A New Open Standard: Lifecycle Modeling Language (LML) a Language for Simple, Rapid Development, Operations and Support

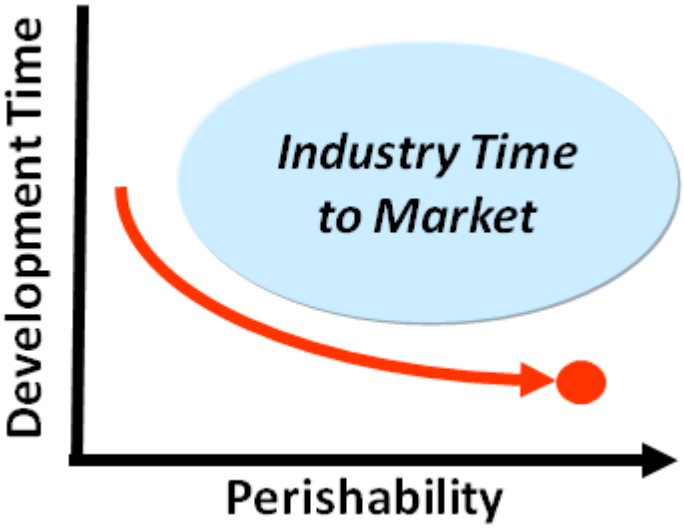
January 25, 2014

Steven H. Dam, Ph.D., ESEP
President, SPEC Innovations
571-485-7805
steven.dam@specinnovations.com

Warren K. Vaneman, Ph.D.
Naval Postgraduate School
wvaneman@nps.edu

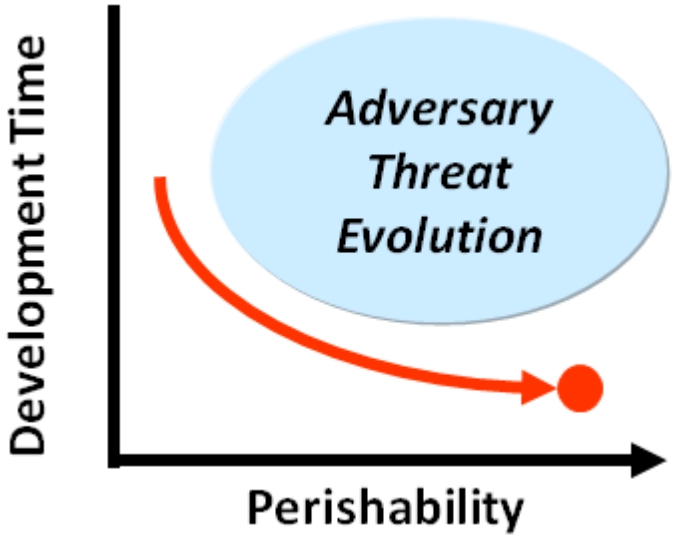
Why a New Language?

Current Environment



Industry Development and Endurance Timelines

- Industry has embraced the benefits of global technology.
- Development time has decreased.
- Perishability/obsolescence increasing

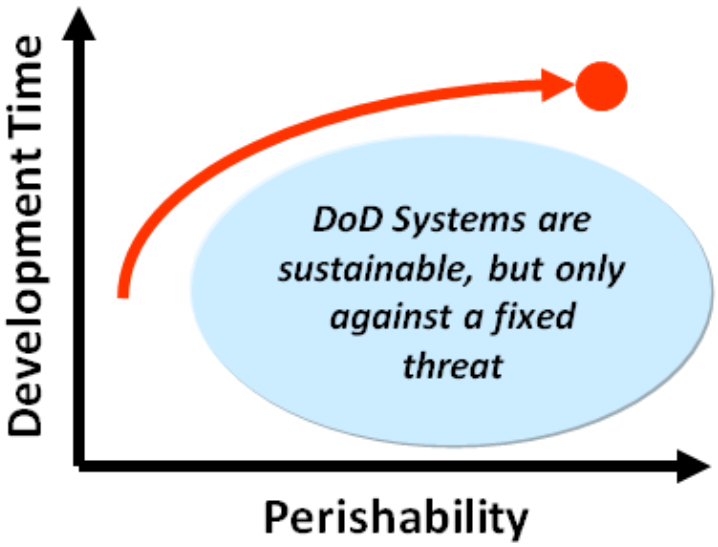


Adversary Development and Counter-Measure Timelines

- Adversaries are also leveraging COTS technology.
- Emerging threats demand we enhance the speed in which we deliver new capabilities.

SOURCE: Systems 2020 Study Final Report, Booz Allen Hamilton, August 2010

Current Environment



DoD Development and Endurance Timelines

- Long development times
- Good for fixed threats.
- However, against asymmetric threats perishability/obsolescence increasing.

U.S. is Faced with Highly Unpredictable, but Surmountable Threats

- Acquisition methods designed for warfare during the Cold War are ill-suited for the dynamic nature of threats today.

SOURCE: Systems 2020 Study Final Report, Booz Allen Hamilton, August 2010

Complex Systems Implications for Systems Engineering

- Larger, more complex systems development implies:
  - A need for larger, distributed teams
  - A need for a clear concise way to express the system design (clear, logically consistent semantics)
  - New tools to enable collaboration across the entire lifecycle

Complexity has been identified by many as a critical problem facing system engineers

Complexity

Complexity is a Major Issue

- Integration of systems create a major problem with complexity
  - Within in a system, interactions grow as N squared or worse
  - Ability to understand and test becomes less certain
  - As more systems are added, the interfaces grow in a non-linear fashion
  - Many of the existing systems are old and not built for these interfaces
  - Conflicting or missing interface standards make it hard to define interface interactions
  - Hardware and software may be re-purposed and "heritage" compromised
  - Future systems will be integrated from multi-organizational, multi-national contributions, adding additional layers of complexity
- Systems engineering must deal with this complexity
  - End-to-end systems engineering is needed, including "reengineering" of old systems
  - Robust M&S, verification and validation testing are A must

Page 8

From a presentation by Dr. Michael Ryschkewitsch,
NASA Chief Engineer, at CSER Conference 15 April 2011

- With the growth of the Internet and daily changes in IT, systems have become more complex and change more rapid than ever before
- Systems engineering methods have not kept up with these changes
- SE has been relegated to the beginning of the lifecycle

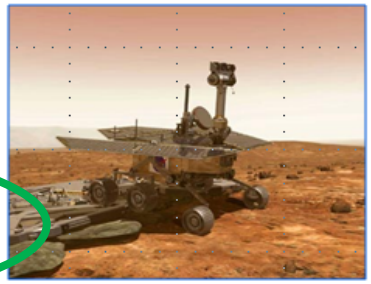
How Does SE Typically Respond to Complexity?

- Focus on upper-left hand side of the “Vee”
- Systems Architecture and Requirements Development
  - More complex MBSE languages
  - More complex processes and governance
- More layers of abstraction
  - “Systems of Systems”
  - “Family of Systems”
  - “Portfolio Management”
- Need more time and money!

More Money Is a Problem

Use and Challenges vary throughout the life cycle

- Early Phases; conduct Systems Analysis and trade studies
  - Exercise through CONOPS
  - Rapidly dismiss faulty options
  - Develop feasible, cost effective options
  - Identify advanced technology requirements
  - How to enable modeling that provides the needed fidelity yet can be done quickly and cheaply?
  - Current methods tend to be "wetware" intense.
  - Need rapid and effective teaming



- Development
  - Refine Design, support Validation & Verification
  - Enhance manufacturing
  - How to better enable integration of discipline oriented design tools into systems models that capture functional and performance behaviors
  - How to capture system design rationale, assumptions and other "background" data.
  - How do we develop the standards that allow lossless integration across organization and tool boundaries?
- Operations
  - Provide data for ops team, resolve in flight issues, address parts obsolescence
  - How do we make the full suite of information captured during design and development available to the operators without having prior knowledge of their needs?

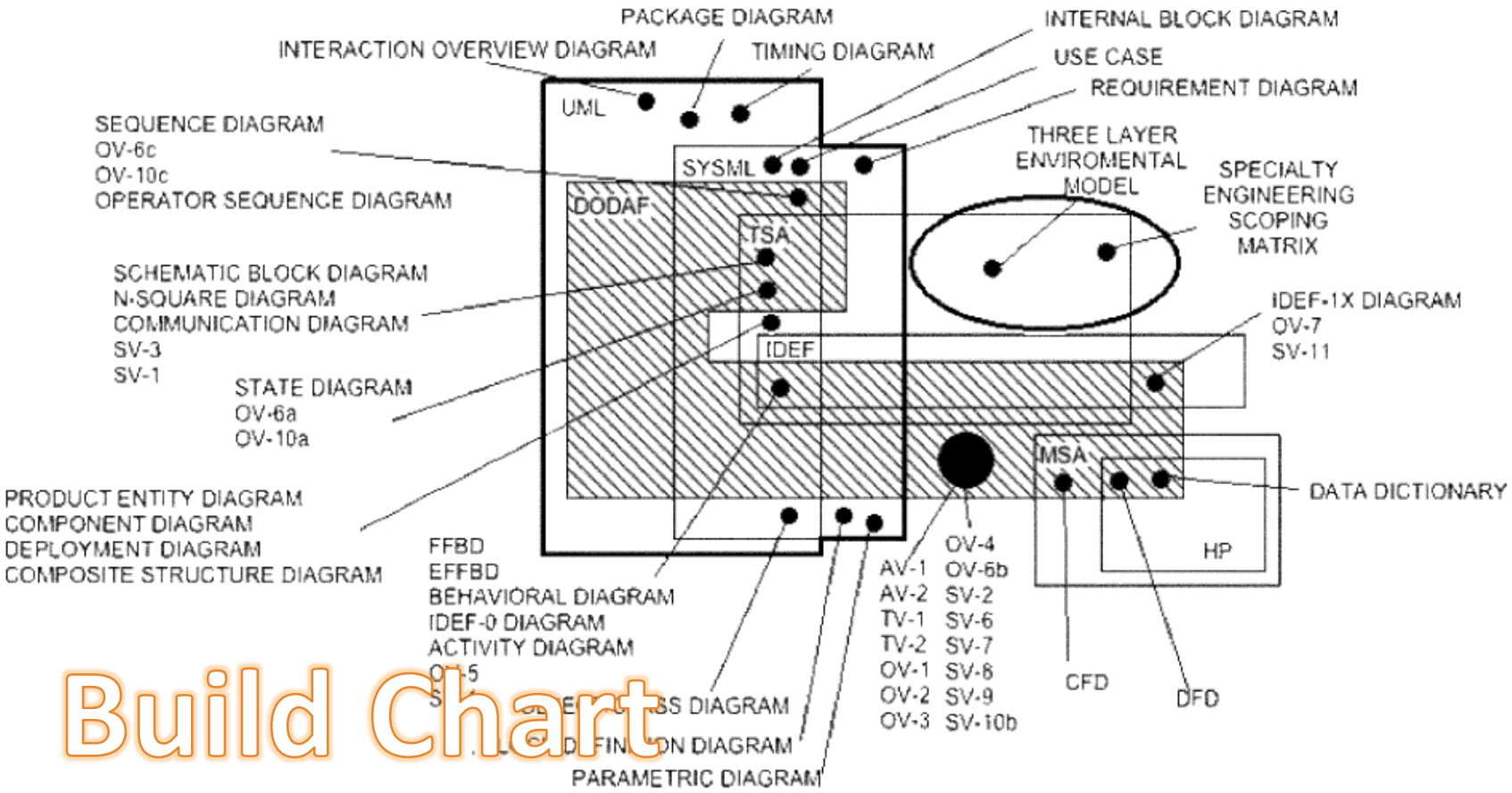
Page 12

From a presentation by Dr. Michael Ryschkewitsch,
NASA Chief Engineer, at CSER Conference 15 April 2011

- Calls for doing more with less continue
- Need for lower labor and tool costs essential for acceptance of SE across the lifecycle

How can we simplify things enable "quicker/ cheaper"? Let's start with the language we use

Overlap Among MBSE Techniques



SOURCE: Grady. J.O. (2009). *Universal Architecture Description Framework,* INCOSE.

State of Current "Languages"

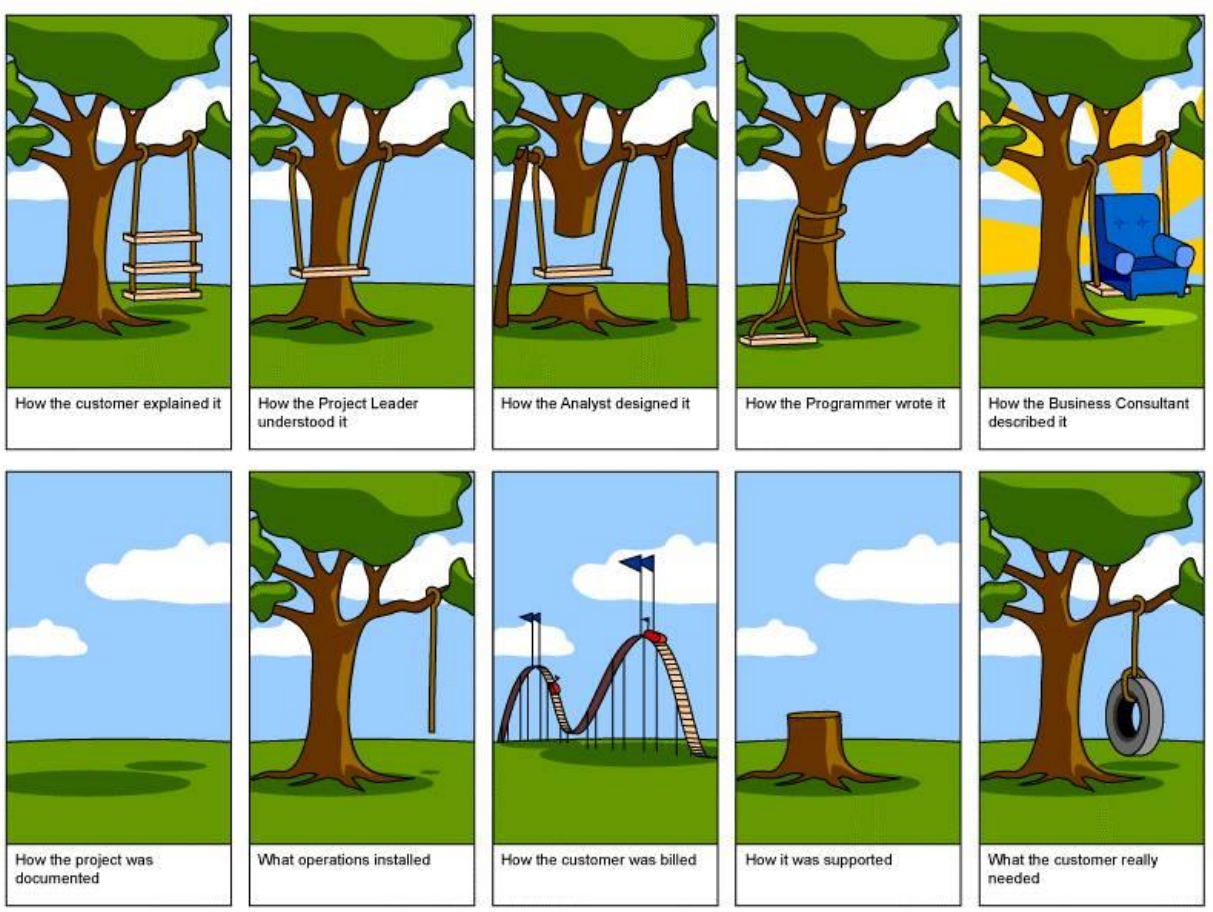
- In the past decade, the Unified Modeling Language (UML) and the Systems Modeling Language (SysML) have dominated the discussion
- Why?
  - Perception that software is "the problem"
  - Hence need for an "object" approach
- SysML was designed to relate systems thinking to software development, thus improving communication between systems engineers (SE) and software developers

Why Objects Are Not the Answer

- Although SysML *may* improve the communication of design between SEs and the software developers it does not communicate well to anyone else
  - No other discipline in the lifecycle uses object oriented design and analysis extensively
  - Users in particular have little interest/acceptance of this technique
  - Software developers who have adopted Agile programming techniques want functional requirements (and resent SEs trying to write software)
  - Many software languages are hybrid object and functional

So What Do We Do?

- Recognize that our primary job as SEs is to communicate between all stakeholders in the lifecycle
- Be prepared to translate between all the disciplines
- Reduce complexity in our language to facilitate communication



Lifecycle Modeling Language (LML) Overview

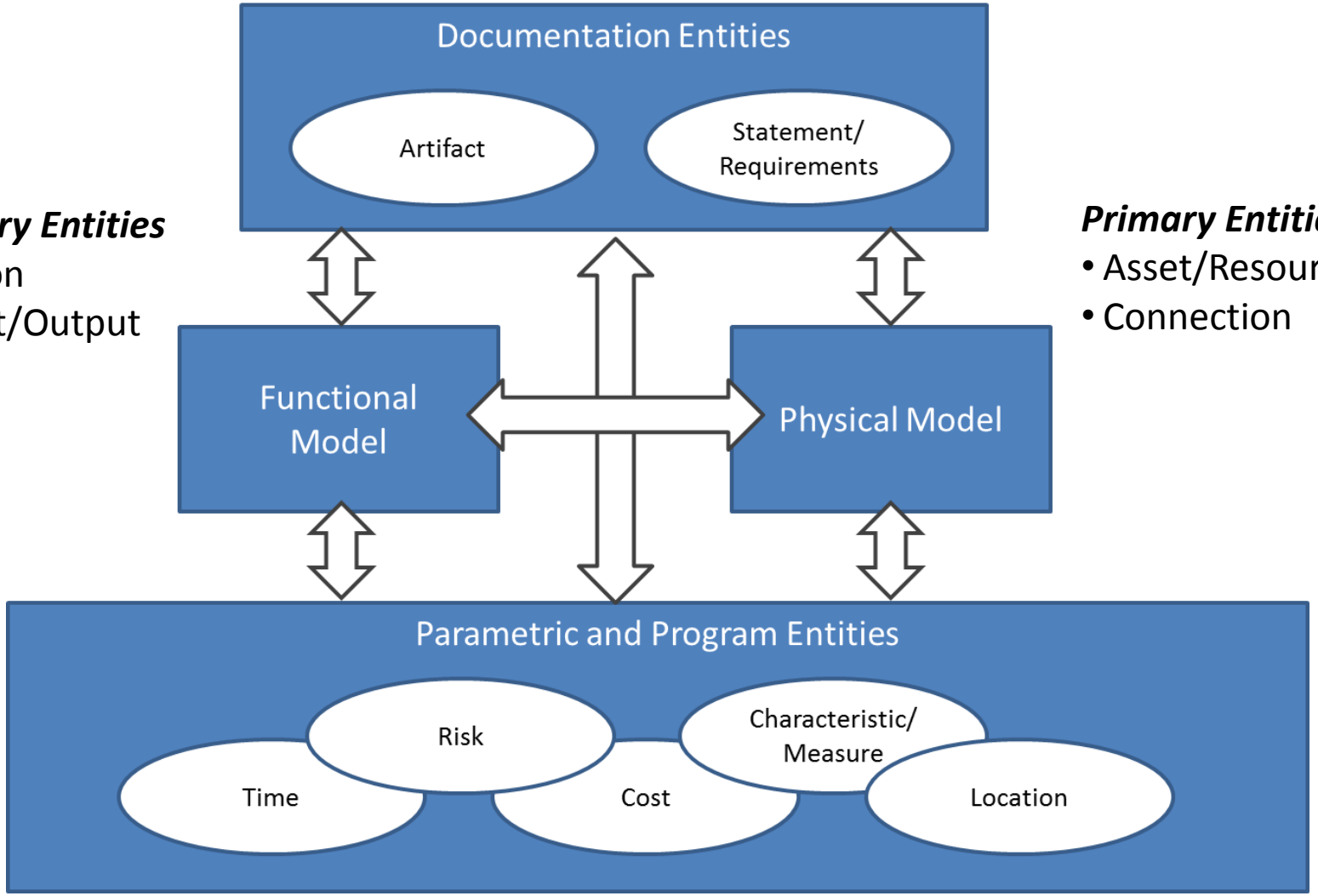
Lifecycle Modeling Language (LML)

- LML combines the logical constructs with an ontology to capture information
  - SysML – mainly constructs – limited ontology
  - DoDAF Metamodel 2.0 (DM2) ontology only
- LML simplifies both the “constructs” and “ontology” to make them more complete, yet easier to use

***Goal:** A language that works across the full lifecycle*

LML Models

Primary Entities
- Action
- Input/Output



Primary Entities
- Asset/Resource
- Connection

LML Ontology* Overview

- Taxonomy**:
  - 12 primary entity classes
  - Many types of each entity class
    - Action (types = Function, Activity, Task, etc.)
- Relationships: almost all classes related to each other and themselves with consistent words
  - Asset performs Action/Action performed by Asset
  - Hierarchies: decomposed by/decomposes
  - Peer-to-Peer: related to/relates

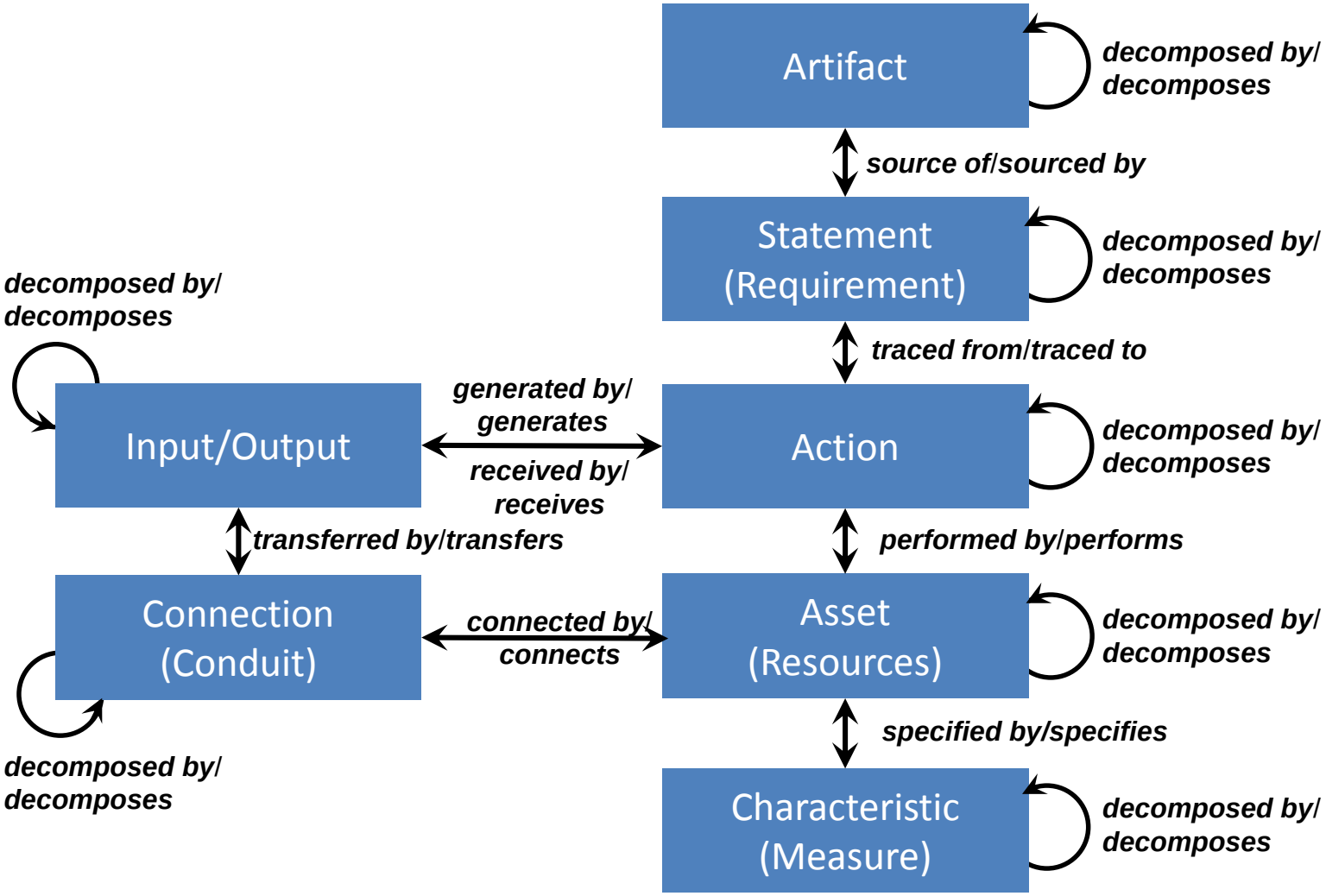
*Ontology = Taxonomy + relationships among terms and concepts

** Taxonomy = Collection of standardized, defined terms or concepts

LML Schema Elements

- Action
- Artifact
- Asset
  - Resource
- Characteristic
  - Measure
- Connection
  - Logical
  - Conduit
- Cost
- Input/Output
- Location
  - Physical
  - Orbital
  - Virtual
- Risk
- Statement
  - Requirement
  - Decision
- Time

LML Primary Entities and Relationships



LML Relationships Provide Linkage Needed Between the Classes

- decomposed by/decomposes
- orbited by/orbits
- related to/relates

	Action	Artifact	Asset (Resource)	Characteristic (Measure)	Connection (Conduit, Logical)	Cost	Decision	Input/Output	Location (Orbital, Physical, Virtual)	Risk	Statement (Requirement)	
Action	decomposed by* related to*	references	(consumes) performed by (produces) (seizes)	specified by	-	incurs	enables results in	generates receives	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Artifact	referenced by	decomposed by* related to*	referenced by	referenced by specified by	defines protocol for referenced by	incurs referenced by	enables referenced by results in	referenced by	located at	causes mitigates referenced by resolves	referenced by (satisfies) source of traced from (verifies)	occurs
Asset (Resource)	(consumed by) performs (produced by) (seized by)	references	decomposed by* orbited by* related to*	specified by	connected by	incurs	enables made responds to results in	-	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Characteristic (Measure)	specifies	references specifies	specifies	decomposed by* related to* specified by*	specifies	incurs specifies	enables results in specifies	specifies	located at specifies	causes mitigates resolves specifies	(satisfies) spacifies traced from (verifies)	occurs specifies
Connection (Conduit, Logical)	-	defined protocol by references	connects to	specified by	decomposed by* joined by* related to*	incurs	enables results in	transfers	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Cost	incurred by	incurred by references	incurred by	incurred by specified by	incurred by	decomposed by* related to*	enables incurred by results in	incurred by	located at	causes incurred by mitigates resolves	incurred by (satisfies) traced from (verifies)	occurs
Decision	enabled by result of	enabled by references result of	enabled by made by responded by result of	enabled by result of specified by	enabled by result of	enabled by incurs result of	decomposed by* related to*	enabled by result of	located at	causes enabled by mitigated by result of resolves	alternative enabled by traced from result of	date resolved by decision due occurs
Input/Output	generated by received by	references	-	specified by	transferred by	incurs	enables results in	decomposed by* related to*	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Location (Orbital, Physical, Logical)	locates	locates	locates	locates specified by	locates	locates	locates	locates	decomposed by* related to*	locates mitigates	locates (satisfies) traced from (verifies)	occurs
Risk	caused by mitigated by resolved by	caused by mitigated by references resolved by	caused by mitigated by resolved by	caused by mitigated by resolved by specified by	caused by mitigated by resolved by	caused by incurs mitigated by resolved by	caused by enables mitigated by results in resolved by	caused by mitigated by resolved by	located at mitigated by	caused by* decomposed by* related to* resolved by*	caused by mitigated by resolved by	occurs mitigated by
Statement (Requirement)	(satisfied by) traced to (verified by)	references (satisfied by) sourced by traced to (verified by)	(satisfied by) traced to (verified by)	(satisfied by) specified by traced to (verified by)	(satisfied by) traced to (verified by)	incurs (satisfied by) traced to (verified by)	alternative of enables traced to results in	(satisfied by) traced to (verified by)	located at (satisfied by) traced to (verified by)	causes mitigates resolves	decomposed by* traced to* related to*	occurs (satisfied by) (verified by)
Time	occurred by	occurred by	occurred by	occurred by specified by	occurred by	occurred by	date resolves decided by occurred by	occurred by	occurred by	occurred by mitigates	occurred by (satisfies) (verifies)	decomposed by* related to*

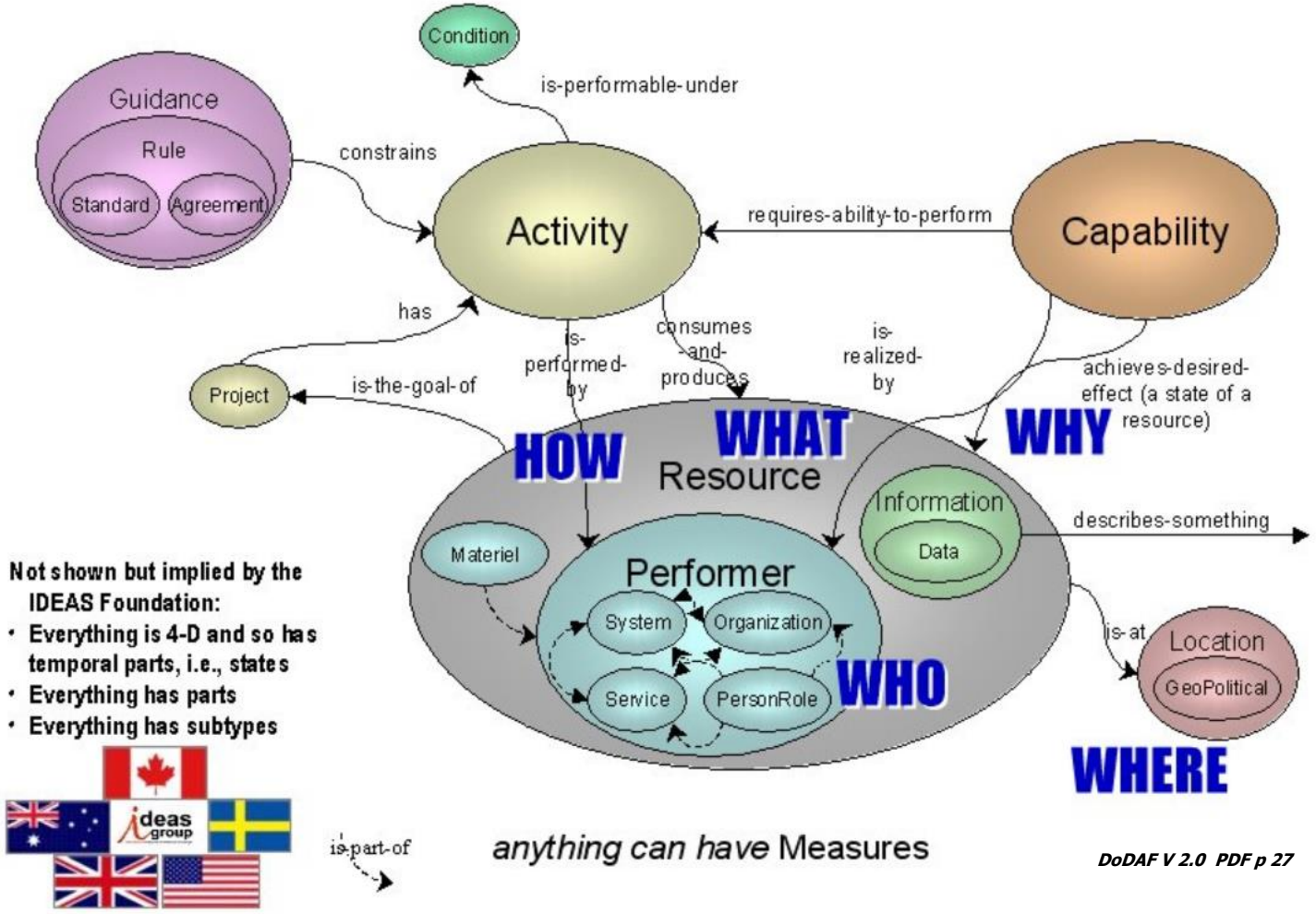
LML Translation

- Two types of mapping for tailoring:
  - Map names of classes to enable other “schema” models to be used
  - Map symbols used (e.g., change from LML Logic to Electrical Engineering symbols)
  - Enable diagram translations (e.g, Action Diagram to IDEF 0)

LML Class	DM2	SysML	…
Action	Activity	Activity	
Asset	Performer	Actor	

LML Symbol	Electrical Engineering	BPMN	…
			

Example: Translation to DM2



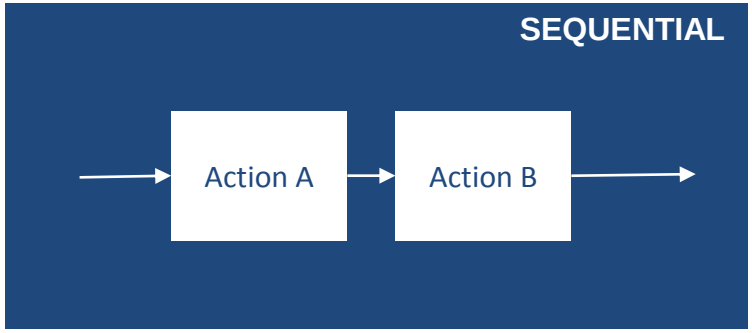
DM2 Conceptual Model to LML Schema Mapping

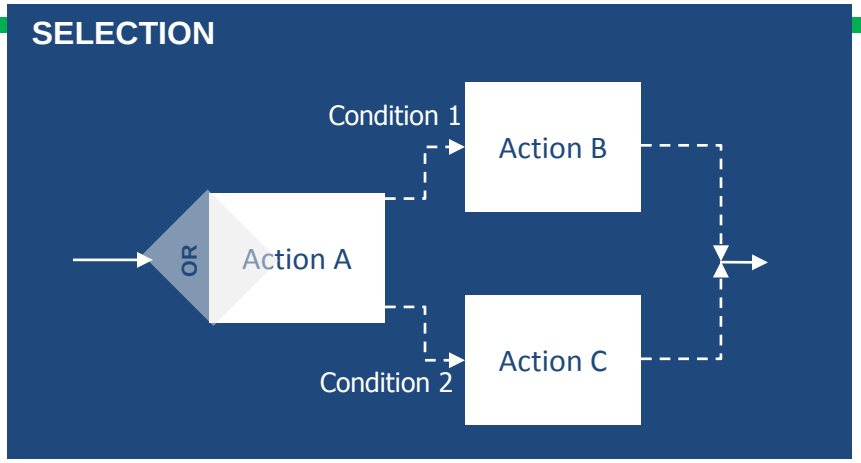
DM2 Schema Element (Conceptual)	LML Equivalent
Activity	Action
Capability	Action with “Capability” type
Condition	Characteristic with “Condition” type
Information/Data	Input/Output
Desired Effect	Statement with “Desired Effect” type
Guidance	Statement with “Guidance” type
Measure	Measure
Measure Type	Measure types
Location	Location
Project	Action with “Project” type
Resource	Asset with types for “Materiel,” “Organization,” etc.
Skill	Characteristic with “Skill” type
Vision	Statement with “Vision” type

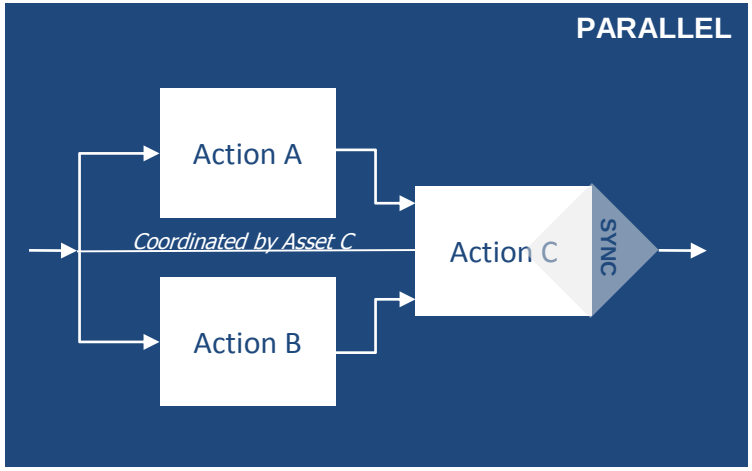
LML Visualizations

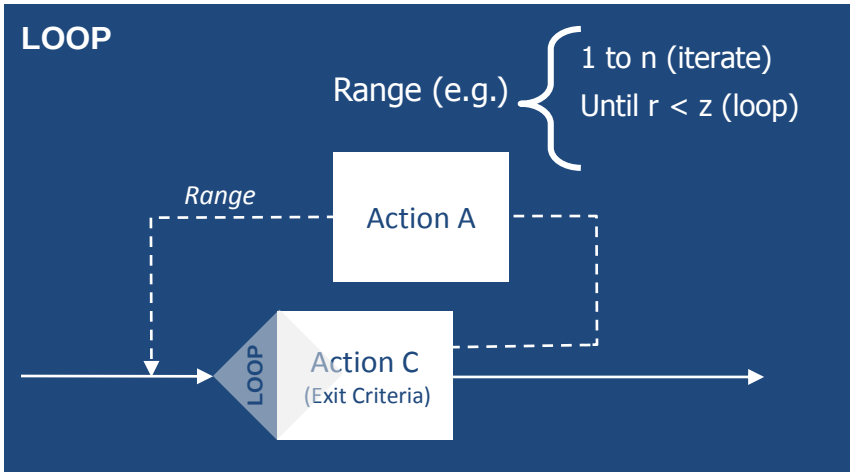
Key diagrams needed to better understand the information

LML Sequencing



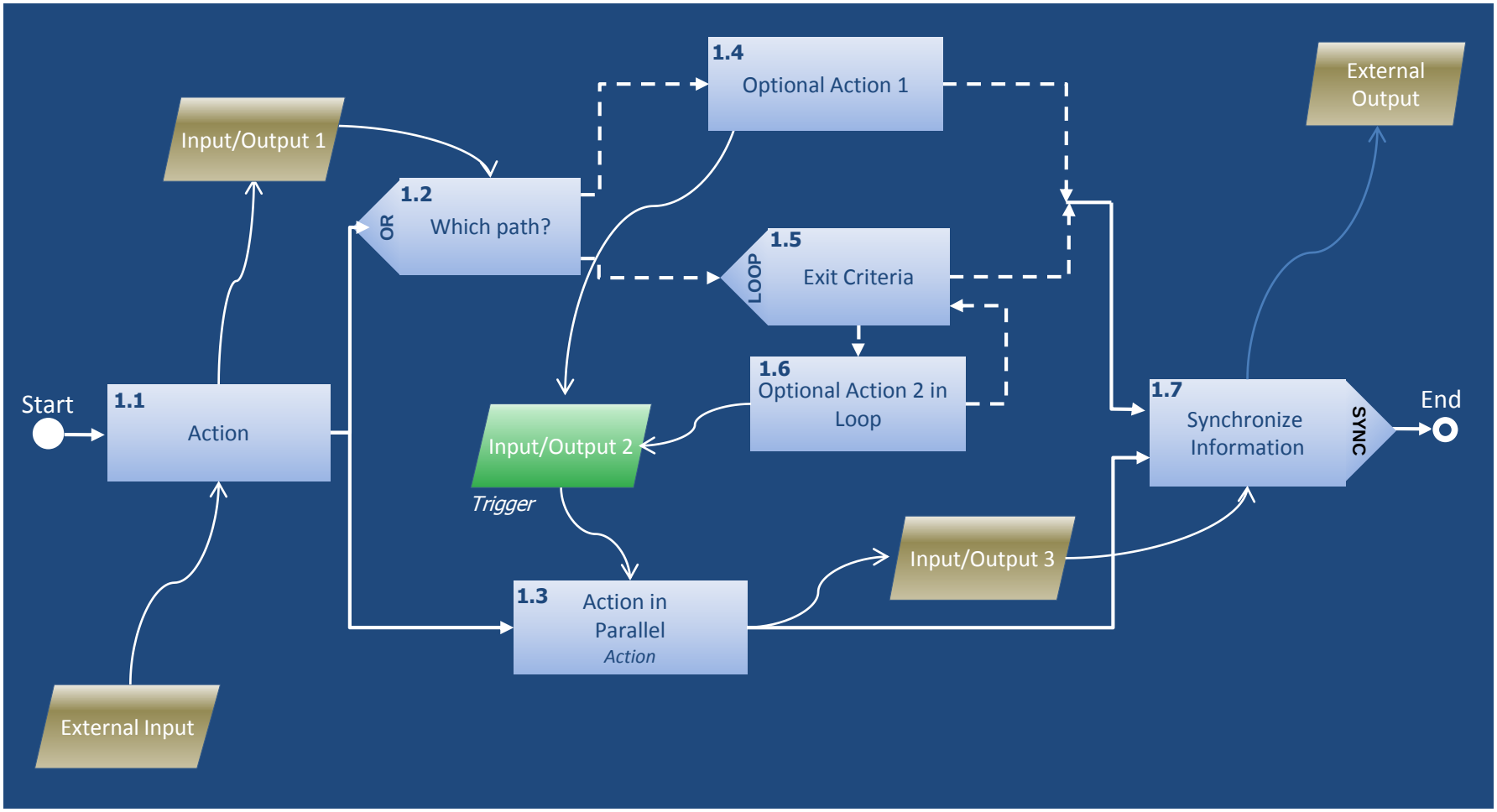






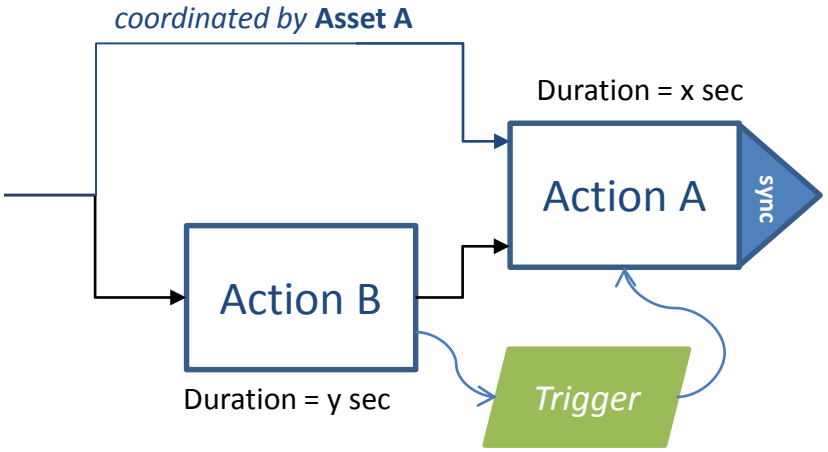
No constructs – only special types of Actions – ones that enable the modeling of command and control/ information assurance to capture the critical decisions in your model

LML Action Diagram Captures Functional and Data Flow

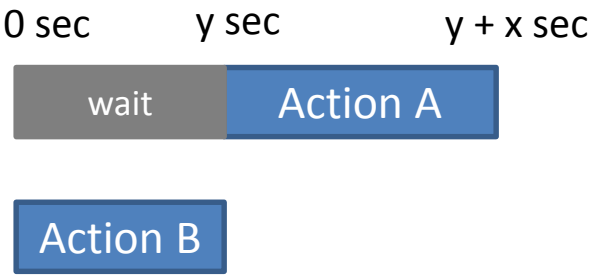


Execution Logic – Concurrency With Trigger; No Coordination Action

Action Diagram



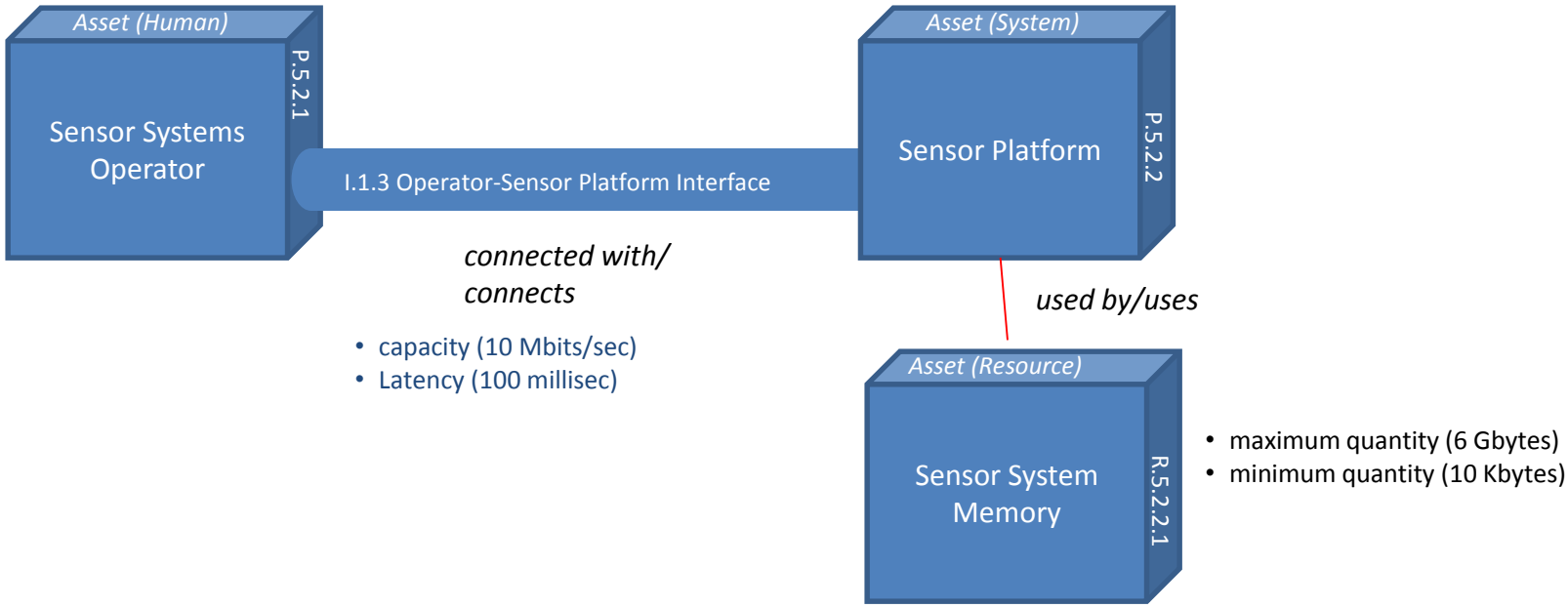
Timeline



Trigger: Action A enabled, but must wait to execute; Asset A performs Action A

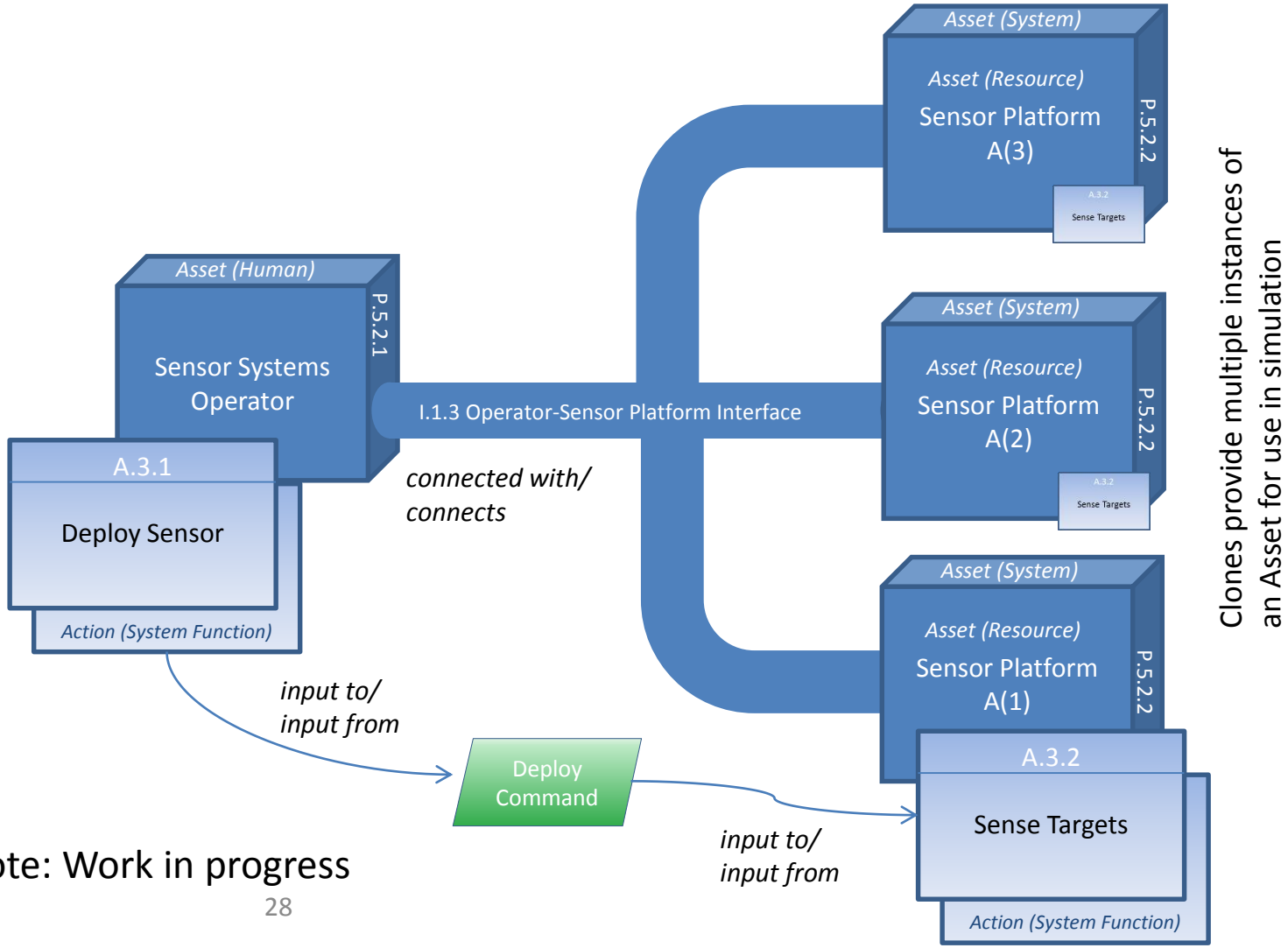
Finish to Start (FS) between B and A

LML Physical Block Diagram*



*Note: Work in progress

LML Combined Physical Behavior Diagram* Enables Instances and Clones



*Note: Work in progress

Diagrams Are Needed for Every Class

- Physical Block (Asset) Diagrams
  - With option for icon substitution
- Interface Diagrams
  - N2 (Assets or Actions)
- Hierarchy Diagrams
  - Automatically color coded by class
- Time Diagrams
  - Gantt Charts
  - Timeline Diagram
- Location Diagrams
  - Maps for Earth
  - Orbital charts
- Class/Block Definition Diagram
  - Data modeling
- Risk Chart
  - Standard risk/opportunity chart
- Organization Charts
  - Showing lines of communication, as well as lines of authority
- Pie/Bar/Line Charts
  - For cost and performance
- Combined Physical and Functional Diagram

LML Summary

- LML provides the fundamental foundation for a tool to support SE in the cloud
- LML contains the basic technical and programmatic classes needed for the lifecycle
- LML defines the Action Diagram to enable better definition of logic as functional requirements
- LML uses Physical Diagram to provide for abstraction, instances, and clones, thus simplifying physical models
- LML provides the “80% solution”
  - It can be extended to meet specific needs (e.g. adding Question and Answer classes for a survey tool that feeds information into the modeling)

Backup Slides

LML Tool Needs

Scalability and collaboration across the lifecycle are essential

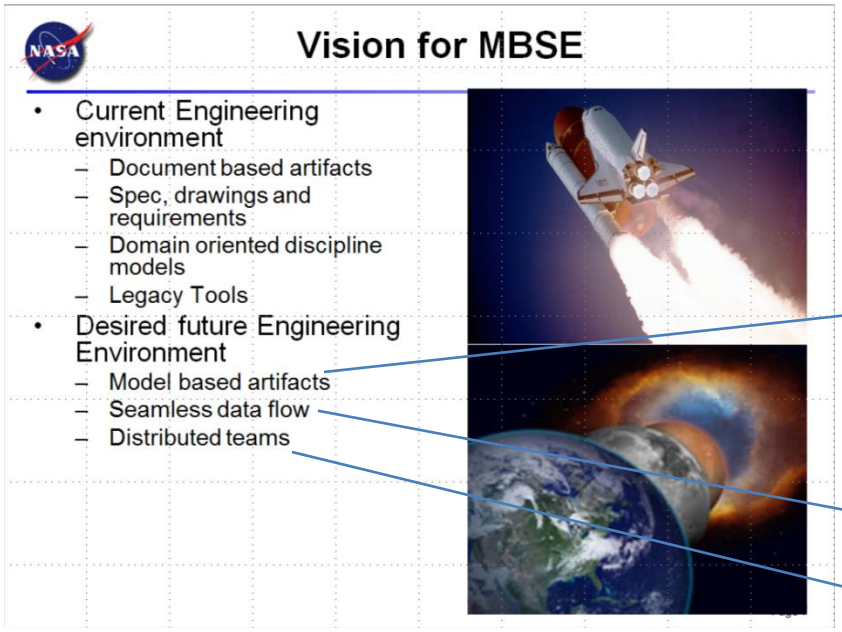
LML Tool Needs

- At a high level, we know that any tool must scale to accommodate a very large data set
  - We are trying to capture information about the systems of systems being developed, their interactions, decisions, risks, cost, and many other parameters throughout the lifecycle
- Also, collaboration is critical as a large data set means many, many people involved, who use different terminology and perhaps even different languages – worldwide

Fortunately we have a new environment for our LML tools – Cloud Computing

NASA “Requirements”

Tool for Vision for MBSE



From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool must be designed as a MBSE tool using Lifecycle Modeling Language (LML)
  - All artifacts must be produced from the tool using standard reports
- Tool must enable both seamless data flow and distributive, collaborative teams working on the same model
- Implies need for ability to scale “infinitely” on demand

Tool for Complexity

Complexity is a Major Issue

- Integration of systems create a major problem with complexity
  - Within in a system, interactions grow as N squared or worse
  - Ability to understand and test becomes less certain
  - As more systems are added, the interfaces grow in a non-linear fashion
  - Many of the existing systems are old and not built for these interfaces
  - Conflicting or missing interface standards make it hard to define interface interactions
  - Hardware and software may be re-purposed and "heritage" compromised
  - Future systems will be integrated from multi-organizational, multi-national contributions, adding additional layers of complexity.
- Systems engineering must deal with this complexity
  - End-to-end systems engineering is needed, including "reengineering" of old systems
  - Robust M&S, verification and validation testing are A must

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

Page 8

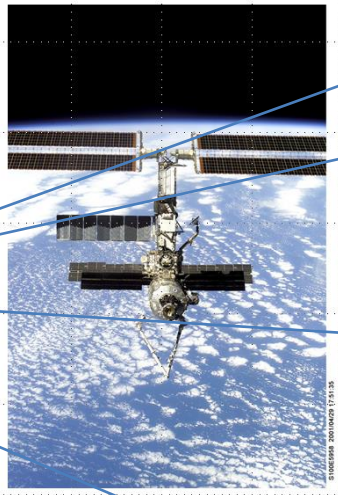
- Tool must also reduce complexity by use of special input screens, which enhance configuration management as well

- Tool must be designed for use throughout the lifecycle, thus enabling end-to-end systems engineering
  - Legacy systems are a key part of any modeling environment – LML has specific information classes for capturing designs at different times in one knowledgebase
- Tool must embed simulation, which enables real time simulations as well as the ability to scale "infinitely" to hundreds of server instances

Tool for Multi-Decadal/Generation

Supporting multi-decadal, multi – generation activities is a major challenge

- The International Space Station will have a lifespan of at least 20 years with evolving uses and constant changes
- Systems analyses show that as we explore beyond low earth orbit, launch costs will remain a driver and thus put a huge value in re-using systems already moved up the gravity well
  - We will need to track systems health and status against predictions and threshold
  - Systems will be modified, updated and re-purposed multiple times
  - Operating environments and conditions may change from those used for design
  - Likely to want to use systems well beyond initial life objectives



Page 9

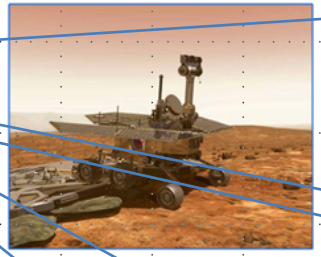
From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool must enable tracking of performance over time as Characteristics of the Assets
- Environments and conditions can also be captured as evolved over time and location (including orbital locations)
- Tool must evolve of the lifecycle of systems, but the data must be captured and reused throughout the lifecycle

Tool for Uses and Challenges

Use and Challenges vary throughout the life cycle

- Early Phases; conduct Systems Analysis and trade studies
  - Exercise through CONOPS
  - Rapidly dismiss faulty options
  - Develop feasible, cost effective options
  - Identify advanced technology requirements
  - How to enable modeling that provides the needed fidelity yet can be done quickly and cheaply?
  - Current methods tend to be "wetware" intense.
  - Need rapid and effective teaming
- Development
  - Refine Design, support Validation & Verification
  - Enhance manufacturing
  - How to better enable integration of discipline oriented design tools into systems models that capture functional and performance behaviors
  - How to capture system design rationale, assumptions and other "background" data.
  - How do we develop the standards that allow lossless integration across organization and tool boundaries?
- Operations
  - Provide data for ops team, resolve in flight issues, address parts obsolescence
  - How do we make the full suite of information captured during design and development available to the operators without having prior knowledge of their needs?



Page 12

From a presentation by Dr. Michael Ryschkewitsch,
NASA Chief Engineer, at CSER Conference 15 April 2011

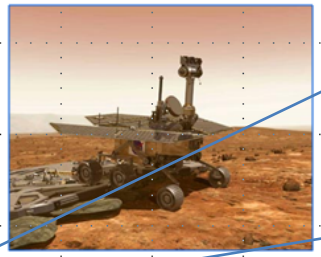
- Tool should automate what can be automated to reduce "wetware" requirements and enhance teaming

- Tool needs to create a CONOPS report based on scenario modeling
- Capture Issues and Decisions, as well as Cost for cost, schedule and performance tradeoffs
- TRL levels must be easily captured in the tool
- Software as a service provides inexpensive tool, while scalability enables fidelity level required; LML method enables rapid SE design and analysis

Tool for Uses and Challenges
(continued)

Use and Challenges vary throughout the life cycle

- Early Phases; conduct Systems Analysis and trade studies
  - Exercise through CONOPS
  - Rapidly dismiss faulty options
  - Develop feasible, cost effective options
  - Identify advanced technology requirements
  - How to enable modeling that provides the needed fidelity yet can be done quickly and cheaply?
  - Current methods tend to be "wetware" intense.
  - Need rapid and effective teaming



- Development
  - Refine Design, support Validation & Verification
  - Enhance manufacturing
  - How to better enable integration of discipline oriented design tools into systems models that capture functional and performance behaviors
  - How to capture system design rationale, assumptions and other "background" data.
  - How do we develop the standards that allow lossless integration across organization and tool boundaries?
- Operations
  - Provide data for ops team, resolve in flight issues, address parts obsolescence
  - How do we make the full suite of information captured during design and development available to the operators without having prior knowledge of their needs?

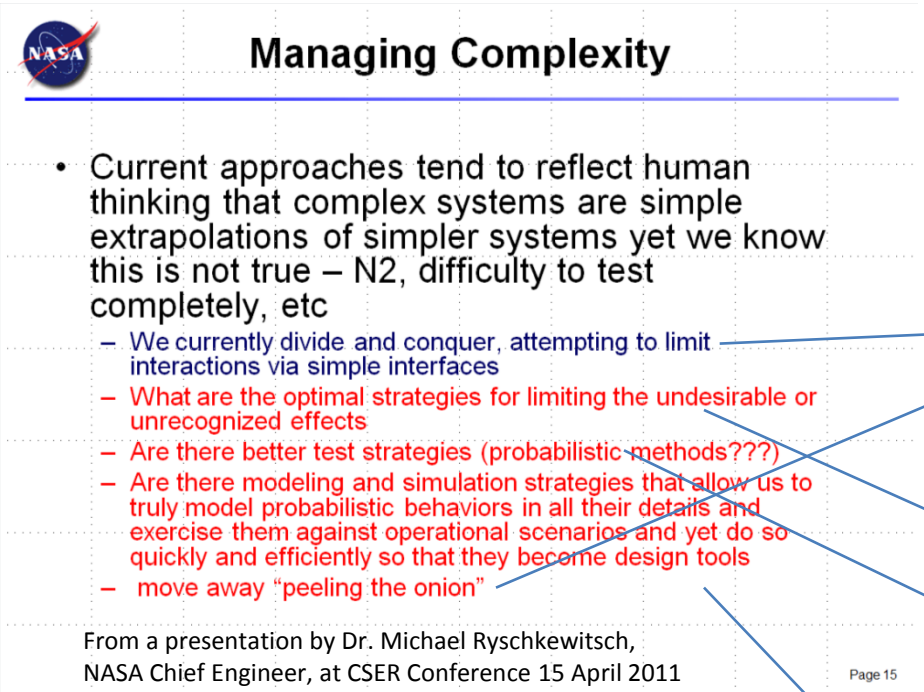
Page 12

From a presentation by Dr. Michael Ryschkewitsch,
NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool should enable design down to detailed level (if desired), as well as V&V support
- Manufacturing enhanced by linking design to CAD/CAM systems (export design information as required)
- Use of XML files to import/export data from other design tools
- Design rationale, assumption and other programmatic information captured as part of Issues, Risk and other program-related elements of LML
- Standards should be captured and developed using the tool; enforcement by tailoring "personal trainer" version

- Operations data and obsolescence part of LML elements (Assets, Characteristics and Time)
- Use of a scenario-based approach enables better information capture from operators

Tool for Managing Complexity



Managing Complexity

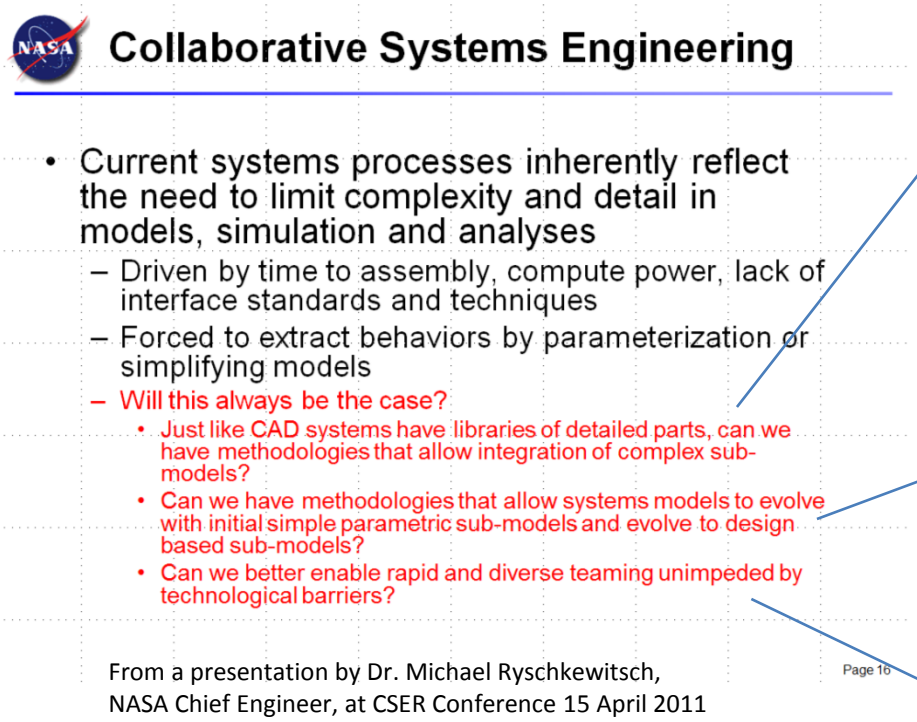
- Current approaches tend to reflect human thinking that complex systems are simple extrapolations of simpler systems yet we know this is not true – N2, difficulty to test completely, etc
  - We currently divide and conquer, attempting to limit interactions via simple interfaces
  - What are the optimal strategies for limiting the undesirable or unrecognized effects
  - Are there better test strategies (probabilistic methods???)
  - Are there modeling and simulation strategies that allow us to truly model probabilistic behaviors in all their details and exercise them against operational scenarios and yet do so quickly and efficiently so that they become design tools
  - move away "peeling the onion"

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

Page 15

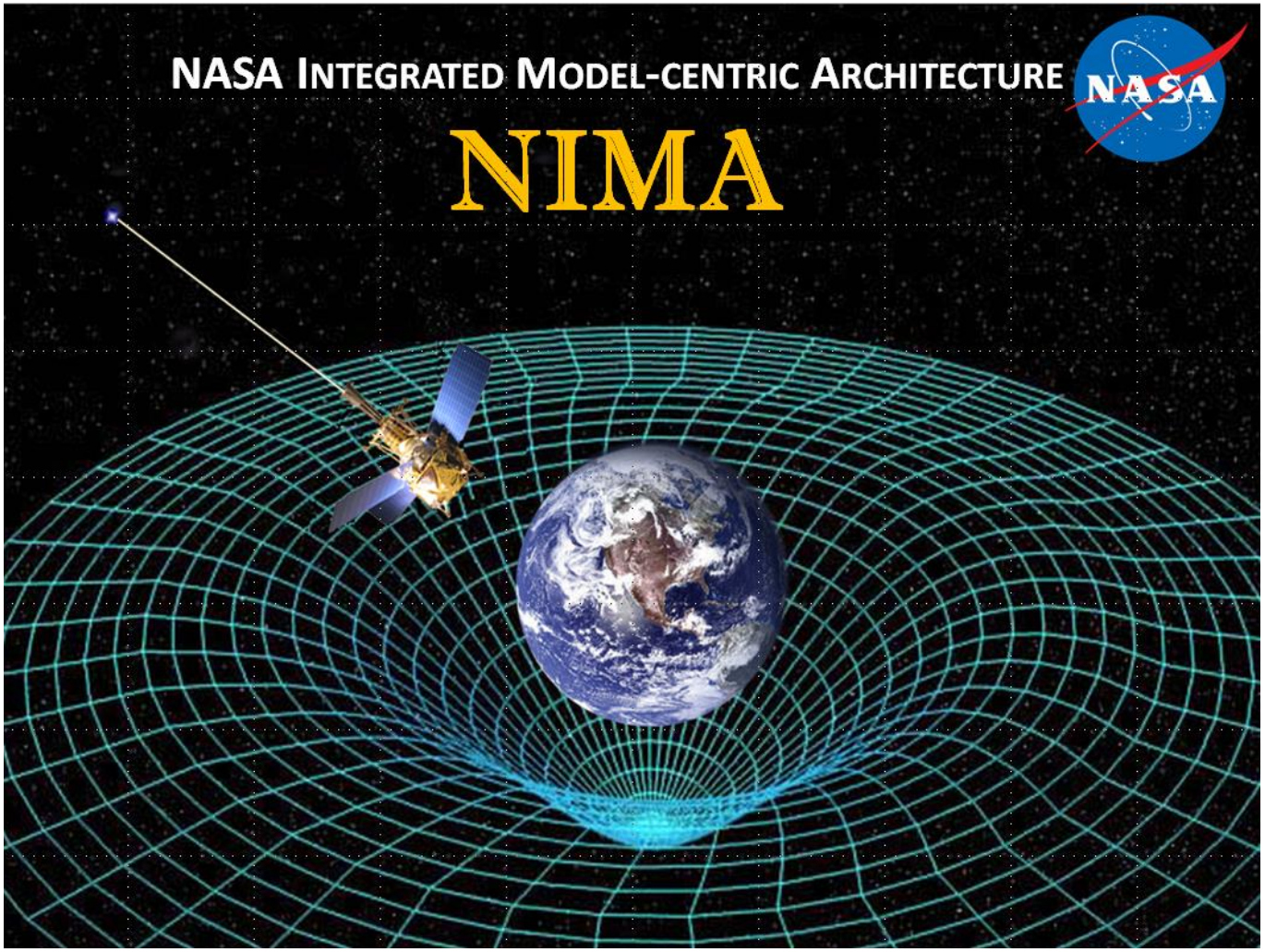
- Tool in conjunction with a process should capture a reasonable number of scenarios for modeling and simulation, using a test matrix approach (move away from "peeling the onion")
- Use "test matrix" approach to provide breadth of problem space, including failure modes, to capture the full functionality required
- Test must become even more dependent on simulation

Tool for Managing Complexity



- Tool should enable sharing of models, thus creating a “library” of component models
- Contributions should be made from academia (tool should be free access to academia) to this library
- Government sponsored research can also be made available to the NASA family via website

Response to Chief Engineer



From a presentation by Mr. Joe Smith, NASA HQ Program Executive for Systems Engineering, at INCOSE WMA Meeting, 17 September 2014

NIMA Goals

Working within the constrains of our budget and charter, we will generate products that move the agency towards achieving the following goals (presented to EMB 10/13/11):

Goal 1: Increase ***affordability*** through use of a model-centric architecture

Goal 2: Achieve ***interoperability*** within and among programs/projects, centers and external partners through use of a model-centric architecture

Goal 3: Inform/train invigorate ***workforce*** on model-centric architecture

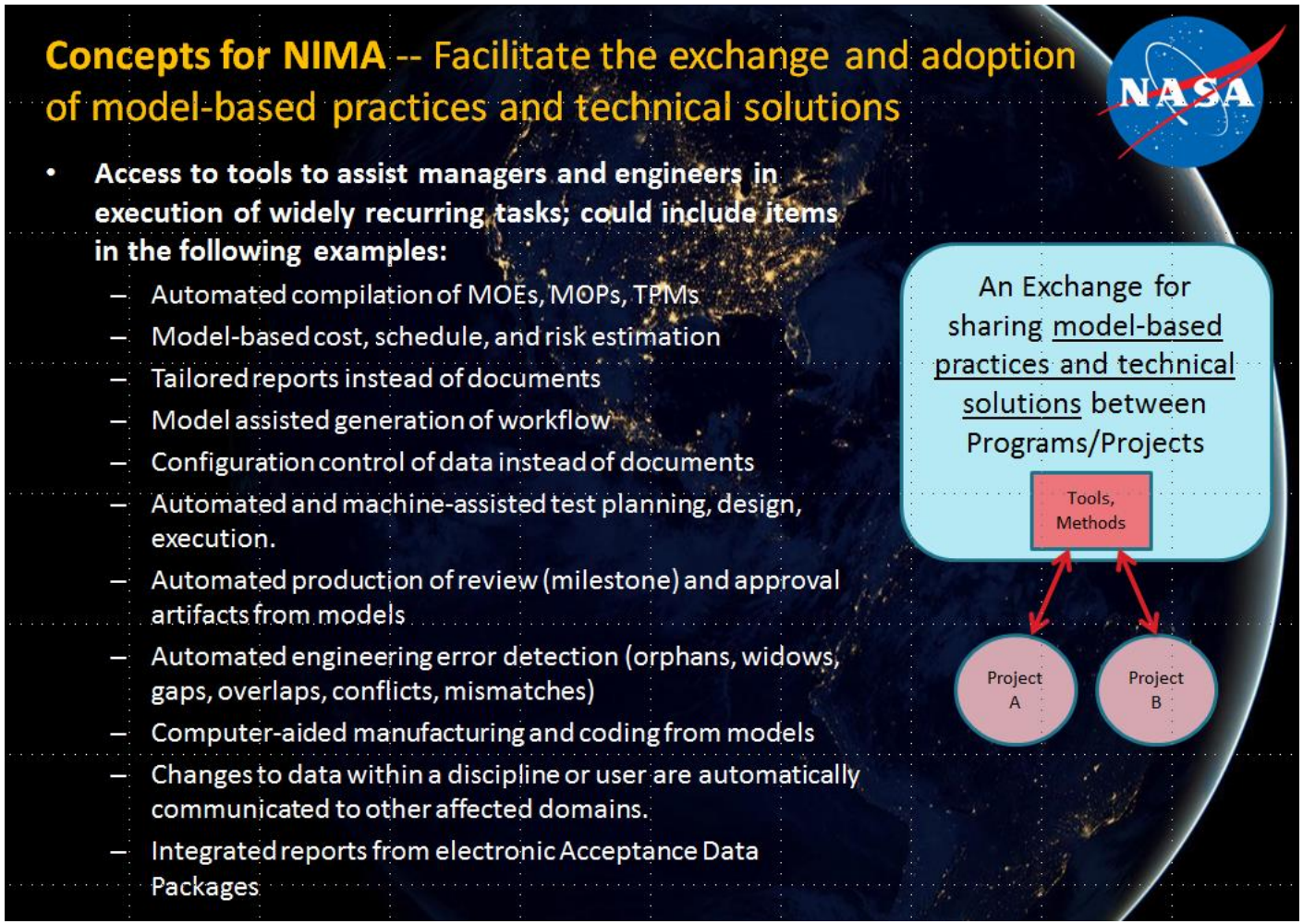
Goal 4: Improve product ***quality*** and success through use of a model-centric architecture

Form of Products:

- Refined standards, requirements, and guidance for model-centric data exchange & management, and model-centric methodologies
- Requirements for enabling products (training, IT infrastructure)
- "App Store" to facilitate exchange of model-centric practices and technical solutions

From a presentation by Mr. Joe Smith, NASA HQ Program Executive for Systems Engineering, at INCOSE WMA Meeting, 17 September 2014

NASA "App Store"

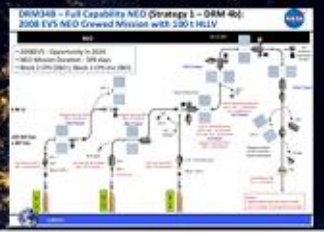


Another DoDAF MetaModel 2.0 (DM2) Physical Exchange Specification (PES)?

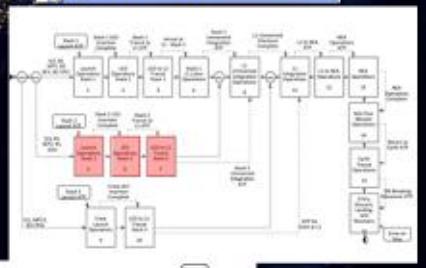
From a presentation by Ms Linda Bromley, Manager, Technical Integration Office at NASA Johnson Space Center to AIAA

Use Case -- Determining an Effect of a Requirement Change

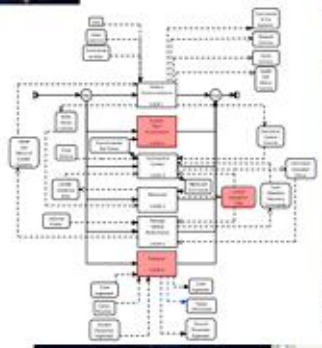
- **Premise:** Late in a development cycle a key requirement is changed.
- A fully model-centric program will be able to trace the effect of that change through many viewpoints



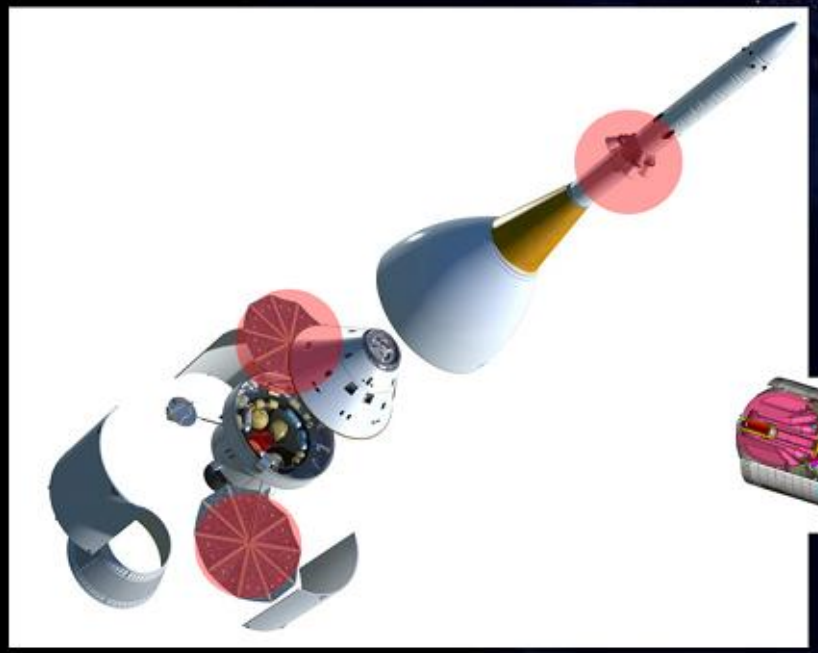
DRM Affected

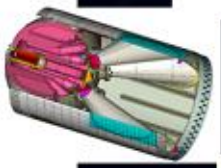


Mission Segment Affected



Functions Affected





Parts Affected

16

From a presentation by Ms Linda Bromley, Manager, Technical Integration Office at NASA Johnson Space Center to AIAA

Organized Functional Tool Requirements

(derived from analysis of Mike R's presentation)

- Models (MBSE)
  - Functional
  - Physical
  - Object
- Simulation
  - Discrete event
  - Monte Carlo
- Data Sharing
- Collaboration
  - Worldwide access to single database
  - User interaction (e.g., Chat)
- Scalable to large datasets
- Decision/design traceability
- Cost and Schedule analysis support
- Risk analysis support
- Explicit time evolution analysis
- Full lifecycle support
  - Requirements
  - Design
  - Acquisition
  - Verification
  - Operation & Support
  - Disposal
- Reports from Models
  - CONOPS
  - DoD or other documents
  - Project-specific
- "User Friendly" & Standards
  - Modern (web-like) UI
  - Desktop Support (PC, Mac, Linux)
  - Tablet Support (iPad, Android)
  - Enforces standards via rules and checkers
  - Embedded training
- "Reasonable" tool cost

Innoslate® is the first tool to implement LML completely

IMPLEMENTATION OF LML

Implementation of LML

- Prior to development of LML, a schema was developed called KBAD and implemented in Vitech's CORE tool
- We used this for a number of years to get many of the rough edges out of the schema
- As such, LML's ontology can easily be implemented in a number of tools
- The diagrams, particularly the Action Diagram, must be implemented by tool vendors
- Innoslate® is the first of these LML tools

What is Innoslate?

- A tool to capture requirements, design, operations, and support information using systems engineering principles
- Enables requirements analysis and management with direct linkages to design models
- Applies cloud computing technology to model-based systems engineering techniques and discrete event simulation
- Available on the public cloud, private clouds, and client-server platforms
- Implements the lifecycle modeling language (LML) and enables translation to UML, SysML, DoDAF (DM2), and other languages

Innoslate® provides software as a service (SaaS)

What are the system requirements?

- Operating system
  - Any (Windows XP/7, MAC OS X, Linux)
- Devices
  - Any (PC, Mac, iPad, Android Tablet)
- Software
  - Any modern web browser (Google Chrome, Firefox 5+, Safari, limited support for IE 9+ or IE 7+ with Google Chrome Frames)
- No downloads required

What Can Innoslate® Produce?

- Tool Capabilities
  - Early Simulation/Design Validation
  - End-to-End Design Management and Collaboration
  - Repository of Analyses from Any Tool
  - Traceability from Requirements to Functions to Components to Test to Operations to Support to Disposal
- Version 2.4 Reports
  - Entity Reports for each class
  - Requirements Report
  - Concepts of Operations (CONOPS)*
  - JCIDS Documents (ICD, CDD, CPD, DCR)*
  - DoDAF and MODAF products
  - Test and Evaluation Plan/Report
- Reports slated for later versions
  - Specifications for detailed design (in the languages of the design engineers)
  - Processes and Procedures for Operations and Training

**These complex reports have special input wizards and user guides*

Innoslate Supports LML's Simplified Schema

- Action
- Artifact
- Asset
  - Resource
- Characteristic
  - Measure
- Connector
- Cost
- Decision

Conduct logical decomposition and analysis

Conduct physical decomposition and analysis

Capture verification and validation data

Capture key stakeholder decisions

- Input/Output
- Location
  - Physical, Orbital, Virtual
- Risk
- Statement
  - Requirement
- Time

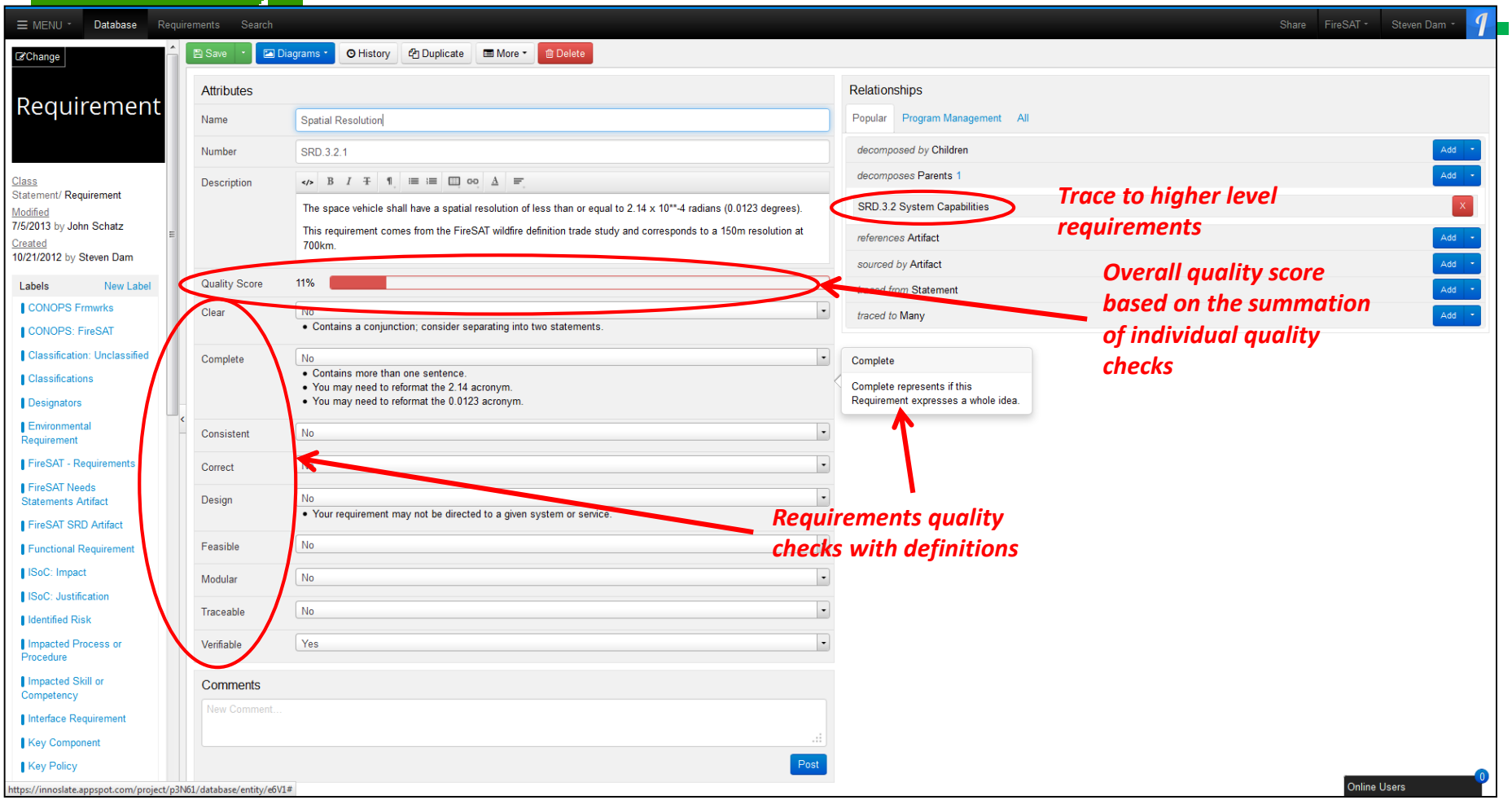
Simplified schema reduces start-up/training time

Designed with space in mind

Capture requirements with quality measures

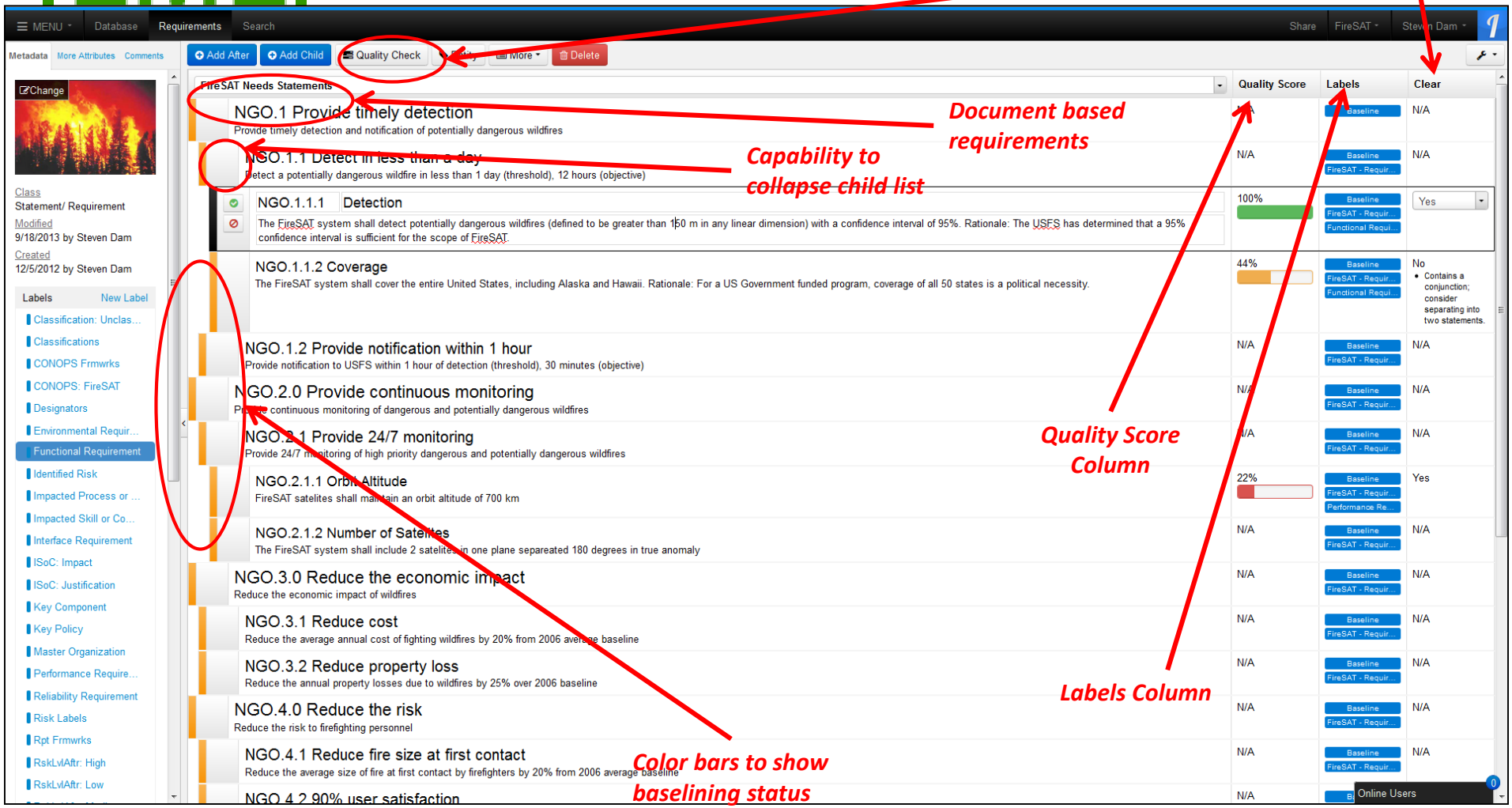
Integrated data analysis of cost, schedule, and performance for the lifecycle

Requirements



True requirements analysis capability, not just management

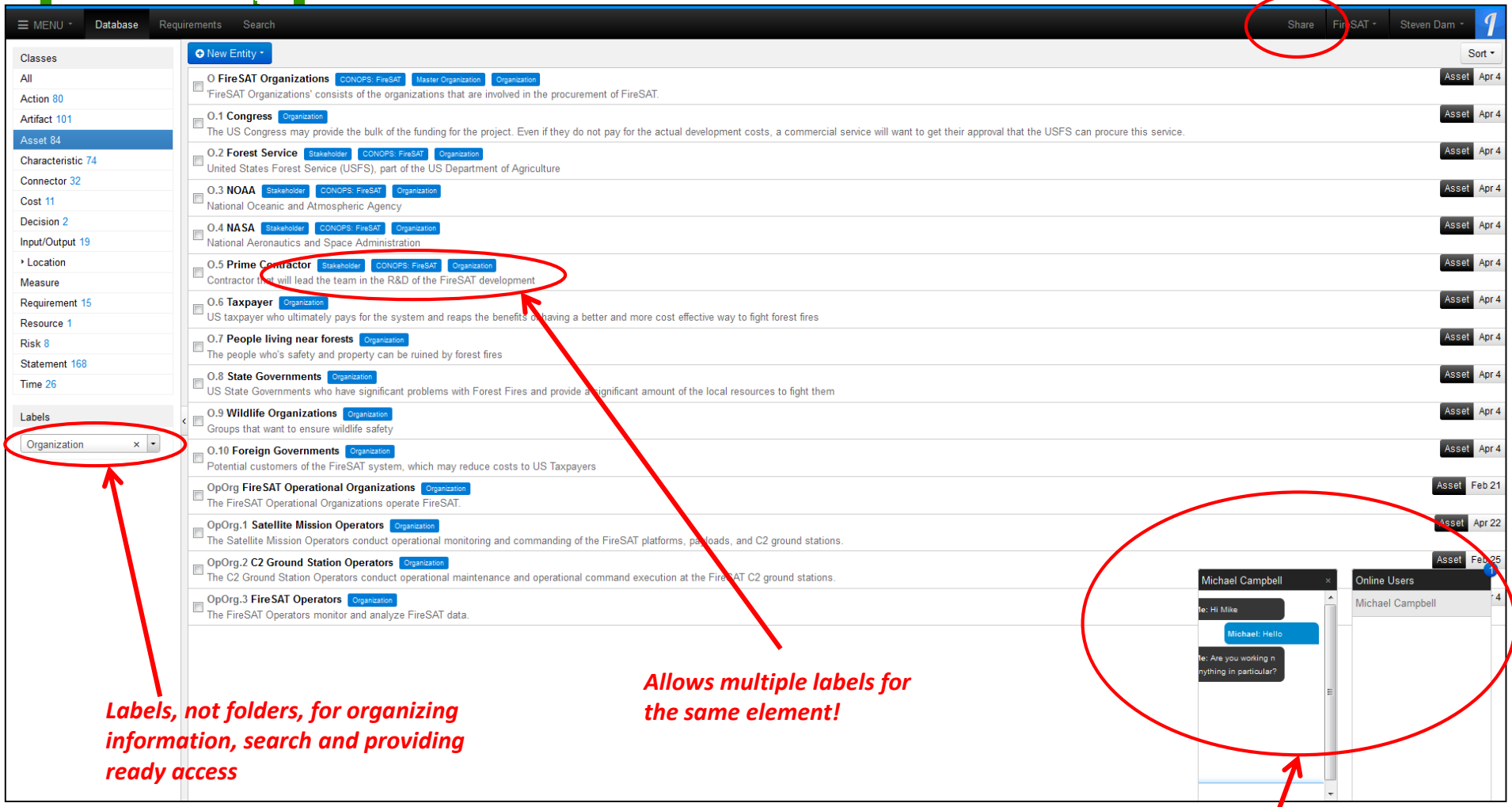
Requirements View



Enhanced requirements management capability, too

Database View

Share database worldwide with colleagues (Read/Write) and reviewers (Read Only with Free version)



Labels, not folders, for organizing information, search and providing ready access

Allows multiple labels for the same element!

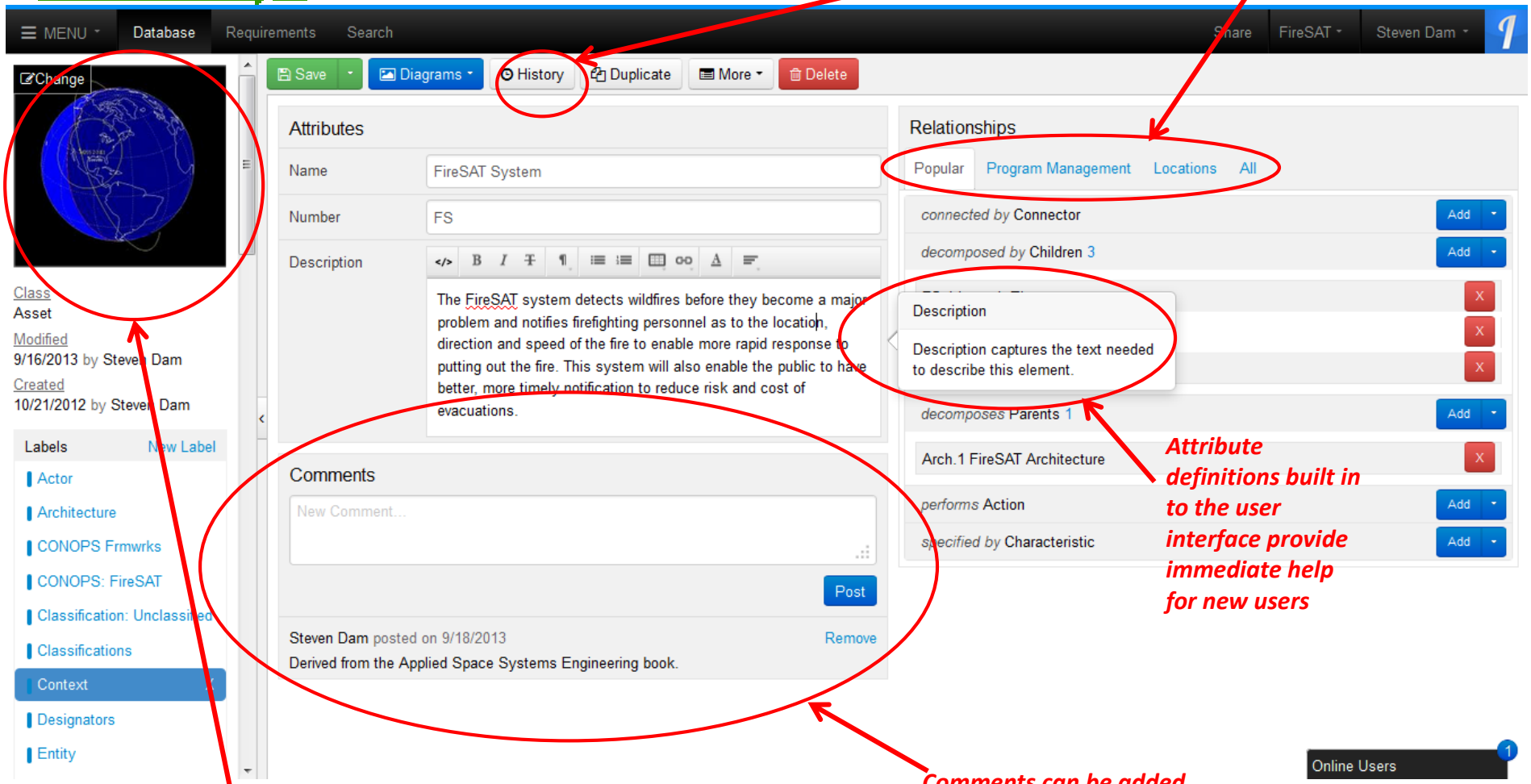
Enable user interaction through built-in Chat

Designed to scale to large data sets

Entity View

Detailed history of changes available for each element

Tabs for logical grouping of relationships reduces information overload

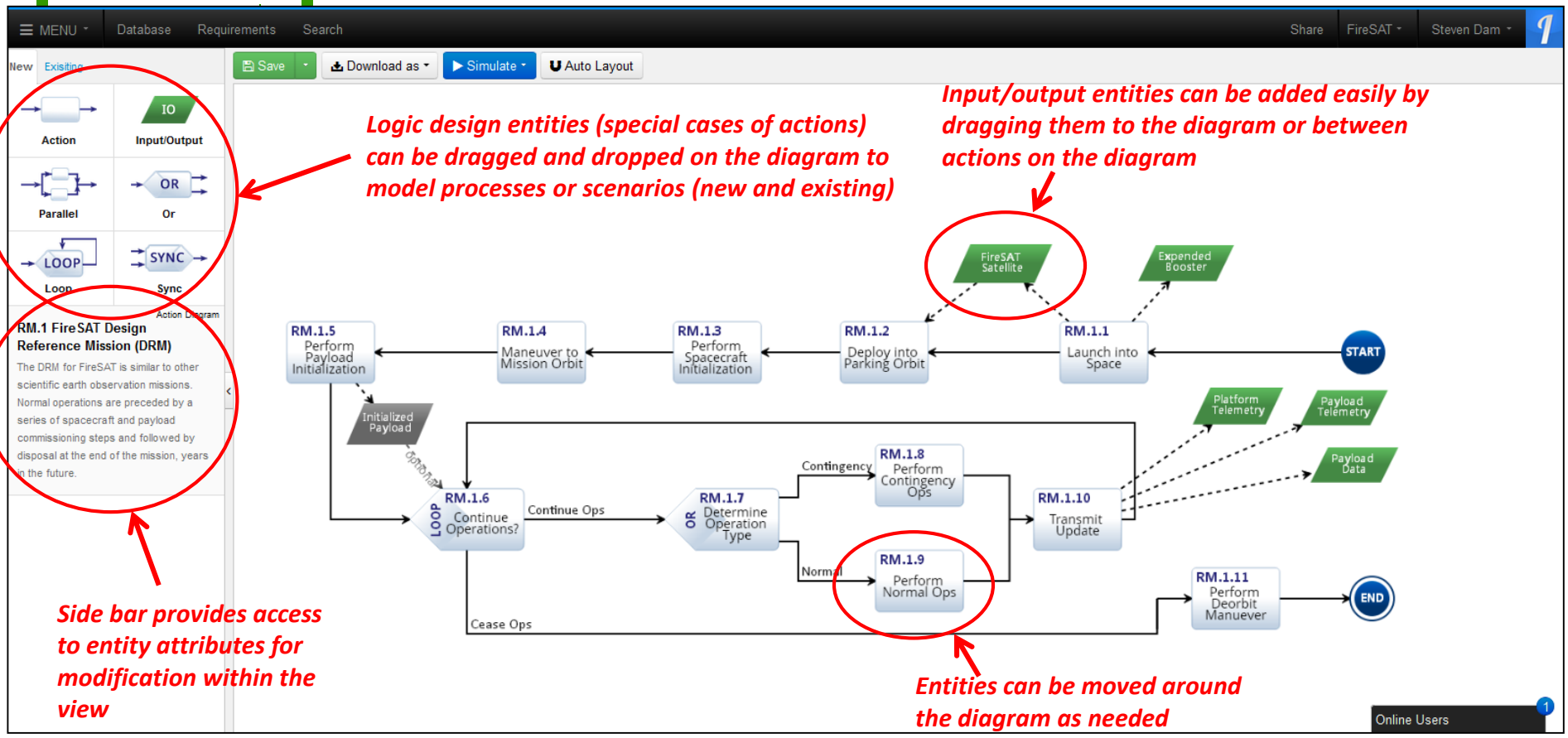


Attribute definitions built in to the user interface provide immediate help for new users

Every element can have it's own picture, which can be used in the reports

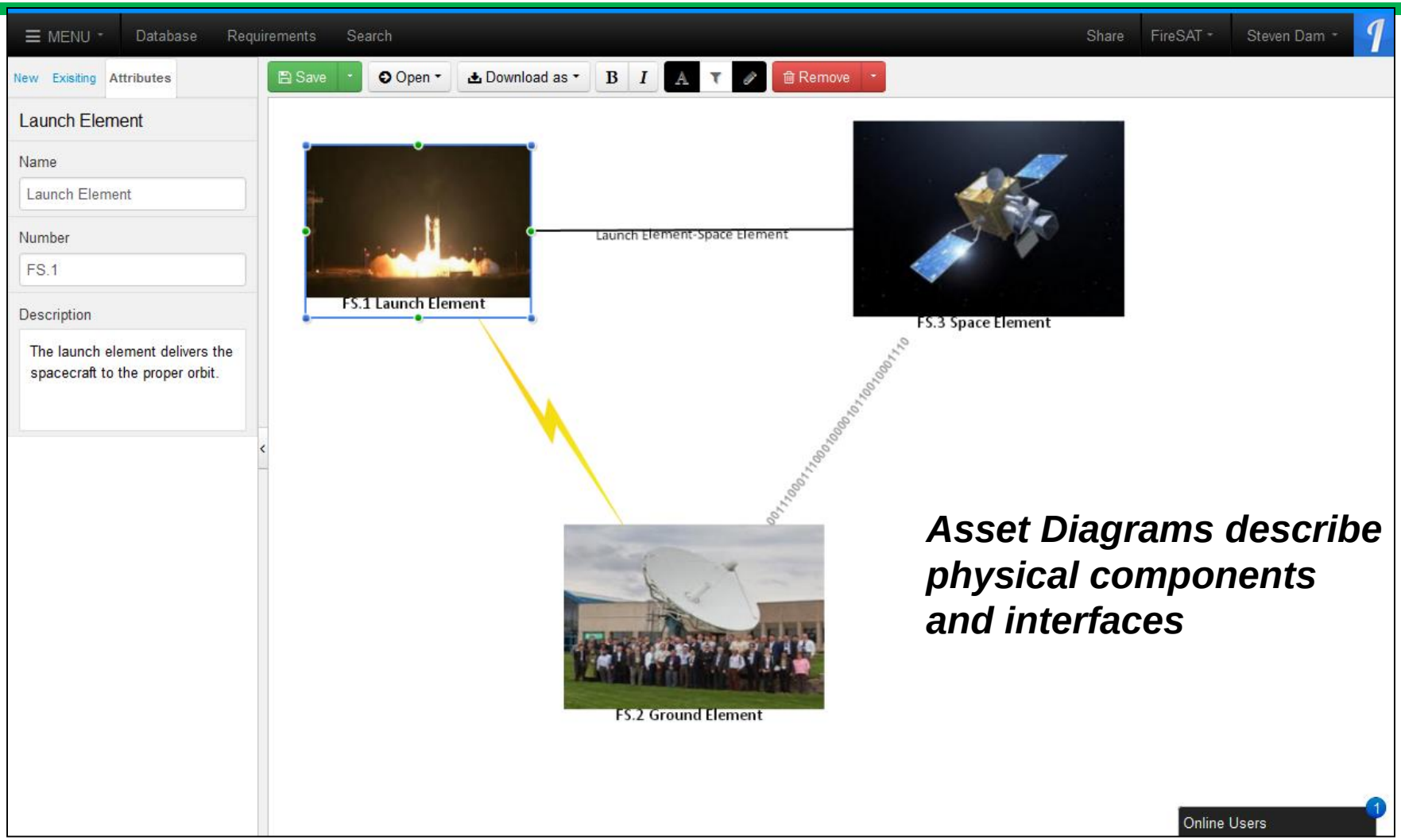
Comments can be added by any user sharing the database, including free users enabling model-based reviews

Action Modeling (Functional View)



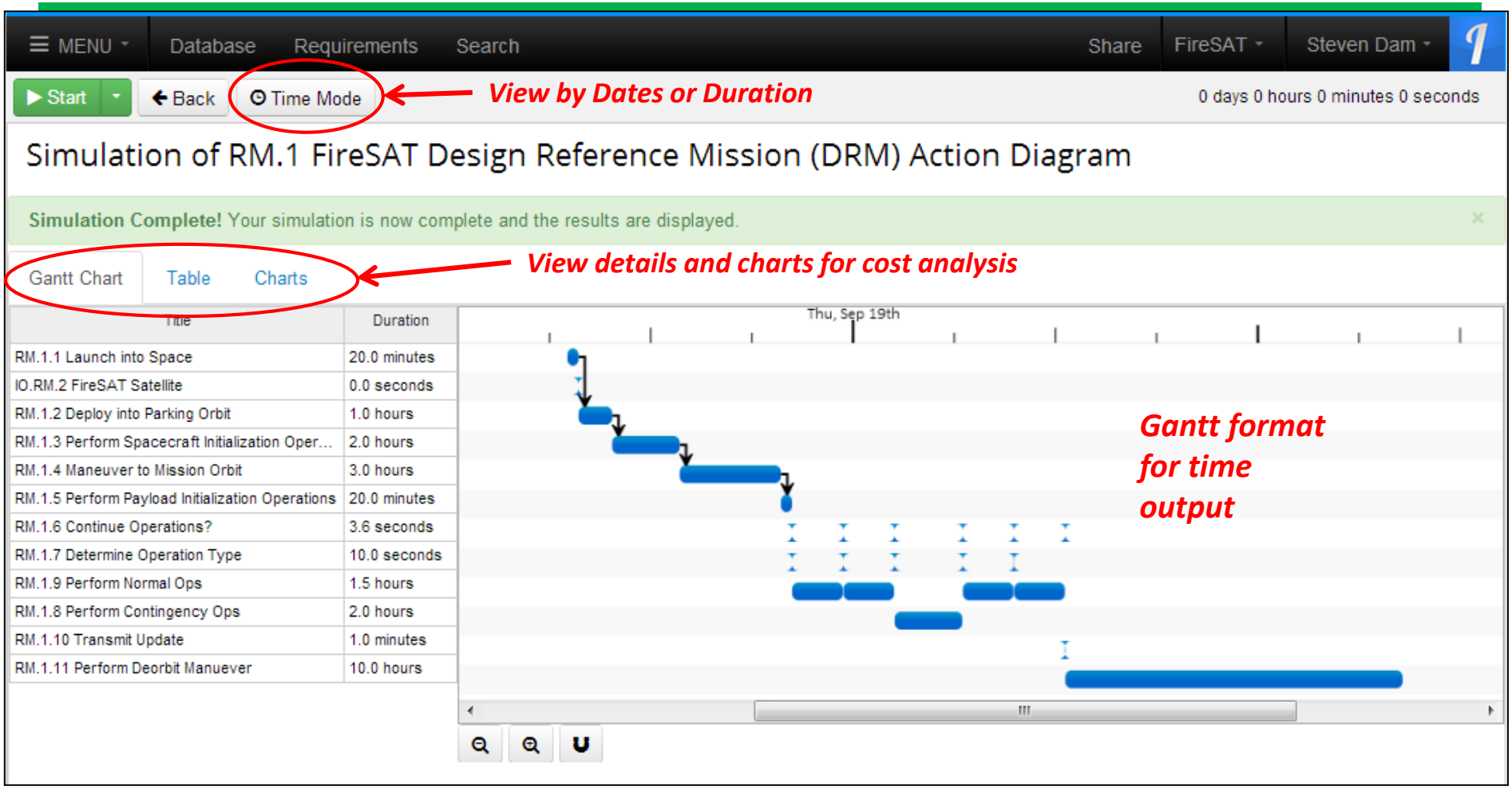
Action Diagrams for functional modeling designed to work on tablets, such as the iPad

Asset Modeling (Physical View)



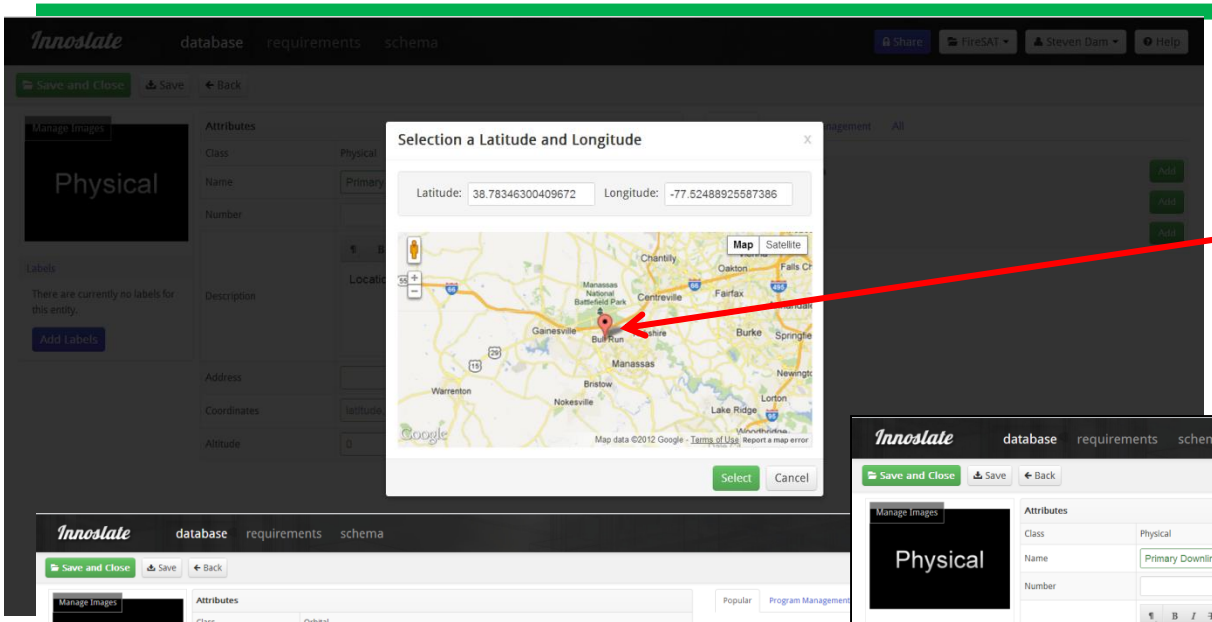
Asset Diagrams describe physical components and interfaces

Discrete Event Simulation

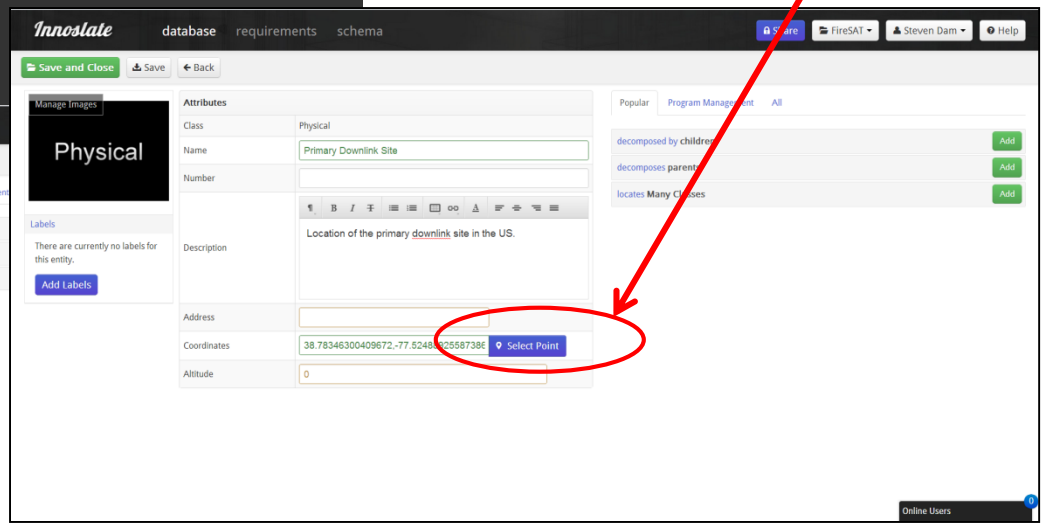


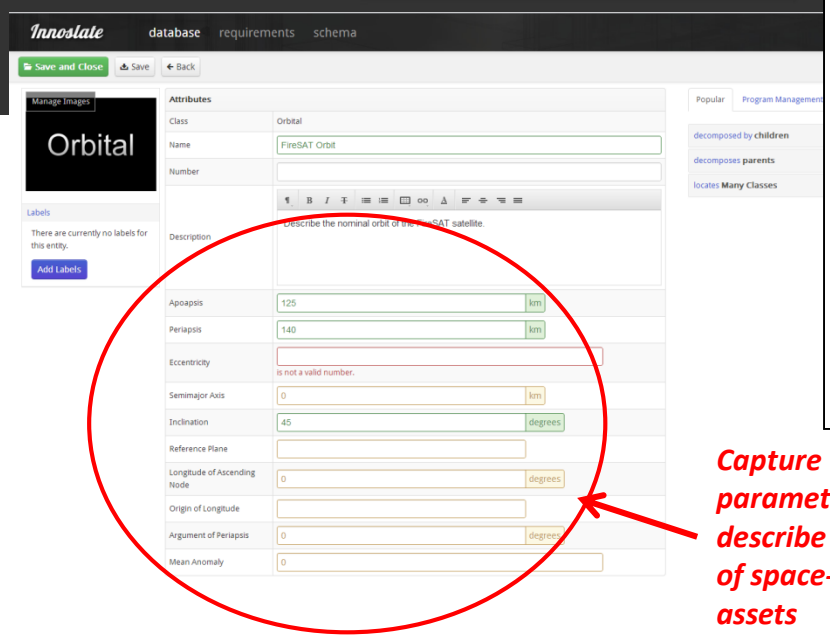
Results saved automatically as an Artifact; Monte Carlo available in Professional Edition

Location



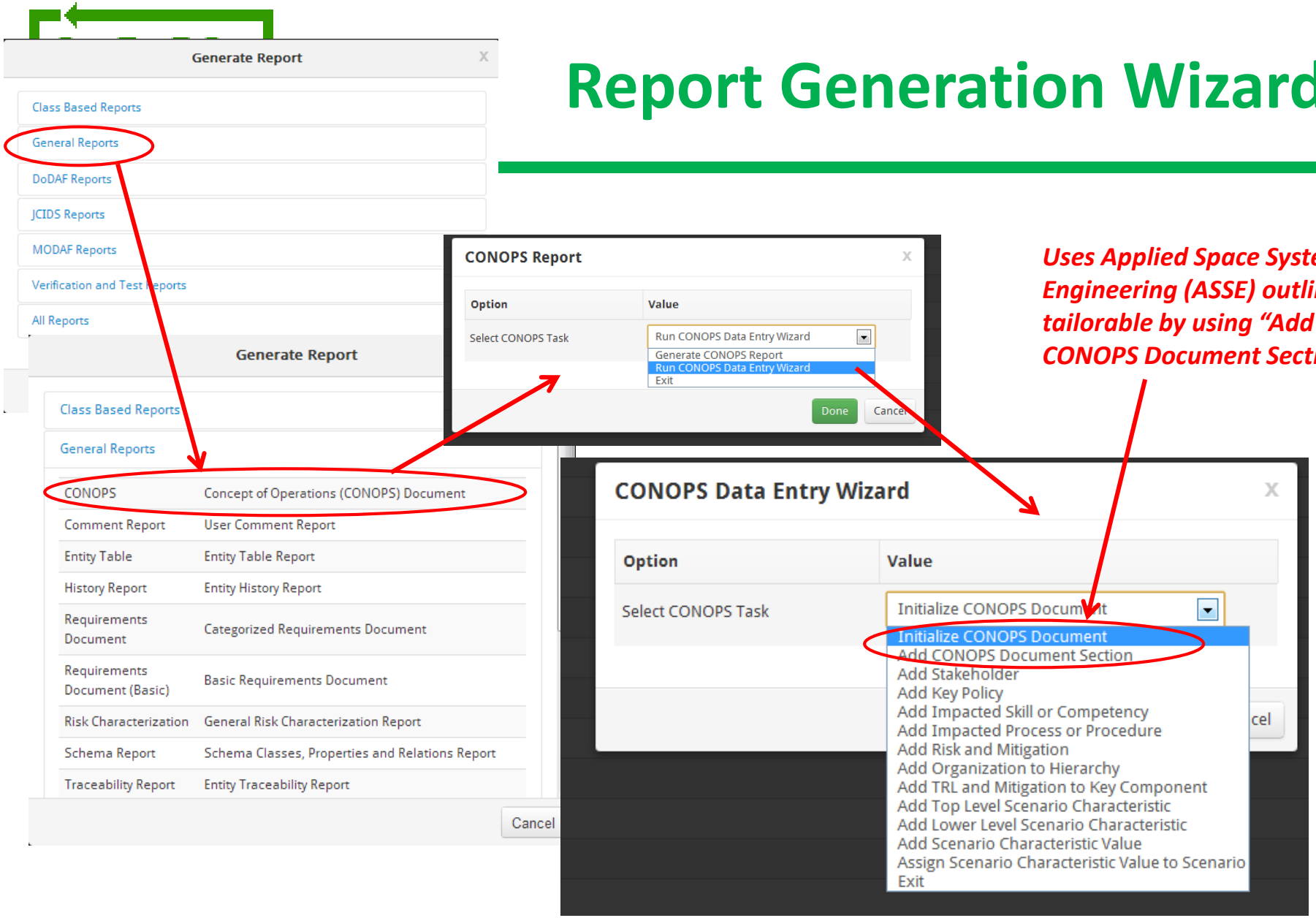
Use Select Point button to provide the longitude and latitude using Google Maps on a Physical Location





Capture parameters that describe the orbit of space-based assets

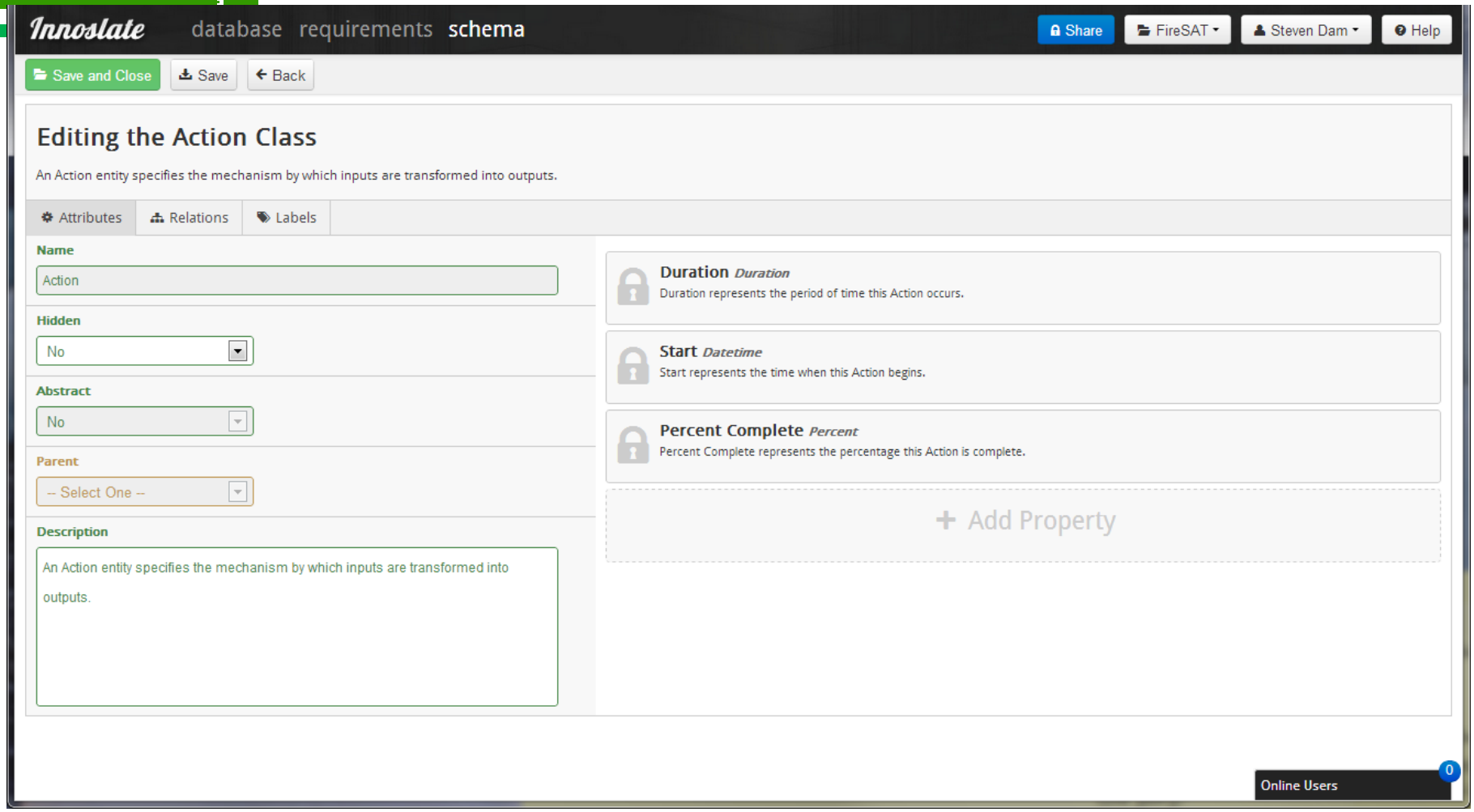
Report Generation Wizards



Uses Applied Space Systems Engineering (ASSE) outline; tailorable by using "Add CONOPS Document Section"

Wizards walk users through documents guiding their inputs, also reducing need for training

Schema Extension



Add new classes, relationships and attributes as you need them

Visual Representations

- Version 2.4 Diagrams
  - Hierarchy charts
  - Action Diagram
  - Asset Diagram (Interfaces)
  - N2 Chart (I/O + Actions)
  - Risk charts
  - Radar Diagram
  - Sequence (Experimental) [Object Modeling]
  - Spider Diagram (All, Custom, Traceability)
  - Timelines (Experimental)
  - Location (maps)
  - IDEF0/ICOM
- Planned for future releases
  - Other location views, including orbital
  - Class, and other UML/SysML diagrams [Object Modeling]

For Risk Analysis Support

For Decision/Design Traceability

For additional time evolution analysis and presentation

Innoslate Features and Benefits

Features	Benefits
Chat/Real-time Notification	Enables collaboration worldwide
Lifecycle Modeling Language (LML) schema and Web User Interface	Provides simplicity and ease of use with little or no training
Open feature requests with voting on priority by users (Feature Tracker)	Supports transparency between users and developers
Google App Engine's cloud computing capability	Can scale design to deal with very large projects that include millions of objects
Innoslate® security layer with Google App Engine security layer and SSL	Provides secure environment for data at rest or use Client-Server version
Share models and parsed documents via Sharespace	Reduces rework of commonly used documents and models
Automatic upgrades; no installation; runs on any computer or device (e.g., iPad)	Reduces IT support costs and trouble significantly
Responsive to user requests	Provides new features for users when they need them
Monthly payment plan	Buy what you need, when you need it

For a Complete Comparison

- Tools that Innoslate can replace
  - Requirements Management (e.g., DOORS)
  - Modeling (e.g., CRADLE, CORE)
  - Simulation (CORESim, ARENA)
  - File Sharing (e.g., Sharepoint, Windchill)
  - Risk Analysis and Management
  - Project Planning (e.g., MS Project)
- Be sure to compare combined cost, as well as features