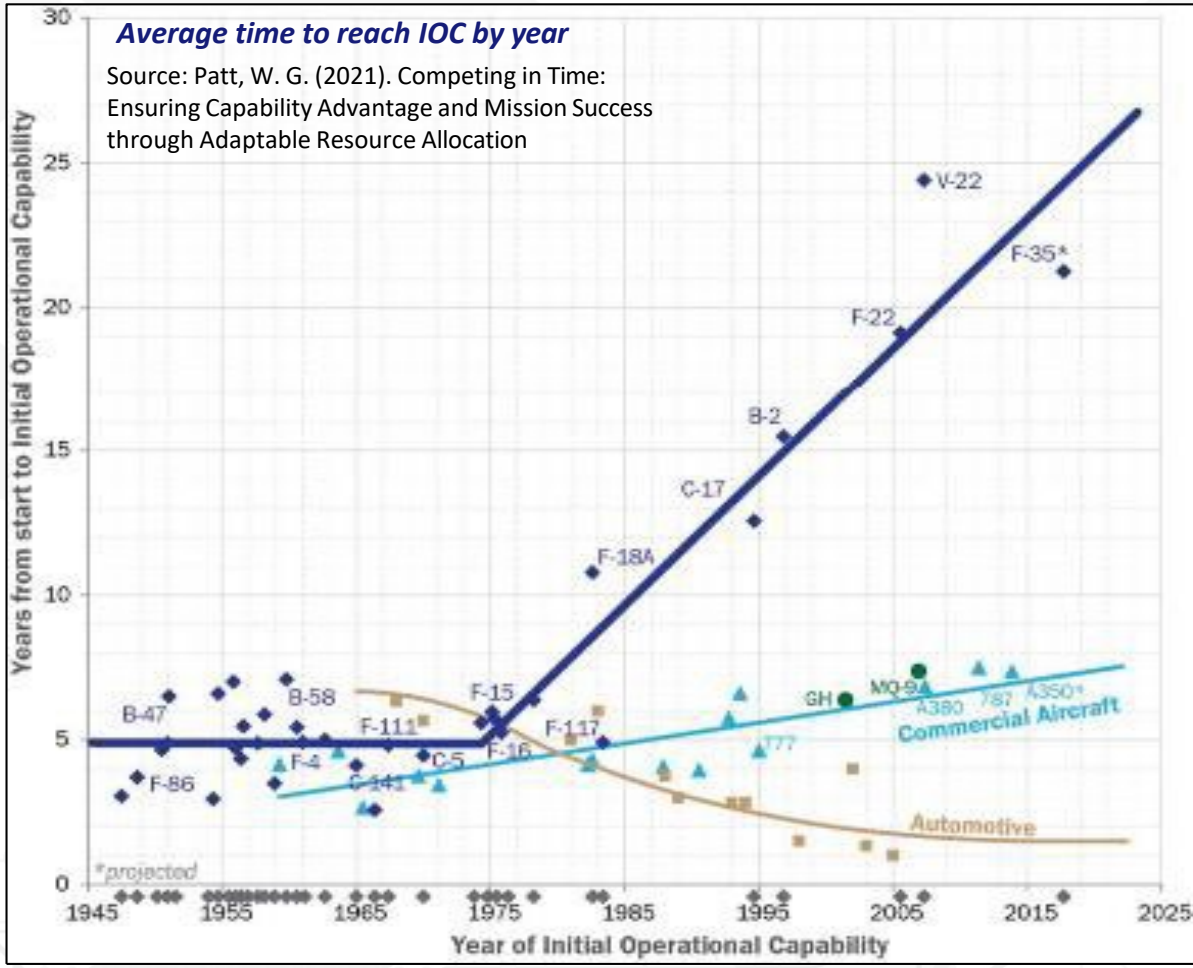
Keith Lucas, DTO Modeling Lead

14 Aug, 2023

Strategic 'Why': Competing in Time

- "it takes the **US on average sixteen years** to deliver an idea to operational capability, versus **fewer than seven for China**"
- "The **PPBE's inflexibility increases the difficulty of rapidly shifting funding to emergent innovations**"
- "Defense acquisition process and legacy defense industrial base approach struggle to accommodate timely adoption of these emerging technologies"
- "**Competitive advantage** in decision-centric operations (whether budgeting or on the battlefield) **comes from the scale of available options, tempo of decision-making, and superior decision processes**"

Digital Transformation yields smarter, faster decision making; but flexible funding and agility in HOW we resource is essential

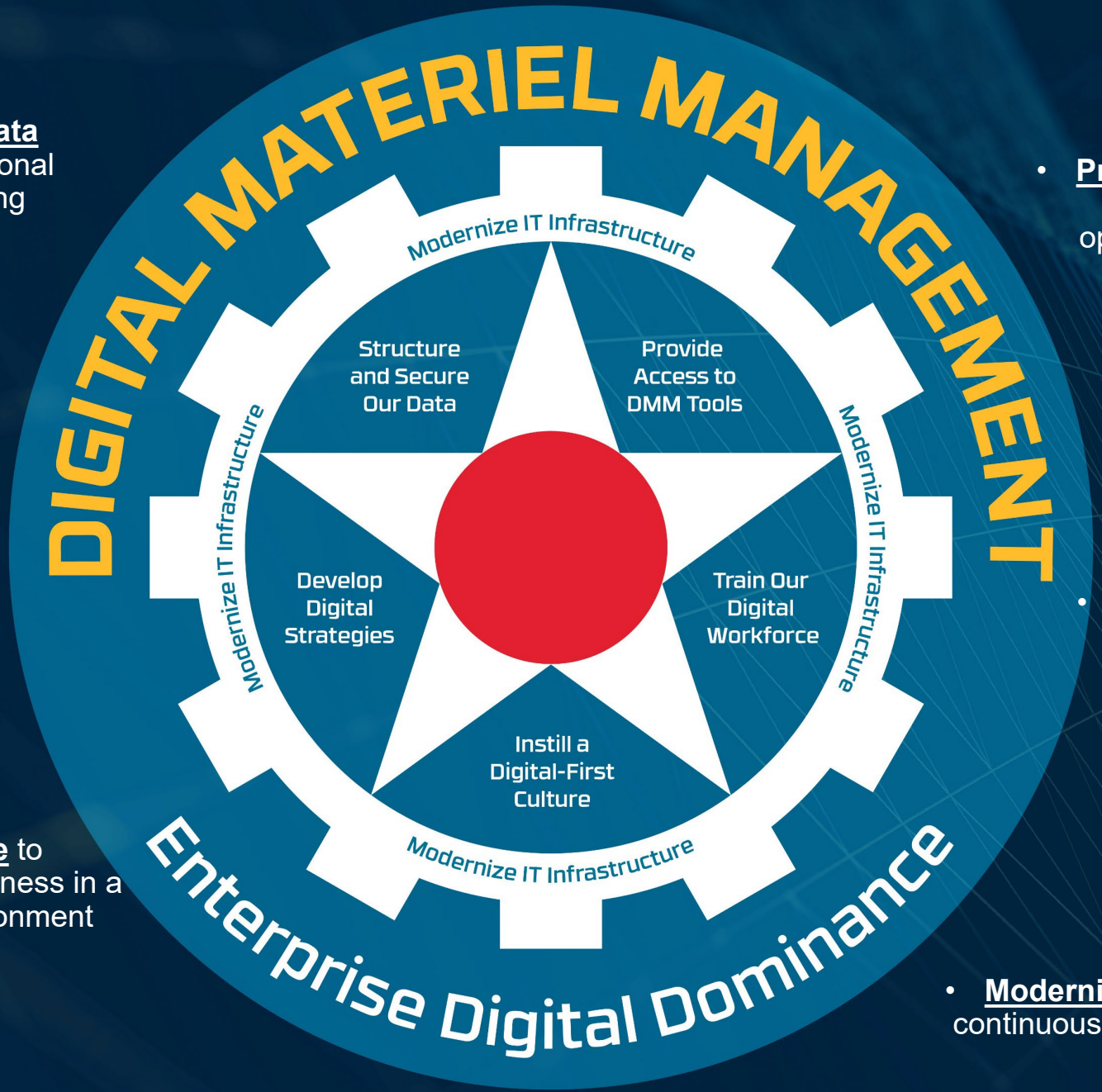


Revolutionize Our Processes via DMM

- **Structure and Secure Our Data** for low friction, cross-organizational teamwork and decision-making

- **Develop Digital Strategies** to leave behind stale practices and pave the way for agile acquisition & sustainment

- **Instill a Digital-First Culture** to revolutionize how AFMC does business in a constantly changing threat environment



- **Provide Access to DMM Tools** to equip our workforce for digital operations with a dynamic toolbox

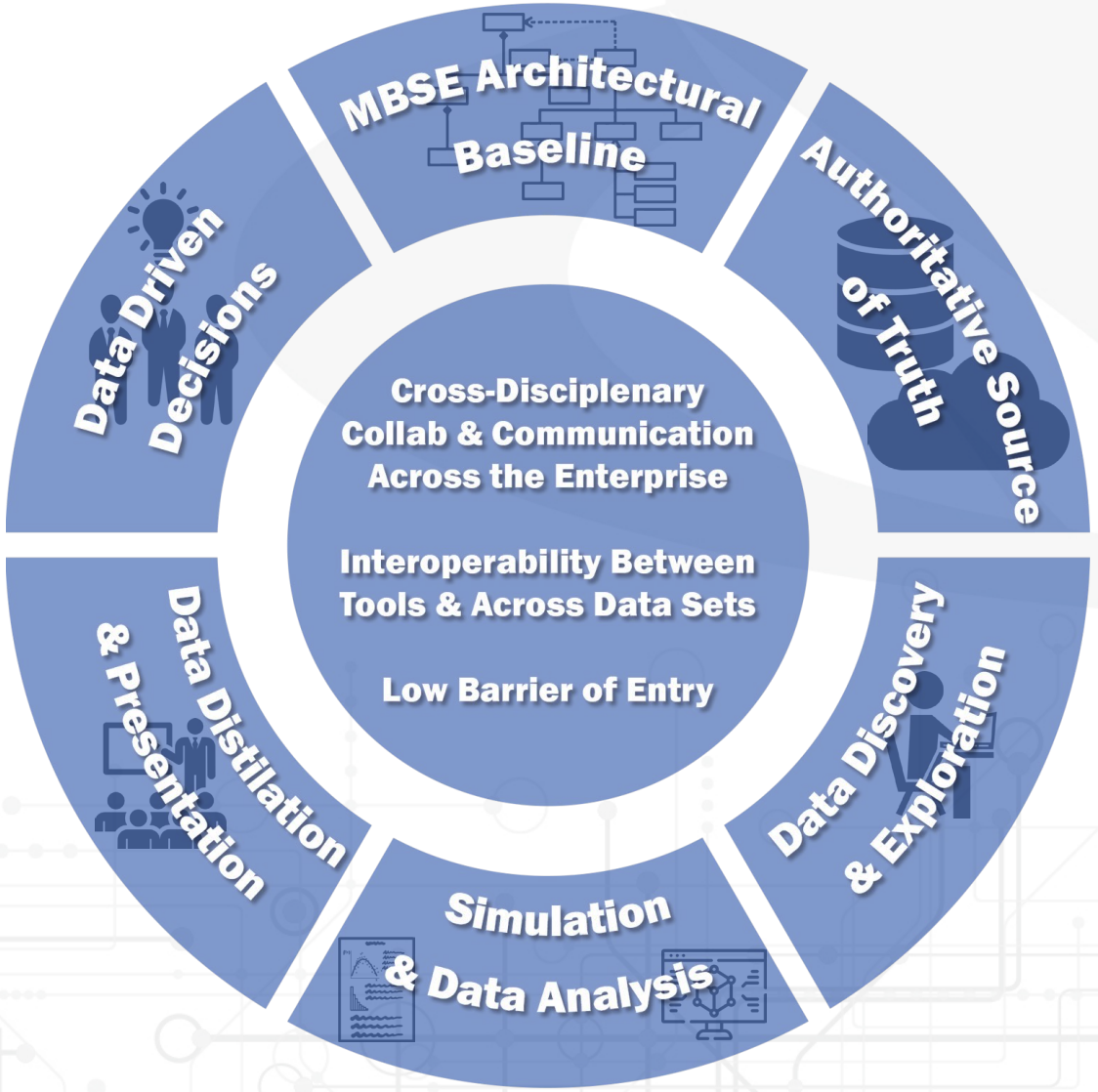
DMM Video

- **Train Our Digital Workforce** so we are prepared to collaborate with partners in a fully digital ecosystem

- **Modernize IT Infrastructure** to continuously enable rapid enterprise solutions

Video OV-1

DMM In Action



DTO

Challenge Areas for DMM

Aging IT Infrastructure and Difficult Pathways for Upgrade

- Obstacles drive slow digital adoption

Modern Data Analysis Tools have Limited Availability on Gov't Networks

- Difficult to extract salient data
- Difficult to present digestible knowledge

Dated But Critical Doc Based Processes

- Multiple proprietary data formats
- Open data formats are limited
- Limited portability between tools

Culture is Slow to Change

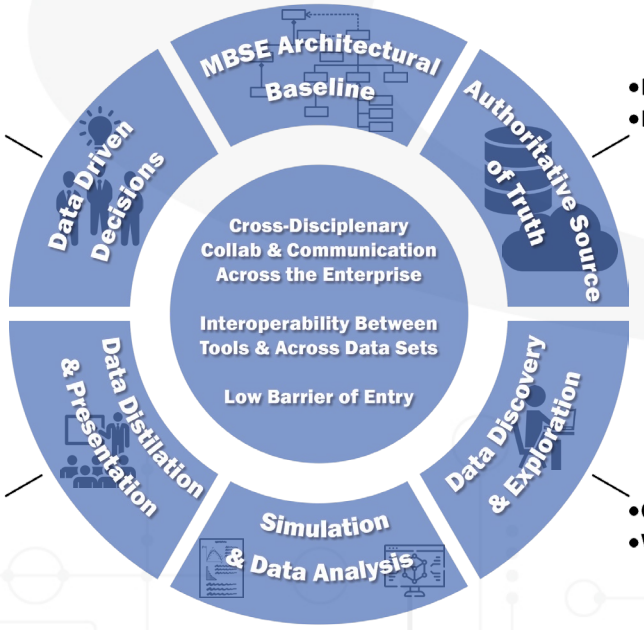
- Requires additional costly, complex plugins
- Each modeling tool has its own solution

Difficult Hiring Processes for Gov't Personnel

- Cannot interact w/ model in native tools
- Workforce requires extensive training

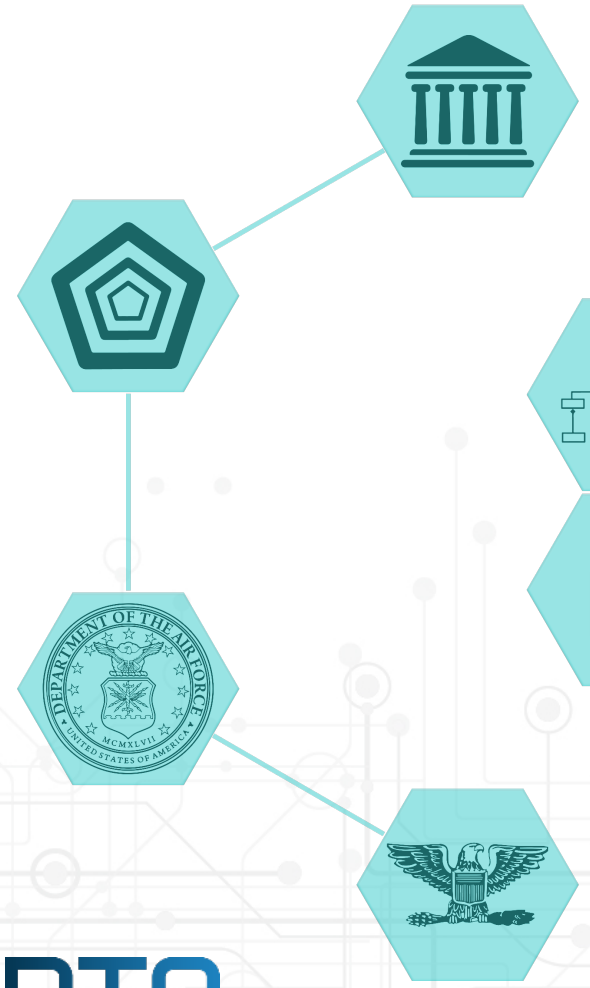
Lack of Trained Personnel in Key Skillsets

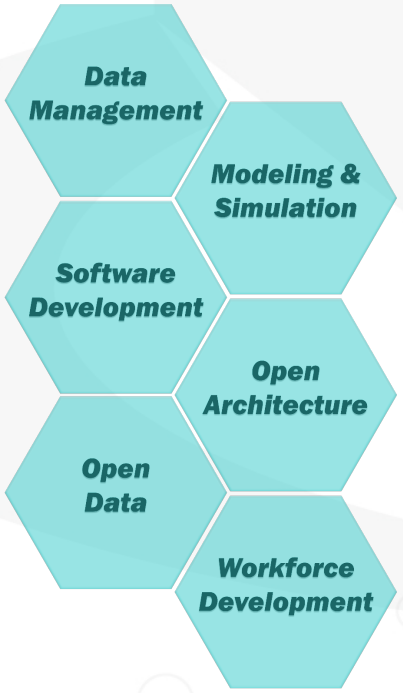
- Lack of interoperability between modeling & analysis tools
- MBSE data formats become obstacles requiring extensive engineering effort



DTO

Digital Transformation Enterprise





Roles

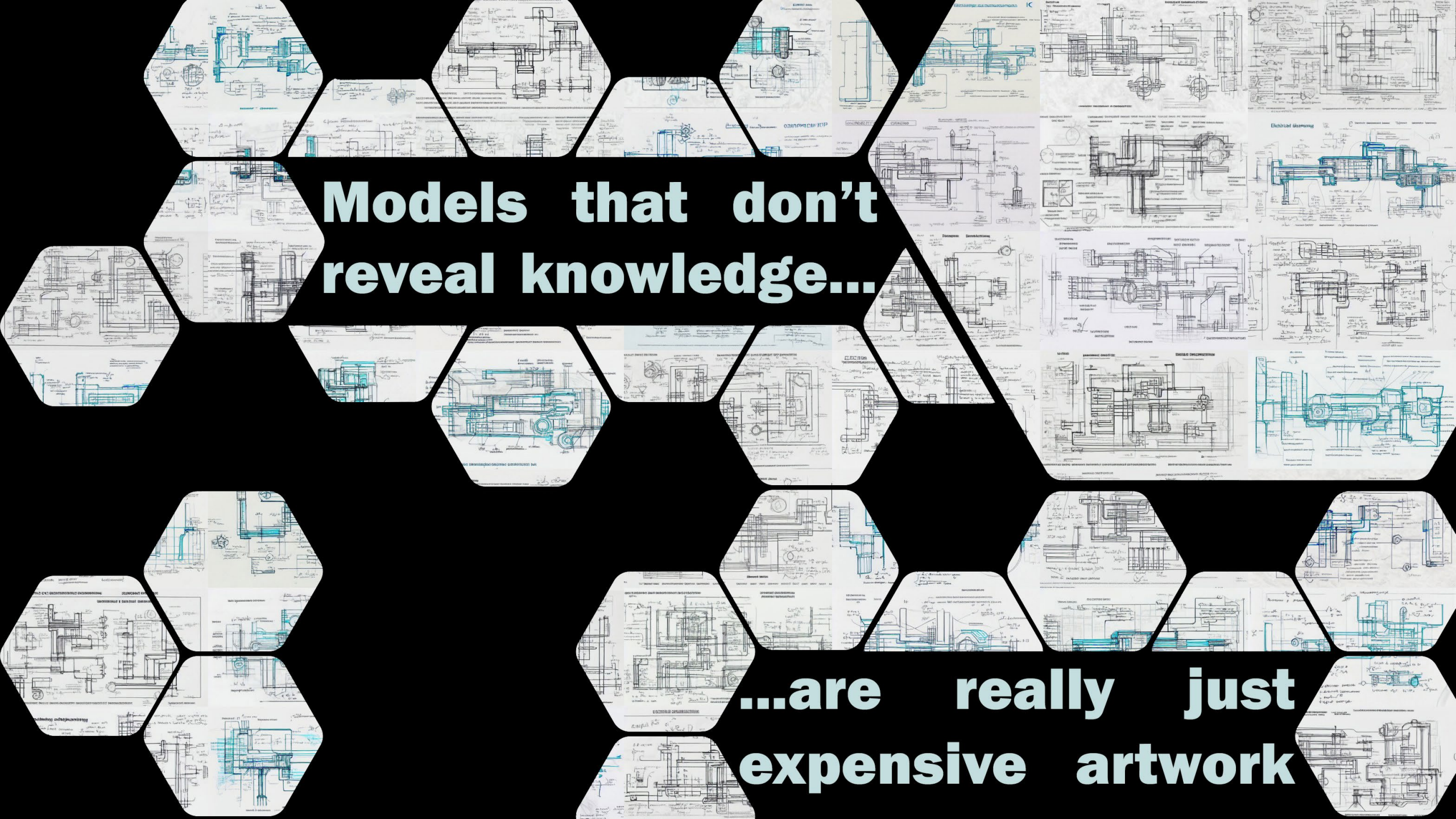
- *Executive Leadership*
- *Program Leadership*
- *Systems Engineers*
- *Program Managers*
- *Contracting Officers*
- *Test & Evaluation*
- *Logisticians*
- *Data Scientists*
- *Systems Architects*
- *Cyber Security Experts*
- *Software Engineers*
- *Schedulers*
- *Financial Managers*
- *Operations Research Analysts*
- *Physicists*
- *Electrical / Mechanical Engineers*
- *Intelligence Community*
- *Maintainers*
- *Trainers*

Skillsets / Tasks

- *Model Based Systems Engineering*
- *Cost Modeling*
- *Monte Carlo Sim*
- *Discrete Event Sim*
- *Artificial Intelligence / Machine Learning*
- *CAD Modeling*
- *DevSecOps*
- *Cloud Infrastructure*
- *Airworthiness / Nuclear Cert*
- *Physics Based Modeling*
- *Cyber Certification*
- *Red / Blue Wargaming*
- *Data Management*
- *Electrical / Mechanical Engineering*
- *Software Engineering*
- *Design of Experiments*
- *Project Management*

Software and Data

DTO


Models that don't reveal knowledge...
...are really just expensive artwork

PySysML2

DTO

Air Force Institute of Technology

Integrity - Service - Excellence

PySysML2

SysML v2 Python Integration

Keith L Lucas

Digital Transformation Office

6 Sep 2022

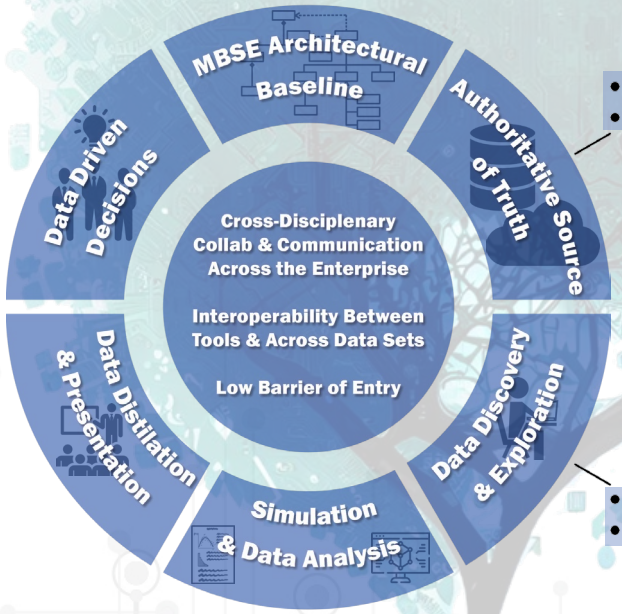
keith.lucas.3@us.af.mil

Acad. Advisor: Lt Col John Situ, Ph.D

Tech. Advisor: Thomas Ford, Ph.D

DTO

SysML 2.0 Addresses Multiple Challenges



- Multiple proprietary data formats
- Open data formats are limited
- Limited portability between tools

- Built on open, extensible, textual language
- APIs are well documented, freely available

- Requires additional costly, complex plugins
- Each modeling tool has its own solution

- Models can be expressed as source code
- Supports widely used version controlsoftware (e.g. Git)

- Cannot interact w/ model in native tools
- Workforce requires extensive training

- Can use general data science tools (e.g. Jupyter NB)
- Compatible with more general STEM skillsets

- Lack of interoperability between modeling & analysis tools
- MBSE data formats become obstacles requiring extensive engineering effort

- Interoperability between tools is a core requirement
- Well documented, FOSS project supports community innovation

DTO

The Model Interoperability Problem

Integrating MBSE Models is Hard!

File formats are mostly proprietary, & current open formats are broken

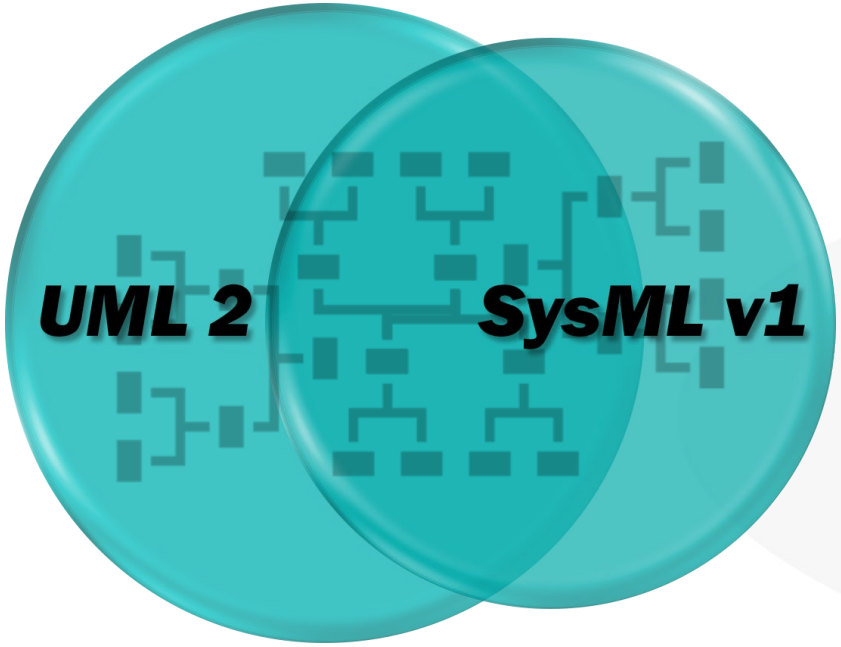
---

Difficulty & high cost of model interoperability limits MBSE utility

The Digital Thread Breaks at the Interface with MBSE Models

DTO

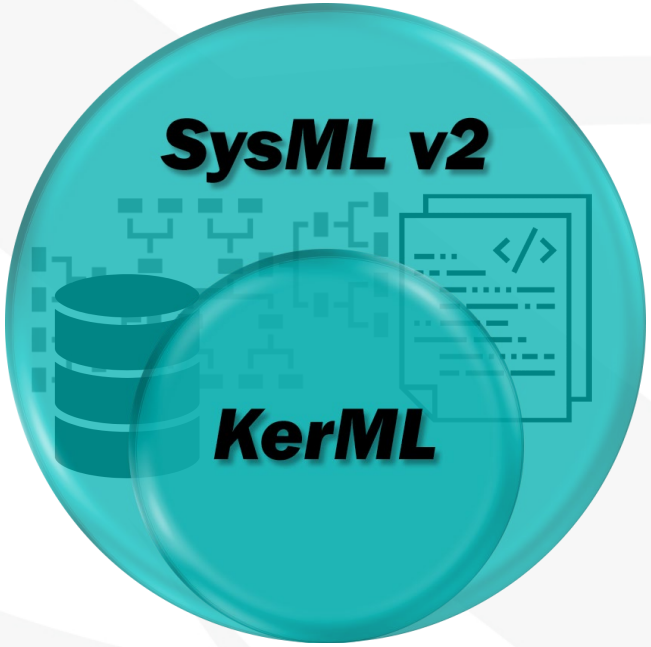
Interoperability: A SysML v2 Requirement



SysML v1 is an extension of UML2.0

Primarily defined as a graphical language

Tool vendors built proprietary, black box computational back ends and standards



SysML v2 is an extension of KerML

Graphical language founded on top of a textual source language and RESTful API

Interoperability with other tools is a requirement in the SysML v2 RFP

Low Interoperability Locks Gov into Modeling Tools, Siloes Models from Analysis Tools, & Limits Insight & Knowledge

PySysML2: Python Interoperability Pathfinder

But Why Python?

Python supports a vast ecosystem of data science, simulation, & analysis tools

Python is among the most widely used, readable, & flexible programming languages

Most STEM grads have some Python training and experience using it for analysis

Python is free and open source, while also strongly supported by industry and academia

PYSysML2 Goals

PySysML2 provides a Pythonic wrapper for the SysML v2 textual modeling language

PySysML2 interfaces SysML v2 models with the Python data science ecosystem

PySysML2 is designed to be maintainable & extendable as SysML v2 changes and grows

PySysML2 is free and open source, hoped to drive more MBSE open source development

DTO

PySysML2 Use Cases

1
Read SysML v2 Models into the Python Environment

Language recognition & grammar parsing

2
Access & Manipulate SysML v2 Models in the Python Environment

Python Tree data structure for SysML v2 model implementation

Multidimensional arrays for numerical analysis

Dataframe datastructure for relational analysis

3
Support SysML v2 Model Serialization & Portability

Model serialization for transfer & storage

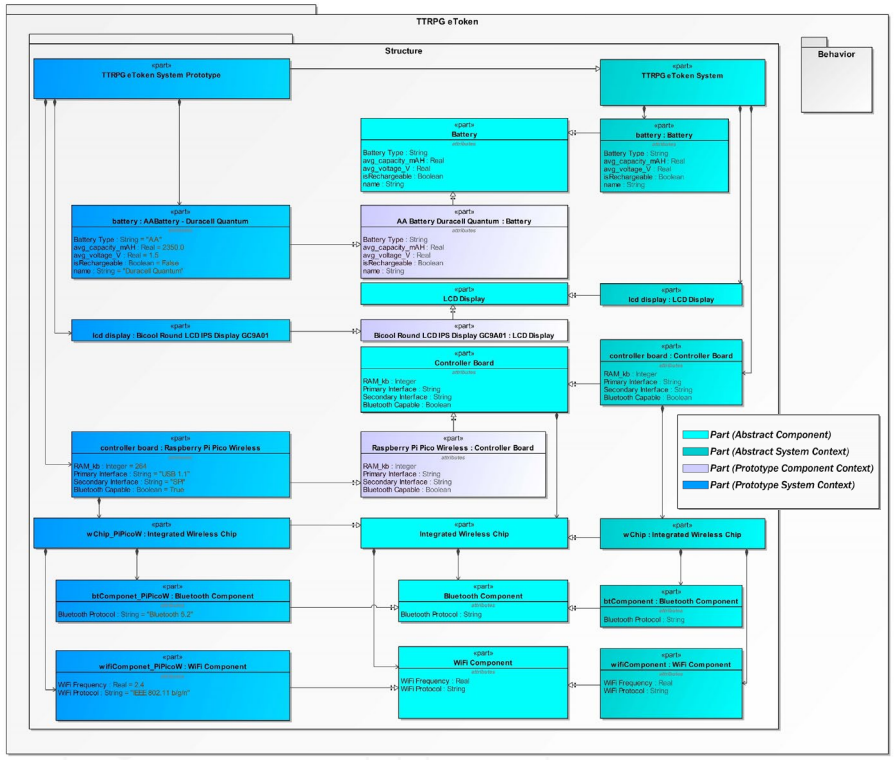
Model tabularization for analysis

MATLAB compatibility

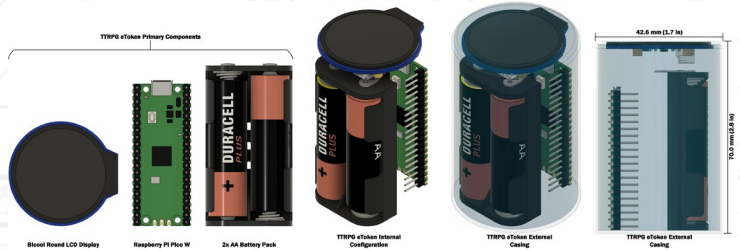
DTO

SysML V2 Example

SysML v2 Graphical Model



Notional System: TTRPG eToken



SysML v2 Graphical Model

Lines 1-66

Line	SysML v2 Textual Code
1	// References --
2	// Intro to the SysML v2 Language-Textual Notation.pdf
3	// https://github.com/Systems-Modeling/SysML-v2-Release/tree/master/doc
4	import ISQ::*;
5	import ISQSpaceTime;
6	import ScalarValues::*;
7	
8	package TTRPGeToken{
9	doc overview /*
10	* The TTRPGeToken is a device used for displaying NPC/PC
11	* avatars in a physical token that can be used on a tabletop
12	*/
13	doc /* TODO: include links to remotely hosted images */
14	doc /* TODO: include links to public facing documentation */
15	comment RevComment_1 /* TO: Maatlock: Please evaluate your vigorous use of
16	* commenting in the TTRPGeToken model. Comment variety
17	* in the language seems... excessive. Don't get
18	* carried away!
19	*/
20	package Structure{
21	doc overview /* Structural elements of model */
22	// This is an example of composite structures, [1] pg. 16
23	part def 'WiFi Component';
24	part def 'Bluetooth Component';
25	part def 'Integrated Wireless Chip' {
26	attribute name : String;
27	part wifiComponent : 'WiFi Component' {
28	attribute 'WiFi Frequency' : Real;
29	attribute 'WiFi Protocol' : String;
30	}
31	part btComponent : 'Bluetooth Component'{
32	attribute 'BT Protocol' : String;
33	}
34	}
35	part def 'Controller Board' {
36	part def 'wChip' specializes 'Integrated Wireless Chip';
37	attribute 'RAM_kb' : Integer;
38	attribute 'Primary Interface' : String;
39	attribute 'Secondary Interface' : String;
40	attribute 'Bluetooth Capable' : Boolean;
41	}
42	part def 'LCD Display' {}
43	part def 'Battery' {
44	attribute isRechargeable : Boolean;
45	attribute 'Battery Type' : String;
46	attribute name : String;
47	attribute avg_voltage_V : Real;
48	attribute avg_capacity_mAh : Real;
49	}
50	part def 'Raspberry Pi Pico Wireless' specializes 'Controller Board' {
51	doc info /*https://en.wikipedia.org/wiki/Raspberry_Pi*/
52	part def wChip_PiPicoW :> wChip{
53	attribute redefines name : String = "Infineon CYW43439";
54	part wifiComponent_PiPicoW :> wifiComponent{
55	attribute redefines 'WiFi Frequency': Real = 2.4;
56	attribute :> 'WiFi Protocol': String = "IEEE 802.11 b/g/n";
57	}
58	part btComponent_PiPicoW :> btComponent{
59	attribute redefines 'BT Protocol' : String="Bluetooth 5.2";
60	}
61	}
62	attribute :> 'RAM_kb' = 264;
63	attribute :> 'Bluetooth Capable' = true;
64	attribute :> 'Primary Interface' = "USB 1.1";
65	attribute redefines 'Secondary Interface' = "SPI";
66	}

Lines 67-123

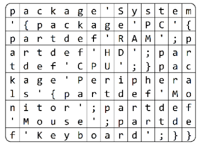
Line	SysML v2 Textual Code
67	part def 'AA Battery Duracell Quantum' specializes 'Battery' {
68	attribute :> isRechargeable : Boolean = false;
69	attribute :> 'Battery Type' : String = "AA";
70	attribute :> name : String = "Duracell Quantum";
71	attribute :> avg_voltage_V : Real = 1.5;
72	attribute :> avg_capacity_mAh : Real = 2350.0;
73	}
74	// Specialize using the "subset" symbol
75	part def 'Bicool Round LCD IPS Display GC9A01' :> 'LCD Display' {}
76	
77	part def 'TTRPG eToken System' {
78	part 'controller board' : 'Controller Board';
79	part 'lcd display' : 'LCD Display';
80	part 'battery' : 'Battery'[1..2];
81	}
82	part def 'TTRPG eToken System Prototype' {
83	part 'controller board' : 'Raspberry Pi Pico Wireless';
84	part 'lcd display' : 'Bicool Round LCD IPS Display GC9A01';
85	part 'battery' : 'AA Battery Duracell Quantum';
86	}
87	}
88	package Behavior{
89	part def 'User' {}
90	use case def 'Change displayed image on eToken'{
91	actor 'user' : 'User';
92	objective {
93	doc /*
94	* The user changes the displayed image on the eToken to one
95	* that is currently stored in storage
96	*/
97	}
98	}
99	use case def 'Remove existing image form eToken'{
100	objective {
101	doc /*
102	* The user deletes an image from the eToken currently
103	* stored in storage
104	*/
105	}
106	actor 'user' : 'User';
107	}
108	use case def 'Load new image to eToken'{
109	objective {
110	doc /*The user uploads a new image to the eToken's storage*/
111	}
112	actor 'user' : 'User';
113	}
114	use case def 'Use eToken as game piece'{
115	objective {
116	doc /*The user places the eToken on the board to use as a
117	*game piece
118	*/
119	}
120	actor 'user' : 'User';
121	}
122	}
123	}
124	
125	
126	
127	
128	
129	
130	

PySysML2 Software Engineering

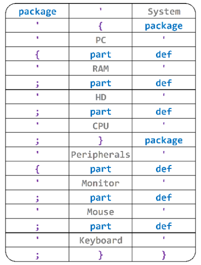
SysML v2 Grammar

```
package 'System'{
    package 'PC'{
        part def 'RAM';
        part def 'HD';
        part def 'CPU';
    }
    package 'Peripherals'{
        part def 'Monitor';
        part def 'Mouse';
        part def 'Keyboard';
    }
}
```

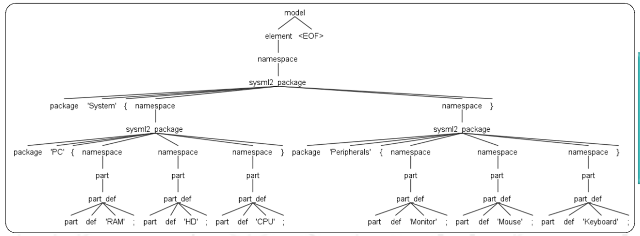
1 Read in Character Stream



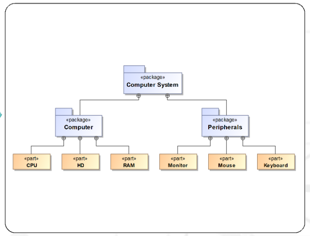
2 Tokenize Characters with Lexer



3 Build Parse Tree from Tokens



Build Model From Parse Tree 4



Grammar Declaration

```
grammar SysML2;
//----------------------------------------------------------------------------------
```

Model Definition

```
// Model, i.e. collection of elements to EOF, e.g. namespaces, features, etc
model: element* EOF;
// An element is anything that can be a part of a model
element : namespace | feature | comment | doc | statement;
//----------------------------------------------------------------------------------
// Namespaces, i.e. elements with a scope defined by curly braces
```

Namespace Definition

```
namespace   : sysml2_package | part | use_case_def | comment | doc;
sysml2_package: KW_PACKAGE ID '{' namespace* '}';
// Parts
```

Part Definition

```
part_blk: (feature | comment | doc | part_def_specializes);
part: (part_def | part_def_specializes);
part_def: ((KW_PART KW_DEF ID '{' part_blk* '}')|(KW_PART KW_DEF ID';'));
part_def_specializes: KW_PART KW_DEF? ID
                      (KW_SPECIALIZES | KW_SYM_SUBSETS) ID
                      (',' ID)*? ('{' part_blk* '}' | ';');
// Use Cases
```

Use Case Definition

```
use_case_blk: part_blk | objective_def;
use_case_def: KW_USE KW_CASE KW_DEF ID '{' use_case_blk* '}';
part_objective_blk: doc;
objective_def: KW_OBJECTIVE '{' part_objective_blk '}';
//----------------------------------------------------------------------------------
// Features, i.e. elements that can be part of a namespace
```

Feature Definition

```
feature : feature_attribute_def | feature_attribute_redefines
        | feature_part_specializes | feature_part_specializes_subsets
        | feature_item_def | feature_item_ref
        | feature_actor_specializes;
// Attributes
```

Attributes

```
feature_attribute_def: KW_ATTRIBUTE ID ':' TYPE ';';
feature_attribute_redefines: KW_ATTRIBUTE
                             (KW_REDEFINES | KW_SYM_REDEFINES | KW_SYM_SUBSETS)
                             ID (':' TYPE)? '=' CONSTANT ';';
```

Specialization

```
feature_part_specializes: KW_PART ID ':' ID MULTIPLICITY?
                          (';' | '{' part_blk* '}');
feature_part_specializes_subsets: KW_PART ID ':' ID MULTIPLICITY?
                                  (KW_SUBSETS | KW_SYM_SUBSETS) ID';';
```

Item Definition

```
feature_item_def: KW_ITEM ID';';
feature_item_ref: KW_REF? KW_ITEM ID ':' ID';';
```

Item Definition

```
feature_actor_specializes: KW_ACTOR ID ':' ID MULTIPLICITY?';';
// SysML2 Comments and Documentation
```

Documentation and Comments

```
comment : comment_unnamed | comment_named | comment_named_about;
comment_unnamed: COMMENT_LONG;
comment_named: KW_COMMENT ID COMMENT_LONG;
comment_named_about: KW_COMMENT KW_ABOUT ID COMMENT_LONG;
doc : doc_unnamed | doc_named;
doc_unnamed: KW_DOC COMMENT_LONG;
doc_named: KW_DOC ID COMMENT_LONG;
// Statements
```

Statements

```
statement : import_package;
```

Import

```
import_package: KW_IMPORT ID (KW_SYM_FQN ID)* (KW_SYM_FQN '*')? ';';
//----------------------------------------------------------------------------------
// Keywords and Tokens
```

Token Definition

```
KW_ABOUT: 'about';
KW_ACTOR: 'actor';
KW_ATTRIBUTE: 'attribute';
KW_CASE: 'case';
KW_COMMENT: 'comment';
KW_DEF: 'def';
KW_DOC: 'doc';
KW_IMPORT: 'import';
KW_ITEM: 'item';
KW_OBJECTIVE: 'objective';
KW_PACKAGE: 'package';
KW_PART: 'part';
KW_REDEFINES: 'redefines';
KW_REF: 'ref';
KW_SPECIALIZES: 'specializes';
KW_SUBJECT: 'subject';
KW_SUBSETS: 'subsets';
KW_USE: 'use';
KW_SYM_FQN: '::';
KW_SYM_REDEFINES: ':>>';
KW_SYM_SUBSETS: ':>';
CONSTANT: INTEGER | REAL | BOOL | STRING | NULL;
TYPE: 'Integer' | 'Real' | 'Boolean' | 'String';
// Characters
```

Character Defintion

```
ID: '\'' [ a-zA-Z_][ a-zA-Z0-9_]* '\'' | [a-zA-Z_][a-zA-Z0-9_]*;
INTEGER: [0-9]+;
REAL: [0-9]+ '.' [0-9]+;
BOOL: 'true' | 'false';
STRING: '"' (ESC | ~ ["\\])* '"';
MULTIPLICITY: '[' INTEGER '..' INTEGER ']';
fragment ESC: '\\' (["\\/bfnrt] | UNICODE);
fragment UNICODE: 'u' HEX HEX HEX HEX;
fragment HEX: [0-9a-fA-F];
NULL: 'null';
WS: [ \t\r\n]+ -> skip;
NOTE: '//' ~[\r\n]* -> skip;
COMMENT_LONG: '/*'.*?'*/';
```

PySysML2 Software Engineering

```
package 'System'{
    package 'PC'{
        part def 'RAM';
        part def 'HD';
        part def 'CPU';
    }
    package 'Peripherals'{
        part def 'Monitor';
        part def 'Mouse';
        part def 'Keyboard';
    }
}
```

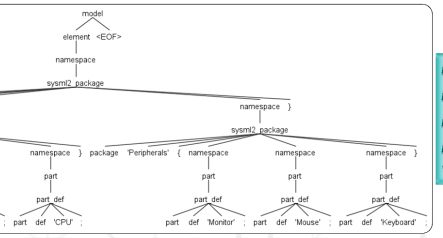
1 Read in Character Stream

p	a	c	k	a	g	e	'	S	y	s	t	e	m
'	{	p	a	c	k	a	g	e	'	P	C	'	{
p	a	r	t	d	e	f	'	R	A	M	'	;	p
a	r	t	d	e	f	'	H	D	'	;	p	a	r
t	d	e	f	'	C	P	U	'	;	}	}	p	a
c	k	a	g	e	'	P	e	r	i	p	h	e	r
a	l	s	'	{	p	a	r	t	d	e	f	'	M
o	n	i	t	o	r	'	;	p	a	r	t	d	e
f	'	M	o	u	s	e	'	;	p	a	r	t	d
e	f	'	K	e	y	b	o	a	r	d	'	;	}
}													

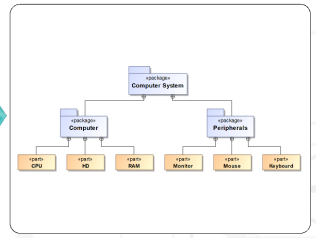
2 Tokenize Characters with Lexer

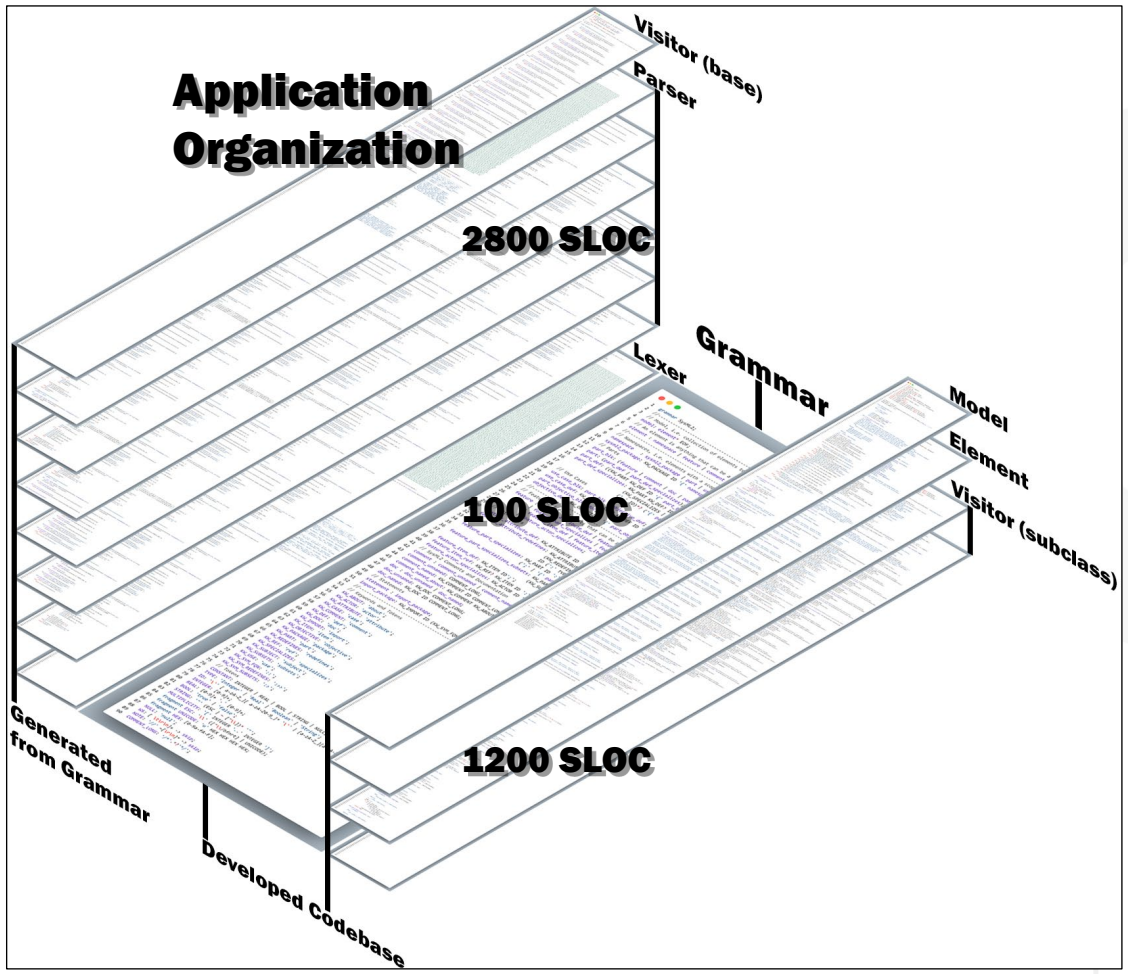
package	'	System
'	{	package
'	PC	'
{	part	def
'	RAM	'
;	part	def
'	HD	'
;	part	def
'	CPU	'
;	}	package
'	Peripherals	'
{	part	def
'	Monitor	'
;	part	def
'	Mouse	'
;	part	def
'	Keyboard	'
;	}	}

3 Build Parse Tree from Tokens



4 Build Model From Parse Tree





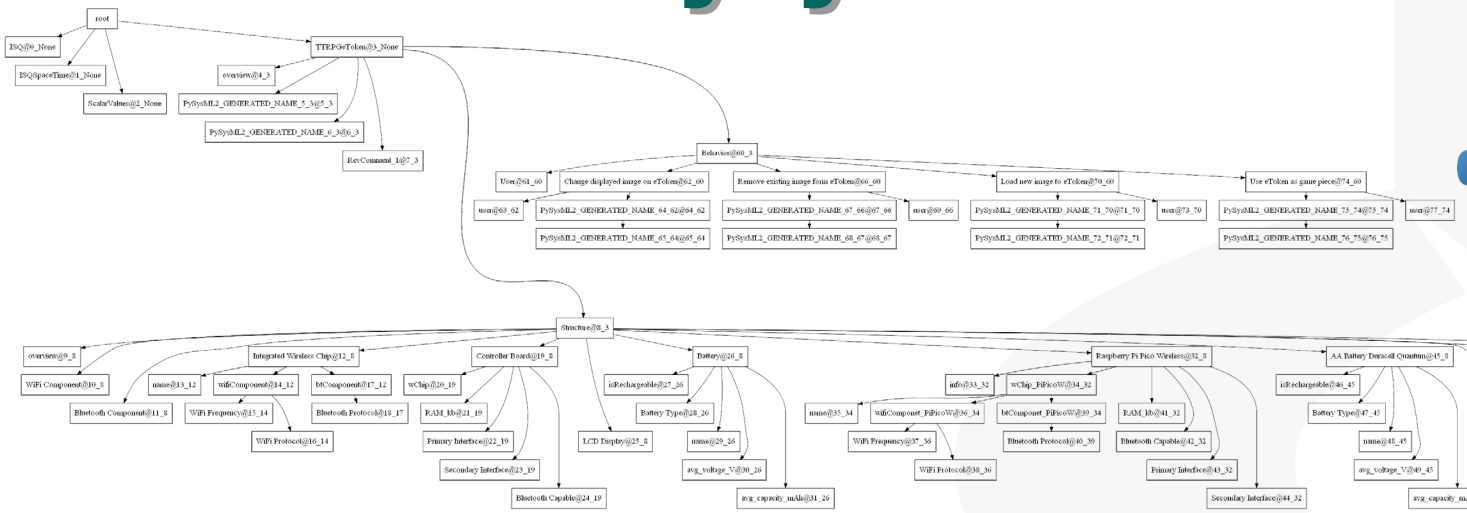
The ANTLR4 based grammar parser is the heart PySysML2

Small subset of SysML v2 supported, but easily extensible

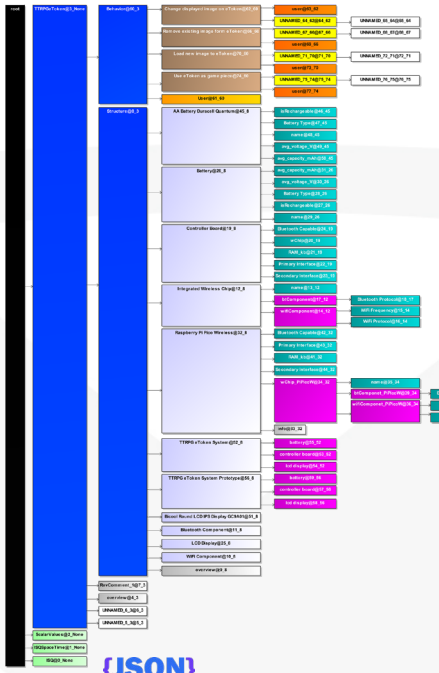
General Processing Workflow

1. Parse tree is generated from SysML v2 textual source
2. Model is built from the parse tree as a Python object in memory
3. From this point, model may be transformed to one of many interoperable data structures

PySysML2 Data Structures

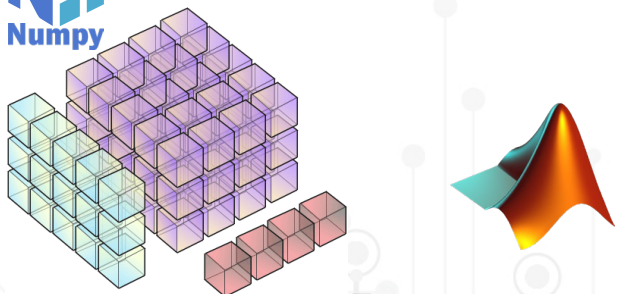


Python Tree data structure for SysML v2 model implementation

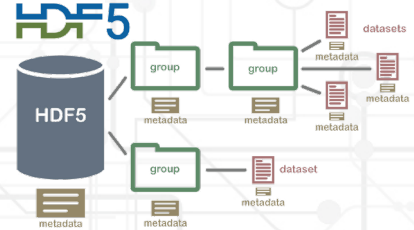


Numpy

Multidimensional arrays for numerical analysis

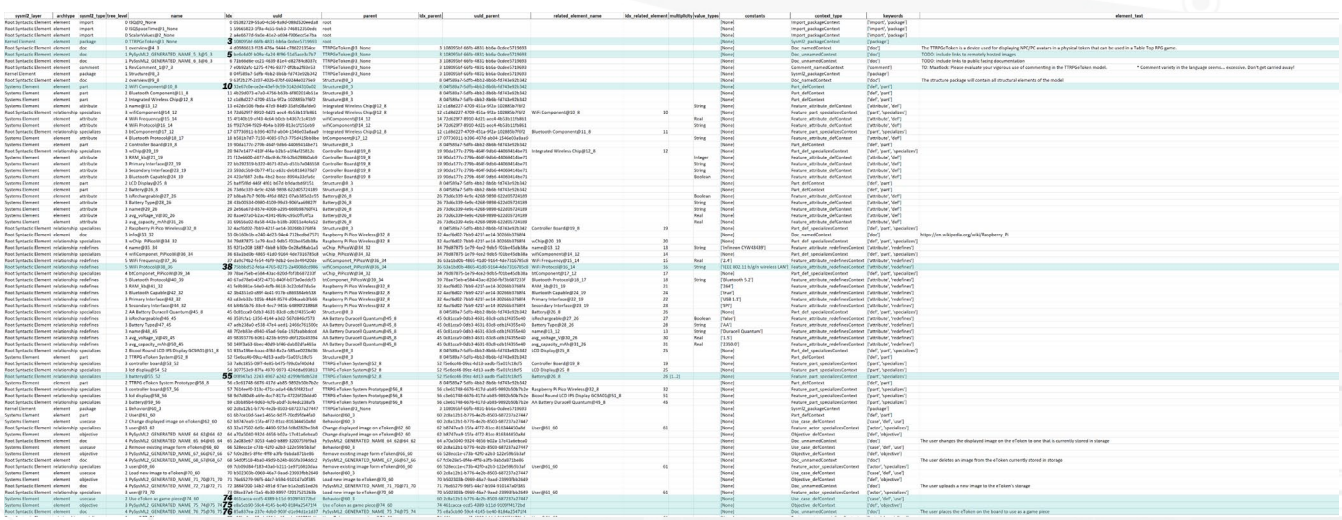


HDF5



pandas

Dataframe datastructure for relational analysis



CSV

Model tabularization

{JSON}

Model serialization for transfer & storage

DTO

SysML v2 Overview

Next generation of Systems Modeling Language

- Addresses many systemic issues of SysML v1.x
- Development driven by the following requirements:
  - **Precision** and **expressiveness** of the language
  - **Consistency** and integration among language concepts
  - **Interoperability** with other engineering models and tools
  - **Usability** by model developers and consumers
  - **Extensibility** to support domain specific applications
  - **Migration path** for SysML v1 users and implementors

SysML v2 Overview

SysML v2 is coming… quickly

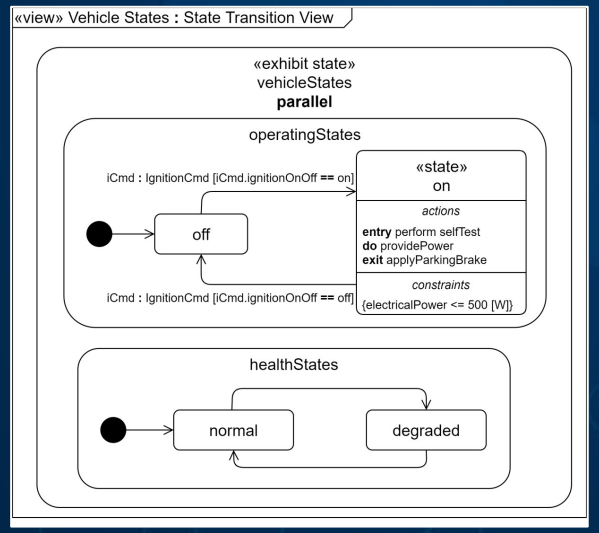
- Tech spec approval completed, OMG conference, Feb 2023
- Adoption of “beta” standard expected in Summer 2023 (soon)
- Final adoption expected in 2024…
- OMG will begin sunsetting SysML v1.x (latest version 1.7)

SysML v2 Overview

- Includes both **graphical** & **textual** modeling languages, & a modern, standardized **API**

```
exhibit state vehicleStates parallel {
    state operatingStates {
        entry action initial;
        state off;
        state on {
            entry action performSelfTest;
            do providePower;
            exit action applyParkingBrake;
            constraint {electricalPower<=500[W]}
        }
        transition initial then off;
        transition off_To_on
            first off
            accept ignitionCmd:IgnitionCmd via ignitionCmdPort
                if ignitionCmd.ignitionOnOff==IgnitionOnOff::on
            then on;
        transition on_To_off
            first on
            accept ignitionCmd:IgnitionCmd via ignitionCmdPort
                if ignitionCmd.ignitionOnOff==IgnitionOnOff::off
            then off;
    }
    state healthStates {
        entry action initial;
        state normal;
        state degraded;
```

Textual Modeling Language



Graphical Modeling Language

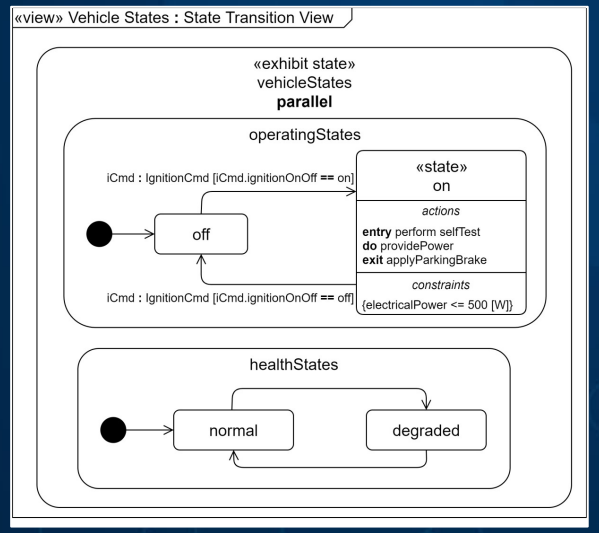
Open API

SysML v2 Overview

- Includes both **graphical** & **textual** modeling languages, & a modern, standardized **API**

```
exhibit state vehicleStates parallel {
    state operatingStates {
        entry action initial;
        state off;
        state on {
            entry action performSelfTest;
            do providePower;
            exit action applyParkingBrake;
            constraint {electricalPower<=500[W]}
        }
        transition initial then off;
        transition off_To_on
            first off
            accept ignitionCmd:IgnitionCmd via ignitionCmdPort
                if ignitionCmd.ignitionOnOff==IgnitionOnOff::on
            then on;
        transition on_To_off
            first on
            accept ignitionCmd:IgnitionCmd via ignitionCmdPort
                if ignitionCmd.ignitionOnOff==IgnitionOnOff::off
            then off;
    }
    state healthStates {
        entry action initial;
        state normal;
        state degraded;
```

Textual Modeling Language

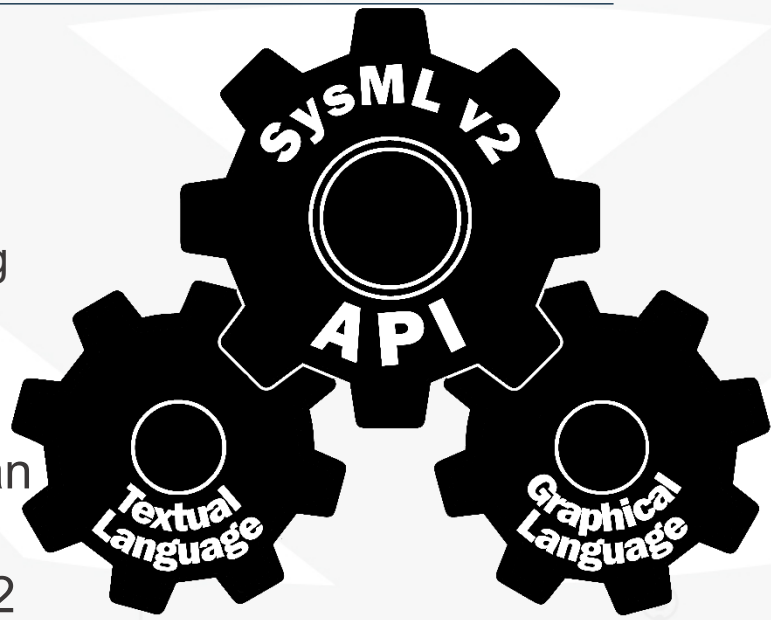


Graphical Modeling Language

Open API

Key Features Enabled by SysML v2

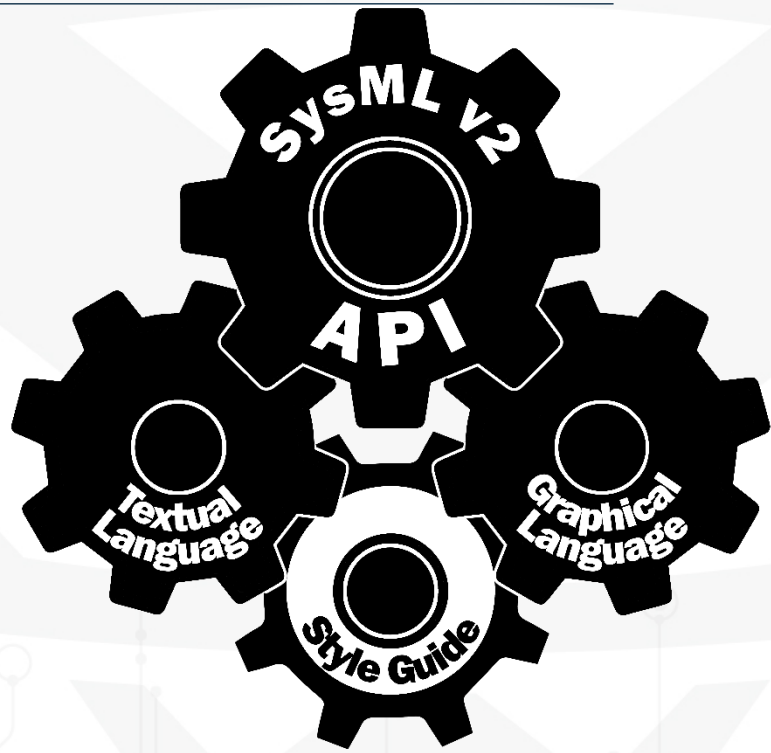
- **Interoperability across Modeling, Simulation, and Analysis Tools:** SysML v2 defines a common API and language implementation that all tool vendors must adopt, rather than leaving the implementation to the individual tool vendors. The API will support web-based interrogation of models across tools. Additionally, a model serialization standard has been built from the ground up in JSON, ensuring greater consistency across tools
- **Managing Models as Code:** SysML v2 implements a “textual” modeling language similar to a programming language. While all models can be expressed graphically as expected, they can also be expressed as human readable text– meaning they can be managed as code
- **Built-in Analytic Capability and Geometric Representation:** SysML v2 includes numerical / quantitative analysis capability in the base implementation, including unit standardization and conversion, in addition to the ability to define simple geometric shapes via spatial coordinates
- **Open-Source Implementation:** A prototype implementation of SysML v2 has been released to the public for free, including textual language parsing and graphical model representation. While the major tool vendors will provide additional capability, the open implementation is robust



DTO

SysML v2 Style Guidance

- **A Style Guide is a comprehensive set of standards agreed upon by an org…**
  - But a Style Guide for one org may not be compatible with another's org's needs
- **Orgs using SysML v2 must adhere to its API, Textual, and Graphical Languages…**
  - But they will choose the best Style Guide to fit their needs or may even build their own
- **As a community of practice, we can adopt a few simple common conventions**
  - Common conventions, while unenforceable, would be advantageous for all Style Guides
- **SysML v2's similarity to coding allows us to learn from other programming languages**
  - Python's recommended Style Guide ("PEP 8") is relatively small and general, covering only the basics:
    - Code Layout; Variable Naming; Commenting & Documentation; and Common Programming Practices
  - It's guiding principle is that "code is read much more than it is written"
- **Models are viewed much more than they are built– users vastly outnumber architects**
  - As in Python, any style conventions should prioritize readability, interoperability, & usability
- **Style Recommendations should be simple, be generalizable, and comply with the API**
  - This makes adoption much more likely– because unadopted style guides are wasted effort



DTO


DTO
DIGITAL TRANSFORMATION OFFICE

MBSE Terms of Art

- **Modeling Language** – a formalized, graphical or textual notation used to represent system models in MBSE. Modeling languages provide the syntax, semantics, and symbols needed to create models that can be easily understood and shared among stakeholders
- **Modeling Framework** – a structured approach, methodology, or environment that supports the creation, management, and analysis of models in MBSE. It defines the organization, relationships, and conventions needed for effective modeling
- **Modeling Standard** – defined as a formal agreement documenting generally accepted specifications or criteria for products, processes, procedures, policies, systems and/or personnel
- **Modeling Profile** – a specific set of customizations or extensions to a modeling language, designed to address the unique needs of a particular domain or industry. Profiles can include new modeling elements, stereotypes, or constraints that tailor the modeling language to a specific context or set of requirements
- **Architecture** – is the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.
- **Reference Architecture** – an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions
- **Government Reference Architecture** – a Government-owned, authoritative source of information about a specific subject area that guides and constrains the instantiations of capability architectures and solutions

MBSE Terms of Art

- **Modeling Language** – a formalized, graphical or textual notation used to represent system models in MBSE. Modeling languages provide the syntax, semantics, and symbols needed to create models that can be easily understood and shared among stakeholders
- **Modeling Framework** – a structured approach, methodology, or environment that supports the creation, management, and analysis of models in MBSE. It defines the organization, relationships, and conventions needed for effective modeling
- **Modeling Standard** – defined as a formal agreement documenting generally accepted specifications or criteria for products, processes, procedures, policies, systems and/or personnel
- **Modeling Profile** – a specific set of customizations or extensions to a modeling language, designed to address the unique needs of a particular domain or industry. Profiles can include new modeling elements, stereotypes, or constraints that tailor the modeling language to a specific context or set of requirements
- **Architecture** – is the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time.
- **Reference Architecture** – an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions
- **Government Reference Architecture** – a Government-owned, authoritative source of information about a specific subject area that guides and constrains the instantiations of capability architectures and solutions

Modeling Languages

- **Systems Modeling Language 1.x (SysML)** – a general-purpose graphical **modeling language** specifically designed for systems engineering applications. It extends the Unified Modeling Language (UML) with additional diagrams & constructs to better represent the system's structure, behavior, & requirements
- **Systems Modeling Language 2.0 (SysML v2)** – evolution of SysML 1.x **modeling language**, offering enhancements in usability and interoperability. Key features include the addition of a textual language for more intuitive model representation, a RESTful API for seamless tool integration, & improvements in model interoperability to facilitate better collaboration & exchange between different modeling and simulation environments & tools
- **Architecture Analysis and design Language (AADL)** – textual & graphical **modeling language** designed for the analysis, specification, & design of real-time, safety-critical, & performance-critical systems. Provides rich set of modeling constructs for structure, behavior, & properties of software, hardware, and hybrid systems
- **Architecture Analysis and design Language (AADL)** – textual & graphical **modeling language** designed for the analysis, specification, & design of real-time, safety-critical, & performance-critical systems. Provides rich set of modeling constructs for structure, behavior, & properties of software, hardware, and hybrid systems
- **Business Process Model & Notation (BPMN)** – graphical **modeling language** designed for representing business processes in a workflow format. Provides standardized set of symbols & notation to describe the flow of activities, events, & decisions within a business process for communication & collaboration between stakeholders

Modeling Frameworks, Profiles, & Standards

- **Unified Architecture Framework (UAF)** – MBSE **framework** that supports the modeling & analysis of complex systems, systems of systems, and enterprises. Built on top of SysML, UAF provides an integrated approach to address the architectural, operational, and technical aspects of a system
- **SysML-UAF Profile** – MBSE **profile** that extends SysML to support the UAF framework, including stereotypes, elements, & relationships to model architectures and interdependencies of systems and enterprises
- **Dept. of Defense Architecture Framework (DoDAF)** – MBSE **framework** used by the United States DoD for architecting & managing complex systems & enterprises. Defines a standardized set of views, products, & guidelines for describing, analyzing, & communicating system architectures. Primarily used with UPDM
- **Ministry of Defense Architecture Framework (MoDAF)** – like DoDAF; used mainly in UK / NATO defense models
- **Unified Profile for DoDAF and MoDAF (UPDM)** – modeling **profile** that integrates DoDAF & MoDAF. Provides a SysML-based notation for creating DoDAF / MoDAF compliant models. Enables DoDAF / MoDAF interoperability
- **Open Services for Lifecycle Collaboration (OSLC)** – set of open **standards** & specifications that enable tools, data, & processes to be integrated across the entire system lifecycle. Promotes collaboration & interoperability across different tools and teams
- **Functional Mock-up Interface (FMI)** – open **standard** that defines a standardized API for model exchange and co-simulation of models across different tools and platforms
- **XML Metadata Interchange (XMI)** – open **standard** specficiation for exchanging MBSE data elements between modeling tools and environments. Defines rules for representing primarily UML & SysML in a serialized XML format, enabling sharing and integration of models across different tools and platforms

DTO Modeling Areas of Interest

- Research MBSE data interoperability across tools and architectures
  - Support development and adoption of open data formats for MBSE
  - Support development and adoption of MBSE modeling standards
- Research MBSE integration between engineering and functional groups
  - Unleash actionable knowledge from MBSE models for PMs, schedulers, and FMers
  - Simulation, analysis, big data analytics harnessed to and driven by MBSE authoritative source of truth
- Model Based Acquisition update to Unified Architecture Framework (UAF)
  - Integrate defense acquisition and sustainment concepts into UAF
  - Work with DAF orgs (AFLCMC), Object Modeling Group (OMG)
- Early SysML v2 Integration and Adoption
  - Support development of FOSS SysML v2 applications
  - Identify and support early adopters and pathfinders of SysML v2 across USAF and USSF
- Alternatives to current leading MBSE tools
  - Survey existing landscape of MBSE modeling tools, including COTS and FOSS
  - Identify and support orgs willing to investigate alternative solutions