify *how* a system can act, both *nominally* and *off-nominally*.
  
- **As a discipline, Behavior Engineering needs improvement:**
  - It is **implemented differently** from project to project,
  - It **lacks formalism**, and
  - Its information is captured in a **plethora of sources**, e.g.
    - Requirements Documents
    - Functional Description Documents
    - Interface Control Documents
    - Mission Plans



## **Model-Based Systems Engineering (MBSE) Motivation:**

- **Provide Flight System Engineers with a common and consistent methodology and repository for capturing behavior-related information**

## **Pilot Goals:**

- **Fold MBSE into JPL's current systems engineering practices**
- **Develop better products and processes to improve organizational and lifecycle handoffs, specifically in system behavior specification**



## **Timeline and the Timeline eXchange Infrastructure (TXI)**

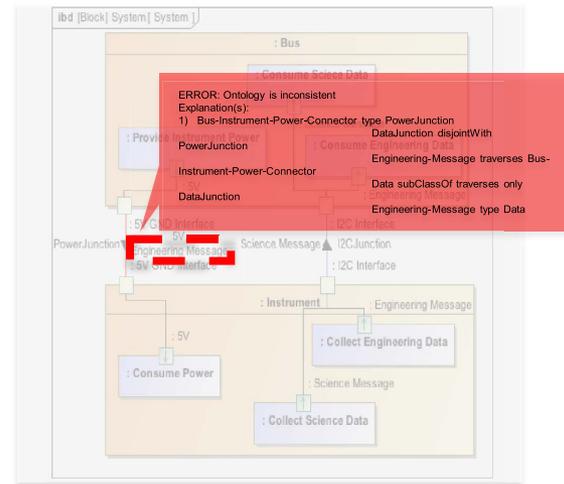
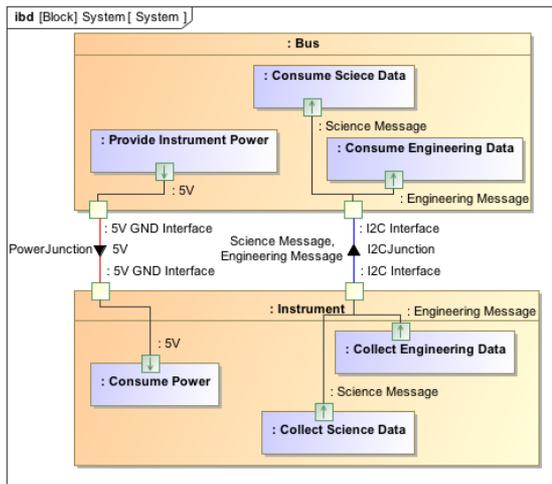


- **To solve the lack-of-formalism issue with Behavior Modeling, we created a Timeline Ontology**
- **A Timeline is**
  - A set of Events
  - A set of Constraints that are valid during those Events
  - A set of Constraints that relate the Events to one another (temporally)
- **We found this representation to be sufficient to capture the types of behavior analyzed early in the project life-cycle**
- **We believe this can be extended to deal with more complex behaviors (e.g. conditional branches and loops)**

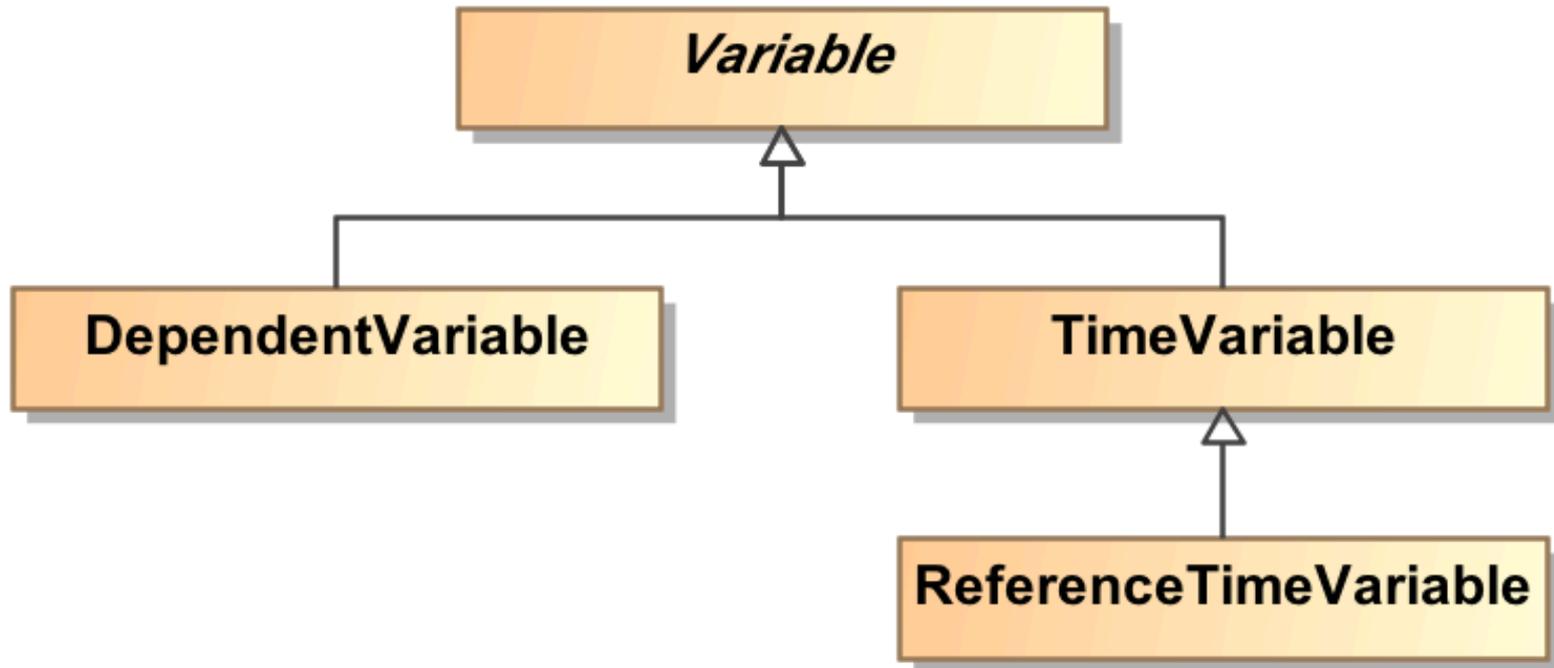
# Benefit of Ontologies - Formal Reasoning



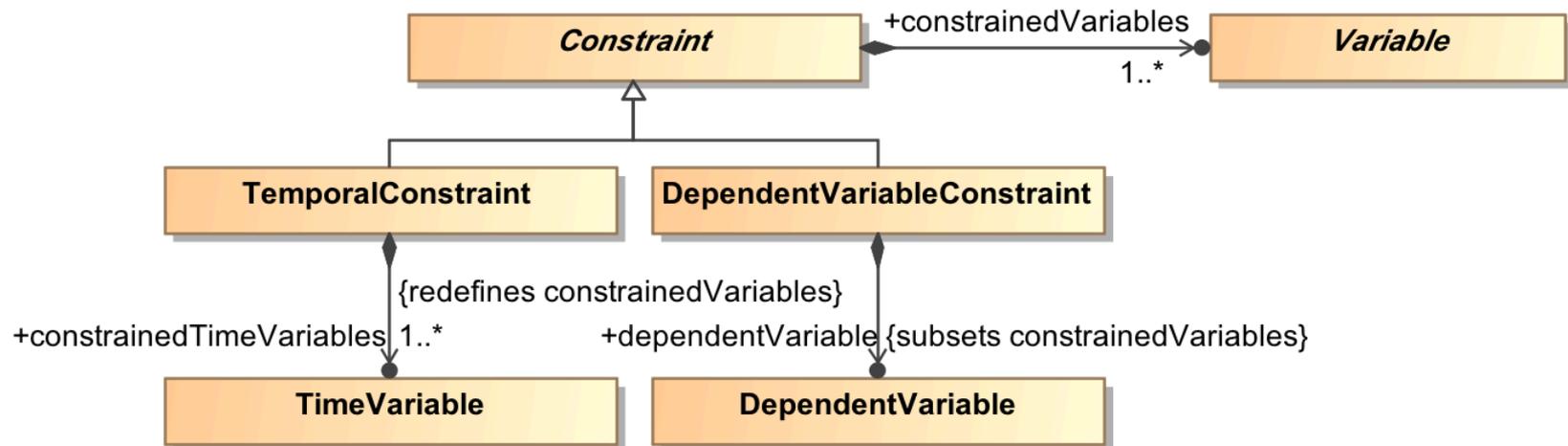
- Pellet is an open source OWL 2 reasoner
- It provides standard and cutting-edge reasoning services for OWL ontologies
- It looks at OWL axioms and checks for satisfiability and consistency
- Basically, if we make a false assertion, Pellet can tell us what we did wrong



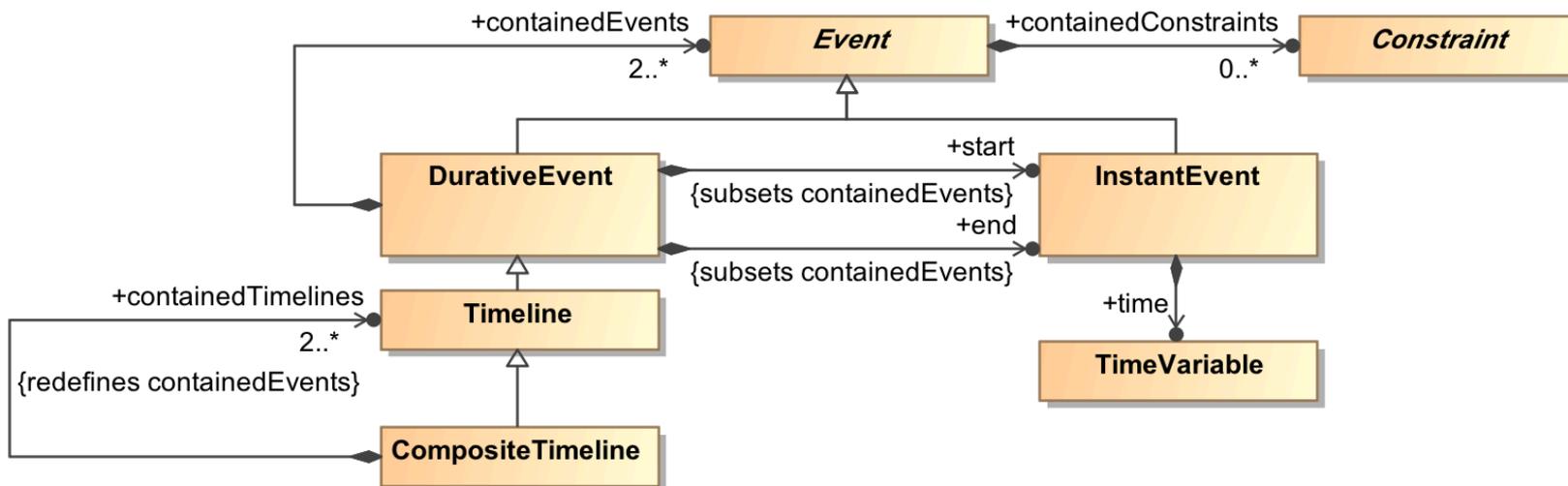
# Timeline Ontology (1/3) - Terms



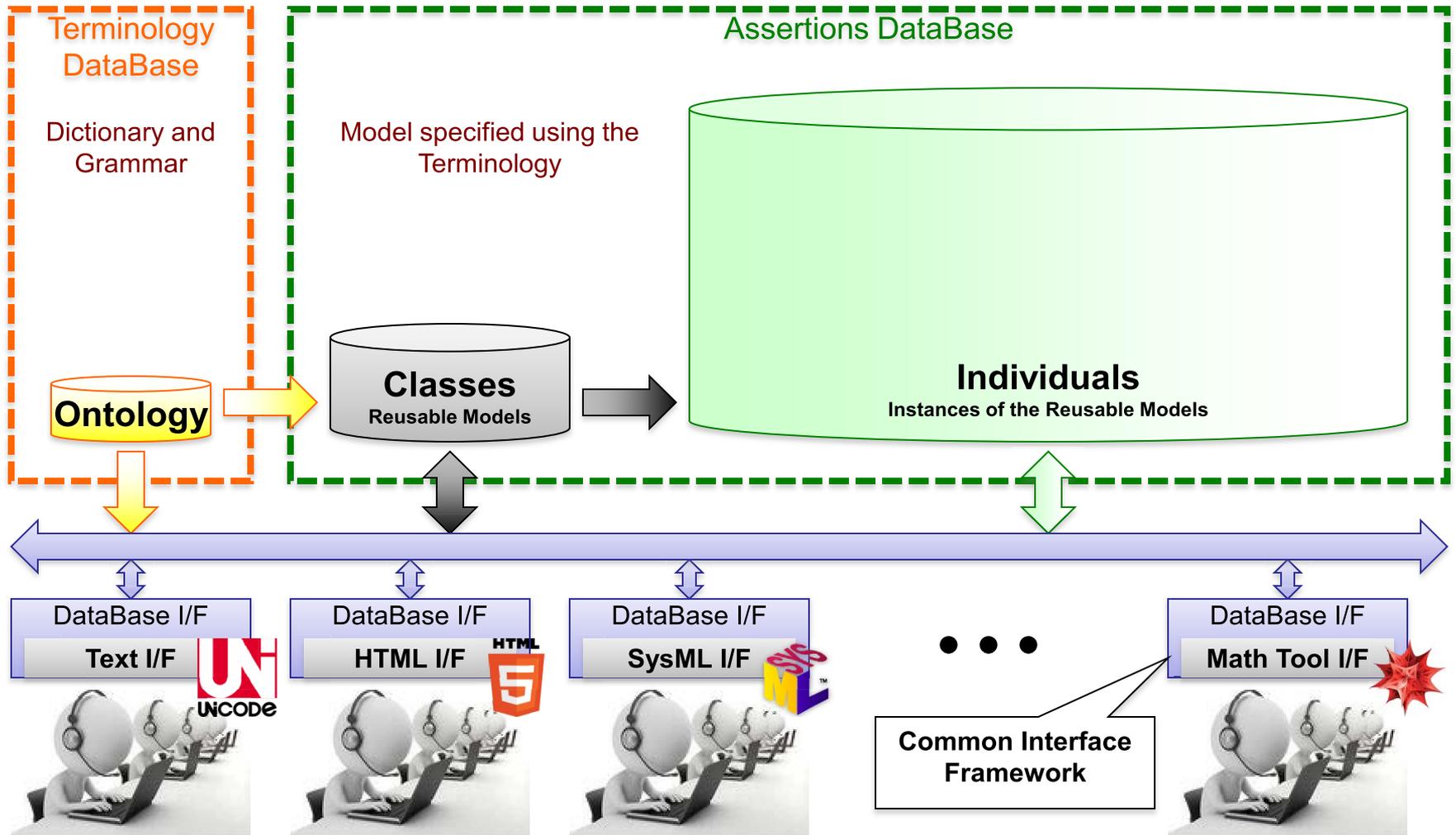
# Timeline Ontology (2/3) - Constraints



# Timeline Ontology (3/3) - Events



# Timeline eXchange Infrastructure (TXI) (1/2)



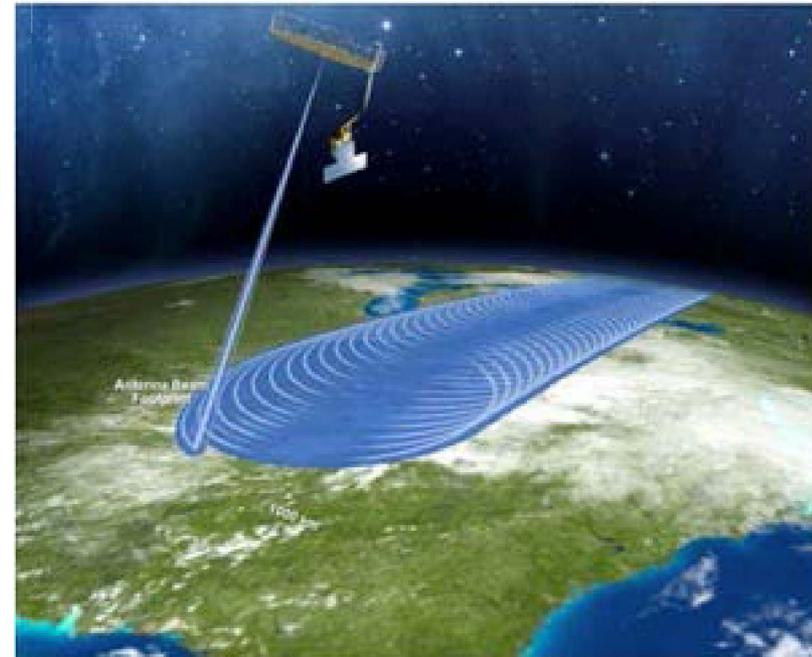


- **TXI basically consists of two things**
  - 1) an OpenRDF Sesame Triplestore
  - 2) an Application that can interact with the Sesame Triplestore
- **We created a Java JAR that can be incorporated into any Java-based application. The JAR provides many Timeline-related functions**
  - Timeline construction from database query
  - Timeline push to database
  - Timeline to/from RDFXML file
  - Timeline to/from Timeline DSL file



# SMAP Power Analysis Application

- **One of four first-tier missions recommended by the Earth Science Decadal Survey**
- **Provide global mapping of soil moisture and freeze/thaw state to**
  - Link water, energy, and carbon cycles
  - Estimate water and energy fluxes at land surface
  - Quantify net carbon flux in boreal landscapes
  - Enhance weather and climate forecast skill
  - Develop improved flood prediction and drought monitoring capabilities



- **The Instrument includes:**
  - A Radiometer,
  - A Synthetic Aperture Radar (SAR),
  - A conically-scanning deployable mesh reflector
- **For this study, we focused on the Launch Scenario.**

# Application – SMAP Power Analysis



«activity» Power Analysis Activity Prototype
powerContingency : W
powerAllocation : W
powerCBE : W
powerCBEPlusContingency : W

- **Setup:**

- Mission has a Launch Scenario
- We can create a notional (nominal) realization of that Launch Scenario using SysML Activities
- Each Component in the Flight System performs Activities, each with their own power properties.
- The leaf-level Actions in the Launch Scenario Activity decomposition are Actions with behaviors that are the aforementioned Activities with the power-related properties

- **Information Extraction:**

- Using this model, we can extract a Power Profile for each component
- Leaf component profiles can be merged to create assembly-level, subsystem-level, or system-level profiles, following the various decomposition relationships in the model.





- **Basic Transformation:**

- Scenario (root Activity) → Timeline
- Action → DurativeEvent
- InitialNode → InstantEvent
- FinalNode → InstantEvent
- ActivityEdge → TemporalConstraint
- Activity/Action Constraint → TemporalConstraint
- Activity Power Property → DependentVariableConstraint

```
<owl:NamedIndividual rdf:about="&LukeDemoLaunchScenario;LukeDemoLaunchScenario.startConstraint.1">  
  <rdf:type rdf:resource="&timeline;TemporalConstraint"/>  
  <dc:description rdf:datatype="&xsd:string">null</dc:description>  
  <timeline:constrains rdf:resource="&LukeDemoLaunchScenario;LukeDemoLaunchScenario"/>  
  <timeline:couplesTerm rdf:resource="&LukeDemoLaunchScenario;LukeDemoLaunchScenario.Start.Time"/>  
  <timeline:hasText rdf:datatype="&xsd:string">LukeDemoLaunchScenario.Start.Time = -1440.0</time  
</owl:NamedIndividual>
```



- **Basic Transformation:**

- Timelines lend themselves to piecewise expression representation
- Construct the piecewise expression for each component
- All constraints are already in the Timeline, so just extract them and solve them in the right order.

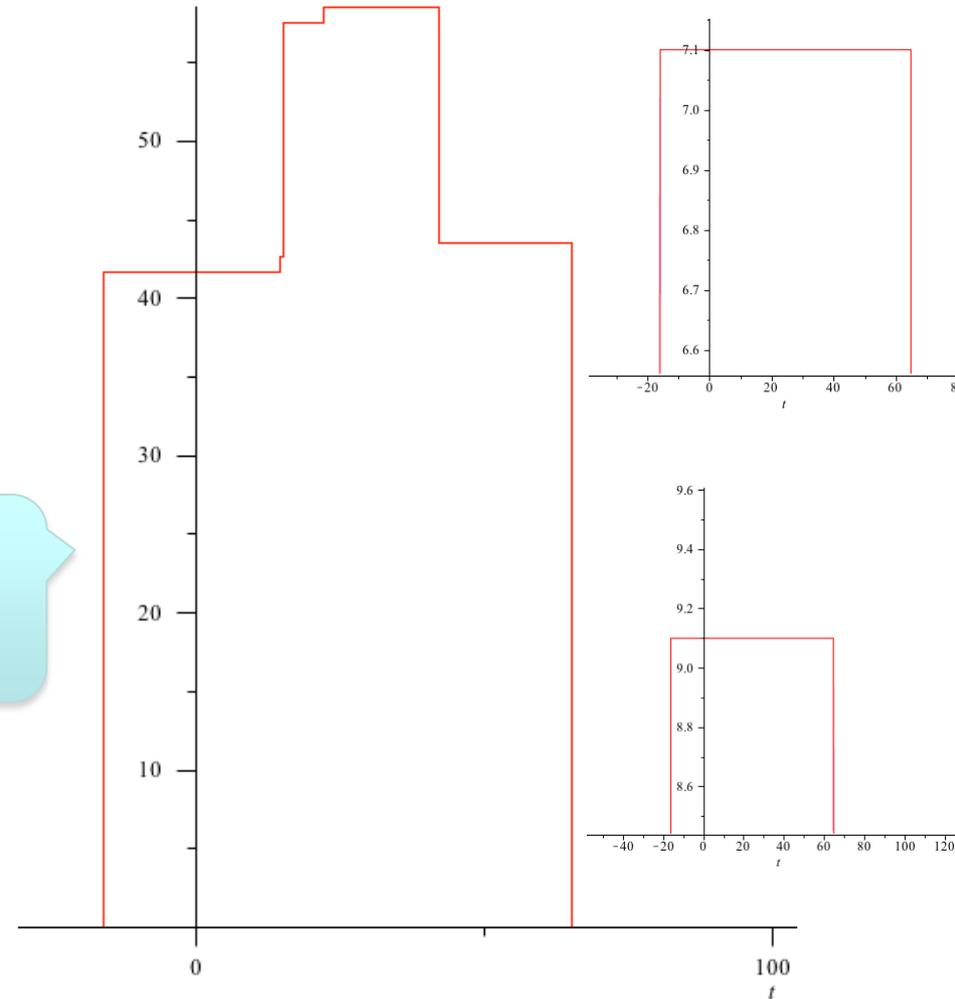
## Example Code:

```
Power[Component] := piecewise(t0 < t and t < t2,  
                                p0,  
                                0) +  
piecewise(t3 < t and t < t4,  
                                p1,  
                                0) +  
...
```

- **Resulting plot was compared to that generated by current methods**

Power and Pyrotechnics during Launch

- **Results were identical**





- **Extension of Timeline Ontology to include special types of Timelines (e.g. state trajectory Timelines)**
- **Extension of Timeline Ontology to include special types of Events (e.g. loops, conditional branches)**
- **SysML-to-Timeline Transformation generalization to allow for many DependentVariables instead of just one.**
- **Extension of TXI to include Application for visualizing and editing Timelines (without SysML)**



- **Representing Behavior via Timelines works**
- **Timelines can be automatically checked for semantic correctness**
- **We can perform analyses on Timelines using the Timeline eXchange Infrastructure**
- **Analyses performed produce the same results**
- **Analyses are done dynamically from information from a remote database, so no need to worry about stale inputs.**

# Backup



# Sample SPARQL Query



```
//create a graph of contained events  
query ="PREFIX tl: " + timelineURI+"\n" +  
      "CONSTRUCT { ?sub tl:containsEvent ?obj }\n" +  
      "WHERE { ?sub tl:containsEvent ?obj }";
```

# Sample SPARQL Query



**//create a graph of contained events**

**query = "PREFIX tl: " + timelineURI + "\n" +**

**//can have many prefixes if using many ontologies**

**"CONSTRUCT { ?sub tl:containsEvent ?obj \n"**

**//can have many edges in this graph**

**"}\n" +**

**"WHERE { ?sub tl:containsEvent ?obj\n"**

**//Can have many basic graph patterns to match**

**// "FILTER regex(?sub, \"ExampleStartString\")"**

**//Can have many filters on the elements that**

**//match the graph pattern**

**"}";**