International Council on Systems Engineering
A better world through a systems approach

Taming the Beast: Best Practices of Extending SysML V2

Dr. Aurelijus Morkevicius

Gintare Krisciuniene

Hello.

Aurelijus Morkevicius

CATIA Systems – MBSE Consulting Director

- PhD, MS, and BS in Software Systems Engineering
- 20 years in Software and Systems Engineering
- UAF co-chairman in OMG, member of INCOSE and NATO ACaT
- Chair of Enterprise Systems Engineering WG in INCOSE
- Originator of the MagicGrid Framework
- CSEP, OCSMP, OCEB, OCUP certified professional

Gintare Krisciuniene

CATIA Systems Modeling Application Manager

Today's Agenda

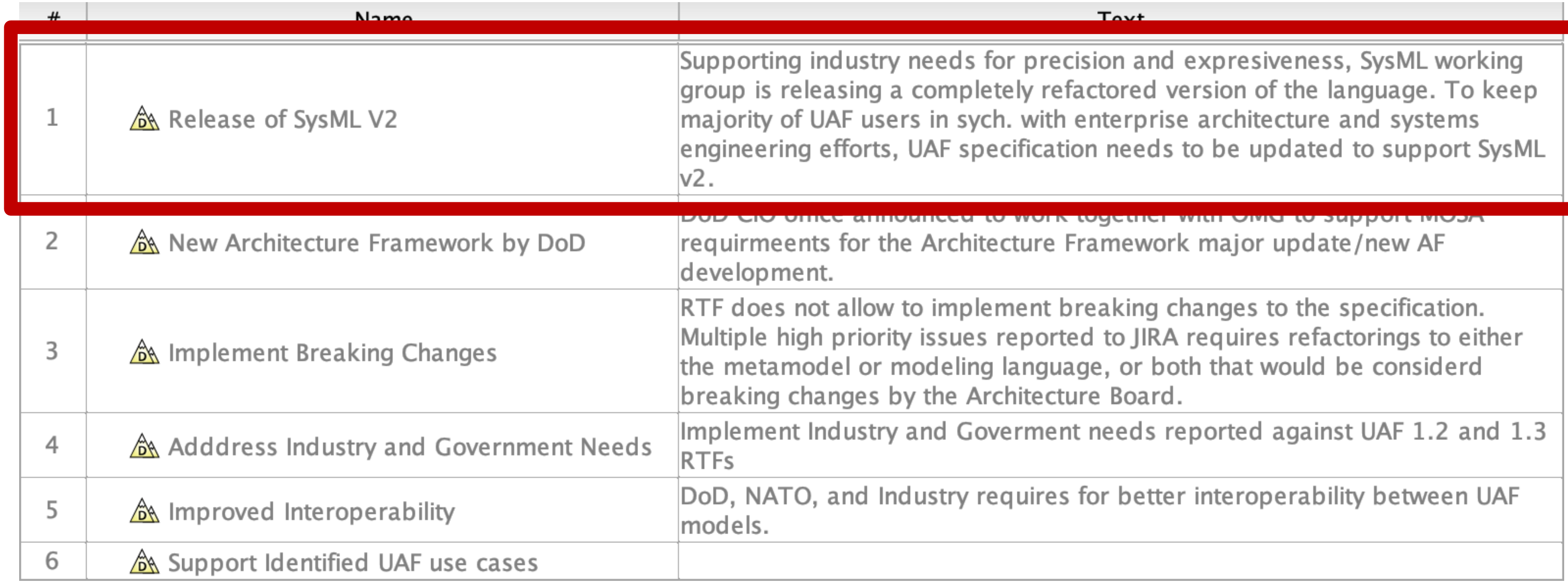
- Why This Topic?
- Standard Model Libraries for UAF V2
- Limitations and Best Practices of Extending SysML V2
- Summary

Introduction

Why this topic?

- Developing Standard SysML V2 Extensions to Unified Architecture Framework (UAF) V2 – a first real, large-scope test of SysML V2 extendibility.
- UAF is a Standard…
  - To develop architectural descriptions of enterprises in commercial industries, federal governments and military organizations
    - Engineered systems include products, services and enterprises (INCOSE-TP-2020-002-06 | 22 July 2019).
    - “Enterprise” is intended to mean a large undertaking, especially one of large scope, complication and risk – “a complex web of interactions distributed across geography and time” (Rebovitch & White, 2011).
  - developed by Object Management Group (OMG) with the leadership from Dassault Systemes, Lockheed Martin and INCOSE
  - international ISO standard ISO/IEC 19540:1 and ISO/IEC 19540:2
  - current version of UAF specification is 1.2 https://www.omg.org/spec/UAF/1.2/About-UAF
- Majority (over 90%) of SysML V1 users have custom profiles, diagrams, and DSL customizations.
- What to consider when setting-up transformation strategy for V2?

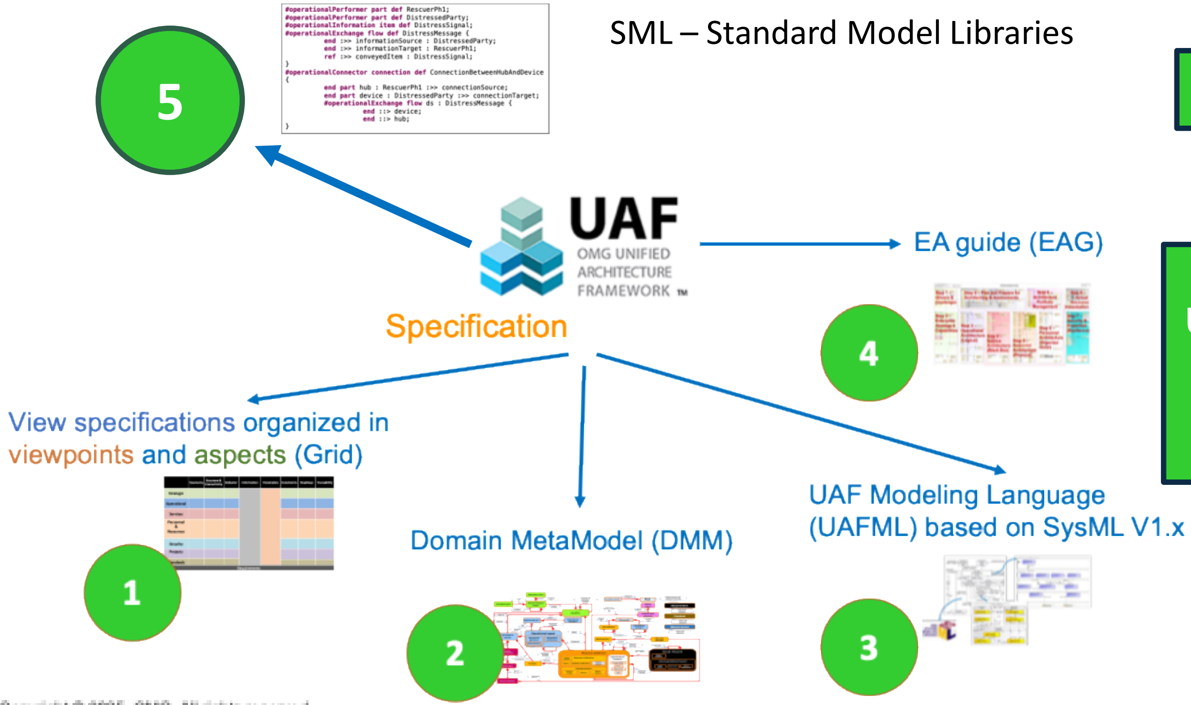
UAF V2 drivers



#	Name	Text
1	Release of SysML V2	Supporting industry needs for precision and expresiveness, SysML working group is releasing a completely refactored version of the language. To keep majority of UAF users in sych. with enterprise architecture and systems engineering efforts, UAF specification needs to be updated to support SysML v2.
2	New Architecture Framework by DoD	DoD CIO office announced to work together with OMG to support MOSA requirmeents for the Architecture Framework major update/new AF development.
3	Implement Breaking Changes	RTF does not allow to implement breaking changes to the specification. Multiple high priority issues reported to JIRA requires refactorings to either the metamodel or modeling language, or both that would be considerd breaking changes by the Architecture Board.
4	Adddress Industry and Government Needs	Implement Industry and Goverment needs reported against UAF 1.2 and 1.3 RTFs
5	Improved Interoperability	DoD, NATO, and Industry requires for better interoperability between UAF models.
6	Support Identified UAF use cases	

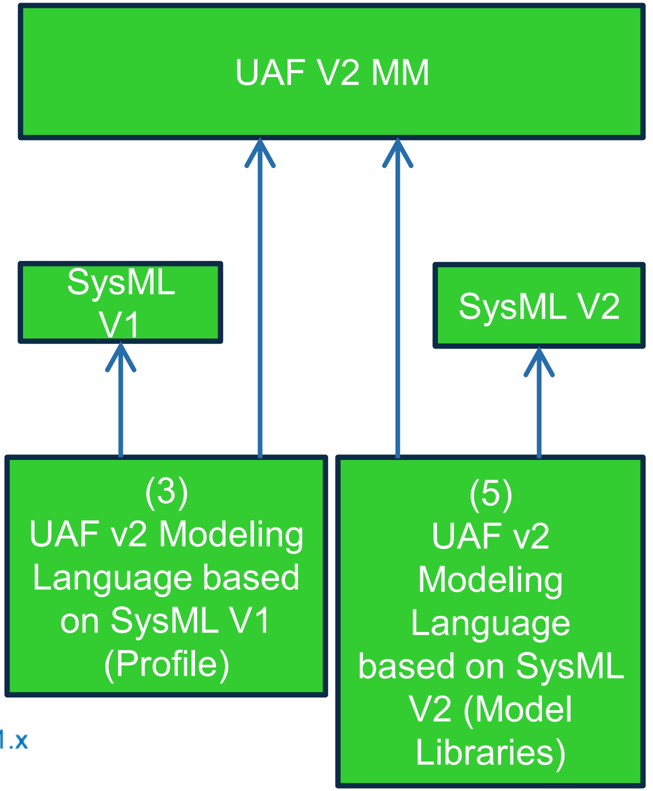
UAF specification at a glance

UAF Modeling Language (UAFSML) based on SysML V2



```
#operationalPerformer part def RescuerPh1;
#operationalPerformer part def DistressedParty;
#operationalInformation item def DistressSignal;
#operationalExchange flow def DistressMessage {
        end :>> informationSource : DistressedParty;
        end :>> informationTarget : RescuerPh1;
        ref :>> conveyedItem : DistressSignal;
}
#operationalConnector connection def ConnectionBetweenHubAndDevice
{
        end part hub : RescuerPh1 :>> connectionSource;
        end part device : DistressedParty :>> connectionTarget;
        #operationalExchange flow ds : DistressMessage {
                end ::> device;
                end ::> hub;
}
```

5
SML – Standard Model Libraries
UAF
OMG UNIFIED ARCHITECTURE FRAMEWORK ™
Specification
EA guide (EAG)
4
View specifications organized in viewpoints and aspects (Grid)
1
Domain MetaModel (DMM)
2
UAF Modeling Language (UAFML) based on SysML V1.x
3


UAF V2 MM
SysML V1
SysML V2
(3) UAF v2 Modeling Language based on SysML V1 (Profile)
(5) UAF v2 Modeling Language based on SysML V2 (Model Libraries)

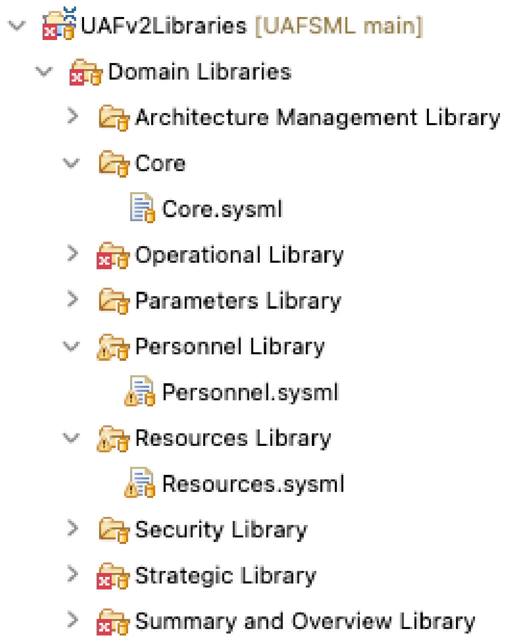
UAF V2
Standard Model Libraries (SML)

Why SysML V2?

- Increase adoption and effectiveness of MBSE with SysML by enhancing…
  - 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
  - Comparing SysML v2 with SysML v1:
    - Simpler to learn and use
    - More precise
    - More expressive
    - More extensible
    - More interoperable

Organization (Modular Approach)

- Core Library, for cross cutting concepts
- Viewpoint Libraries for different viewpoints of the framework
- Interrelated together
- Specialized Libraries, e.g. Model-based Acquisition, Mission Architecture, Resilience, Safety, etc.
- Metadata



UAFv2Libraries [UAFSML main]
- Domain Libraries
  - Architecture Management Library
  - Core
    - Core.sysml
  - Operational Library
  - Parameters Library
  - Personnel Library
    - Personnel.sysml
  - Resources Library
    - Resources.sysml
  - Security Library
  - Strategic Library
  - Summary and Overview Library

UAF	Motivation Mv	Taxonomy Tx	Structure Sr	Connectivity Cn	Processes Pr	States St	Sequences Sq	Information[c] If	Parameters[d] Pm	Constraints Ct	Roadmap Rm	Traceability Tr
Architecture Management[a] Am	Architecture Principles Am-Mv	Architecture Extensions Am-Tx[e]	Architecture Views Am-Sr	Architecture References Am-Cn	Architecture Development Method Am-Pr	Architecture Status Am-St		Dictionary Am-If	Architecture Parameters Am-Pm	Architecture Constraints Am-Ct	Architecture Roadmap Am-Rm	Architecture Traceability Am-Tr
Summary & Overview Sm-Ov												
Strategic St	Strategic Motivation St-Mv	Strategic Taxonomy St-Tx	Strategic Structure St-Sr	Strategic Connectivity St-Cn	Strategic Processes St-Pr	Strategic States St-St		Strategic Information St-If	Environment En-Pm-E and Measurements Me-Pm-M and Risks Rk-Pm-R	Strategic Constraints St-Ct	Strategic Deployment, St-Rm-D Strategic Phasing St-Rm-P	Strategic Traceability St-Tr
Operational Op	Requirements Rq-Mv	Operational Taxonomy Op-Tx	Operational Structure Op-Sr	Operational Connectivity Op-Cn	Operational Processes Op-Pr	Operational States Op-St	Operational Sequences Op-Sq	Operational Information Op-If		Operational Constraints Op-Ct		Operational Traceability Op-Tr
Services Sv		Services Taxonomy Sv-Tx	Services Structure Sv-Sr	Services Connectivity Sv-Cn	Services Processes Sv-Pr	Services States Sv-St	Services Sequences Sv-Sq			Services Constraints Sv-Ct	Services Roadmap Sv-Rm	Services Traceability Sv-Tr
Personnel Ps		Personnel Taxonomy Ps-Tx	Personnel Structure Ps-Sr	Personnel Connectivity Ps-Cn	Personnel Processes Ps-Pr	Personnel States Ps-St	Personnel Sequences Ps-Sq	Resources Information Rs-If		Competence, Drivers, Performance Ps-Ct	Personnel Availability Ps-Rm-A Personnel Evolution PS-Rm-E Personnel Forecast Ps-Rm-F	Personnel Traceability Ps-Tr
Resources Rs		Resources Taxonomy Rs-Tx	Resources Structure Rs-Sr	Resources Connectivity Rs-Cn	Resources Processes Rs-Pr	Resources States Rs-St	Resources Sequences Rs-Sq			Resources Constraints Rs-Ct	Resources evolution Rs-Rm-E Resources forecast Rs-Rm-F	Resources Traceability Rs-Tr
Security Sc	Security Controls Sc-Mv	Security Taxonomy Sc-Tx	Security Structure Sc-Sr	Security Connectivity Sc-Cn	Security Processes Sc-Pr					Security Constraints Sc-Ct		Security Traceability Sc-Tr
Projects Pj		Projects Taxonomy Pj-Tx	Projects Structure Pj-Sr	Projects Connectivity Pj-Cn	Projects Processes Pj-Pr						Projects Roadmap Pj-Rm	Projects Traceability Pj-Tr
Standards Sd		Standards Taxonomy Sd-Tx	Standards Structure Sd-Sr								Standards Roadmap Sd-Rm	Standards Traceability Sd-Tr
Actual Resources Ar			Actual Resources Structure, Ar-Sr	Actual Resources Connectivity, Ar-Cn	Simulation[b]					Parametric Execution/ Evaluation[b]		

Views and Viewpoints: Grid as a Model!

View Specification

```
viewpoint def <'Op-Sr'> 'Operational Structure' {
    doc /* Shows composition and aggregation hierarchies of Operational Agents. */
    ref #aspect :>> aspects : Sr [1];
    concern def OperationalStructureConcern {
       doc /* Operational structure used to support a capability(ies). */
       stakeholder enterpriseArchitects : Stakeholders::'Enterprise Architect' [0..*];
       stakeholder systemsEngineers : Stakeholders::'Systems Engineer' [0..*];
       stakeholder businessArchitects : Stakeholders::'Business Architect' [0..*];
    }
    frame concern operationalStructureConcern : OperationalStructureConcern;
}
```

View

```
view def 'Operational Structure View' {
    viewpoint :>> Operational::operationalStructure;
    filter istype OperationalMetadata::operationalConfiguration or istype OperationalMetadata::operationalPerformer;
    render asTreeDiagram;
}
```

Viewpoint

```
viewpoint def <Op> Operational :> Viewpoint {
    doc Description
      /* Illustrates the Logical Architecture of the enterprise. Describes the requirements,
       * operational behavior, structure, and exchanges required to support (exhibit)
       * capabilities. Defines all operational elements in an implementation/solution
       * independent manner. */
    concern def 'Defined Architecture' {
       stakeholder enterpriseArchitects : Stakeholders::'Enterprise Architect' [0..*];
    }
    frame concern definedArchitecture : 'Defined Architecture';
    viewpoint operationalTaxonomy : 'Operational Taxonomy' :> 'view specifications';
   }
```

User Model

```
view 'Operational Structure Diagram' : ViewSpecifications::'Operational Structure View' {
   expose OperationalUserModel::rescueContext::**;
}
```

CATIA

Operational Library

```
public import CoreLibrary::*;

abstract part def OperationalAgent :> Desirer, CapableElement, OperationalAsset, SubjectOfOperationalRule, OperationalConnectableElement {
                doc /*
                                * An abstract type grouping OperationalArchitecture and OperationalPerformer.
                */
                part operationalParts[*] : OperationalAgent :> operationalFeatures :>> subparts;
                port operationalPorts[*] : OperationalPort :> operationalFeatures :>> ownedPorts;
                abstract occurrence operationalFeatures[*] : OperationalConnectableElement;
                action :>> ownedActions = (operationalActivities, operationalStates);
                action operationalActivities : OperationalActivity[*] :> ownedActions;
                perform action performedOperationalActivities : OperationalActivity[*];
                state operationalStates : OperationalStateDescription [*] :>> ownedStates;
                exhibit state performedOperationalStates[*] : OperationalStateDescription;
                connection operationalConnectors[*] : OperationalConnector connect operationalFeatures to operationalFeatures;
      include use case performedOperationalUseCases : OperationalUseCase[0..*];
}

part def OperationalPerformer :> OperationalAgent{
                doc /*
                                * A logical agent that IsCapableToPerform OperationalActivities which produce, consume, and process Resources.
                */
}

action def OperationalActivity :> Process, SubjectOfOperationalRule, OperationalConnectableElement, OperationalUseCaseScenario {
                doc /*
                                * An Activity that captures a logical process, specified independently of how the process is carried out.
                */
                out item outPin [*] : OperationalExchangeItem;
                in item inPin [*] : OperationalExchangeItem ;
                action operationalActivityActions[*] : OperationalActivity = subactions -> ControlFunctions::select {
                                in act; act istype OperationalActivity} ;
                succession flow operationalActivityEdges[*] : OperationalExchange from
                                operationalActivityActions.outPin to operationalActivityActions.inPin;
}
```

Operational Metadata

```
public import OperationalLibrary::*;

metadata def <operationalPerformer> OperationalPerformerMetadata :> SemanticMetadata{
     :> annotatedElement : SysML::PartDefinition;
     :> annotatedElement : SysML::PartUsage;
     :>> baseType =
     if (annotatedElement istype SysML::PartDefinition)?
       OperationalPerformer meta SysML::PartDefinition else
       OperationalAgent::operationalRoles meta SysML::PartUsage;

  }
  metadata def <operationalActivity> operationalActivityMetadata :> SemanticMetadata {
    :> annotatedElement : SysML::ActionDefinition;
     :> annotatedElement : SysML::ActionUsage;
     :>> baseType =
     if (annotatedElement istype SysML::ActionDefinition)?
       OperationalActivity meta SysML::ActionDefinition else
       OperationalActivity::operationalActivityActions meta SysML::ActionUsage;
  }
}
```

INCOSE

Operational Example

Textual Notation

Definitions

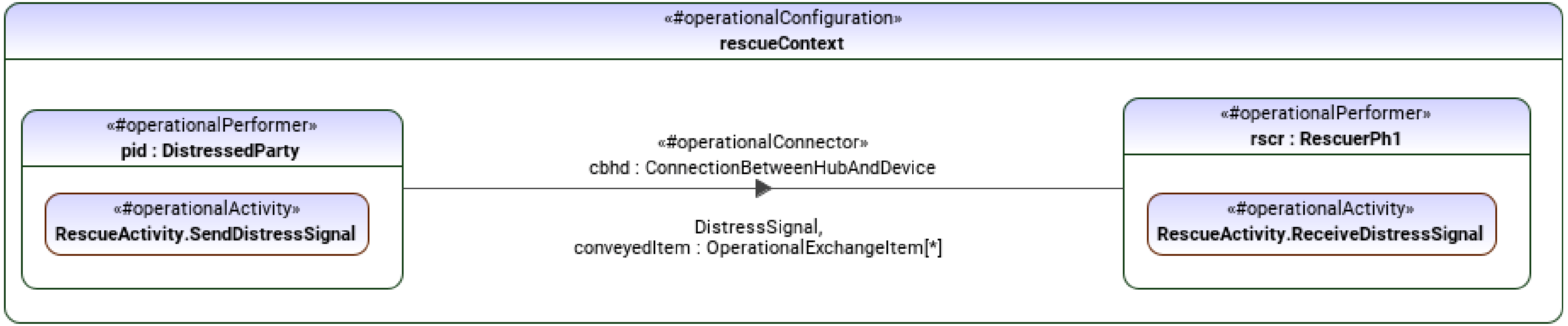
```
#operationalPerformer part def RescuerPh1;
#operationalPerformer part def DistressedParty;
#operationalInformation item def DistressSignal;
#operationalExchange flow def DistressMessage {
        end :>> informationSource : DistressedParty;
        end :>> informationTarget : RescuerPh1;
        ref :>> conveyedItem : DistressSignal;
}
#operationalConnector connection def ConnectionBetweenHubAndDevice {
        end part hub : RescuerPh1 :>> connectionSource;
        end part device : DistressedParty :>> connectionTarget;
        #operationalExchange flow ds : DistressMessage {
                end ::> device;
                end ::> hub;
}
```

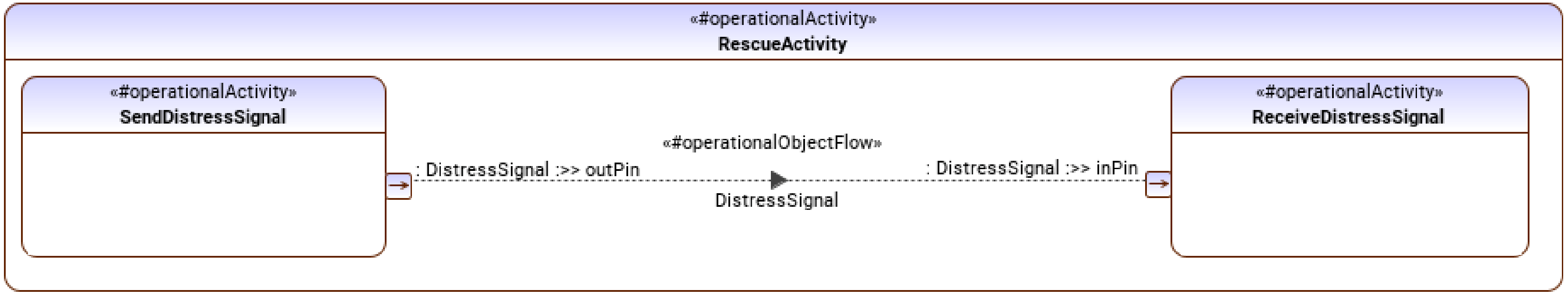
Usages

```
#operationalConfiguration part rescueContext {
        #operationalPerformer part pid: DistressedParty{
                #operationalActivity perform RescueActivity.SendDistressSignal;
        }
        #operationalPerformer part rscr: RescuerPh1 {
                #operationalActivity perform RescueActivity.ReceiveDistressSignal;
        }
        #operationalConnector connection cbhd : ConnectionBetweenHubAndDevice connect pid to rscr {
                perform RescueActivity.ObjectFlowWithDM;
        }
}
#operationalActivity action RescueActivity {
        #operationalActivity action SendDistressSignal {
                out : DistressSignal :>> outPin ;
        }
        #operationalActivity action ReceiveDistressSignal {
                in : DistressSignal :>> inPin ;
        }
        #operationalExchange succession flow ObjectFlowWithDM : DistressMessage from SendDistressSignal.outPin       to ReceiveDistressSignal.inPin;
}
```

Operational Example

Graphical Notation

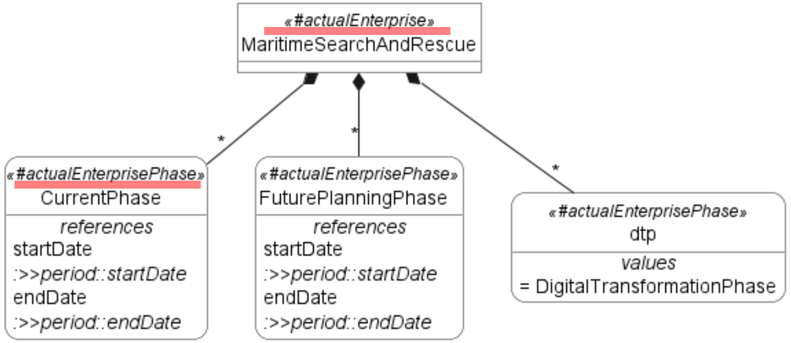


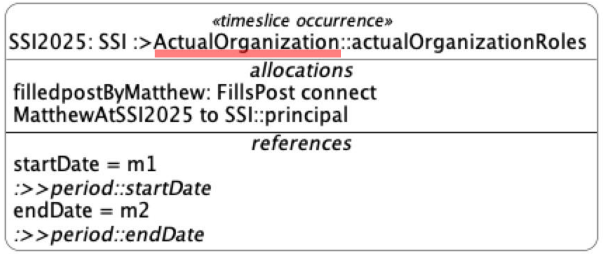


Limitations and Best practices of extending SysML V2

Metadatas

- Use of metadata (similar to stereotypes in SysML 1.x) is challenging
- Requires metadata definition for every concept which is already defined in the library.
- Using metadata keywords without SysML V2 standard keywords in the textual notation is not possible. You are still required to use SysML keywords e.g. *Item* when defining UAF concepts e.g. Capability.
  - Enterprise Architects and other non-SE roles will be forced to learn SysML concepts to use textual notation. Same for any other domain specific extension.




«timeslice occurrence»
SSI2025: SSI :>ActualOrganization::actualOrganizationRoles
allocations
filledpostByMatthew: FillsPost connect
MatthewAtSSI2025 to SSI::principal
references
startDate = m1
:>>period::startDate
endDate = m2
:>>period::endDate

- Easier to handle on the graphical notation

Fewer Concepts

- Fewer concepts. Every semantically rich element in SysML V2 can be modelled as a part, as a definition by adding the keyword *def,* and as an Individual adding keyword *individual*. For example, Operational Performer is a part and Operational Performer def is the definition.

```
#operationalPerformer part def RescuerPh1;
#operationalPerformer part def DistressedParty;
#operationalConfiguration part RescueContext{
    #operationalPerformer part rscr : RescuerPh1;
    #operationalPerformer part pid : DistressedParty;
}

metadata def <operationalPerformer> OperationalPerformerMetadata :> SemanticMetadata{
        :> annotatedElement : SysML::PartDefinition;
        :> annotatedElement : SysML::PartUsage;
        :>> baseType =
                if (annotatedElement istype SysML::PartDefinition)?
                OperationalPerformer meta SysML::PartDefinition else
                OperationalAgent::operationalRoles meta SysML::PartUsage;
}
```

Definitions

- SysML promise a language that allows end user to model usage scenarios without using definitions.
- With libraries, usages need to be typed by definitions in most of the cases. Why?
  - If no definition defined by the user, definition from library should be used.
  - Use of metadatas can help to avoid using definitions from the library.

```
requirement def RequiredServiceLevel :> PropertySet{
      doc /* ... */
      subject requiredServiceLevelSubject : Service;
   }
#requiredServiceLevel requirement purchasingServiceRequirements ~~: RequiredServiceLevel~~{
        subject p :> purchasing;
        require constraint {
           p.POCreationTime <= 72[h]
        }
        require constraint {
           p.orderFulfillmentTime <= 120[h]
        }
}
```

Different Layers of Abstractions

- There is no formalism in SysML V2 to model traces between different layers of abstraction.
- In SysML V2 a Dependency relationship is defined; however, it has no semantics.
- How to connect two decoupled levels of abstraction?
  - Connect using . notation without having a common context. Implications?

```
#operationalConfiguration part AirportTerminal {
        #operationalPerformer part gs : GroundSystem;
        #operationalPerformer part oac : Aircraft;
        #operationalConnector connection gsTooAC : PushBack connect gs to oac{
                ref provider :>> groundSystem :> gs;
                ref consumer :>> operatedAC :> oac;
}

#resourceConfiguration part Terminal{
        #resourceArtifact part atvs: AircraftTowingVehileSystem;
        #resourceArtifact part airbus320 : Airbus320;
}
#implements allocation a
        allocate Terminal.atvs to AirportTerminal.gs :> ResourceAgentImplementsOperationalAgent;
```

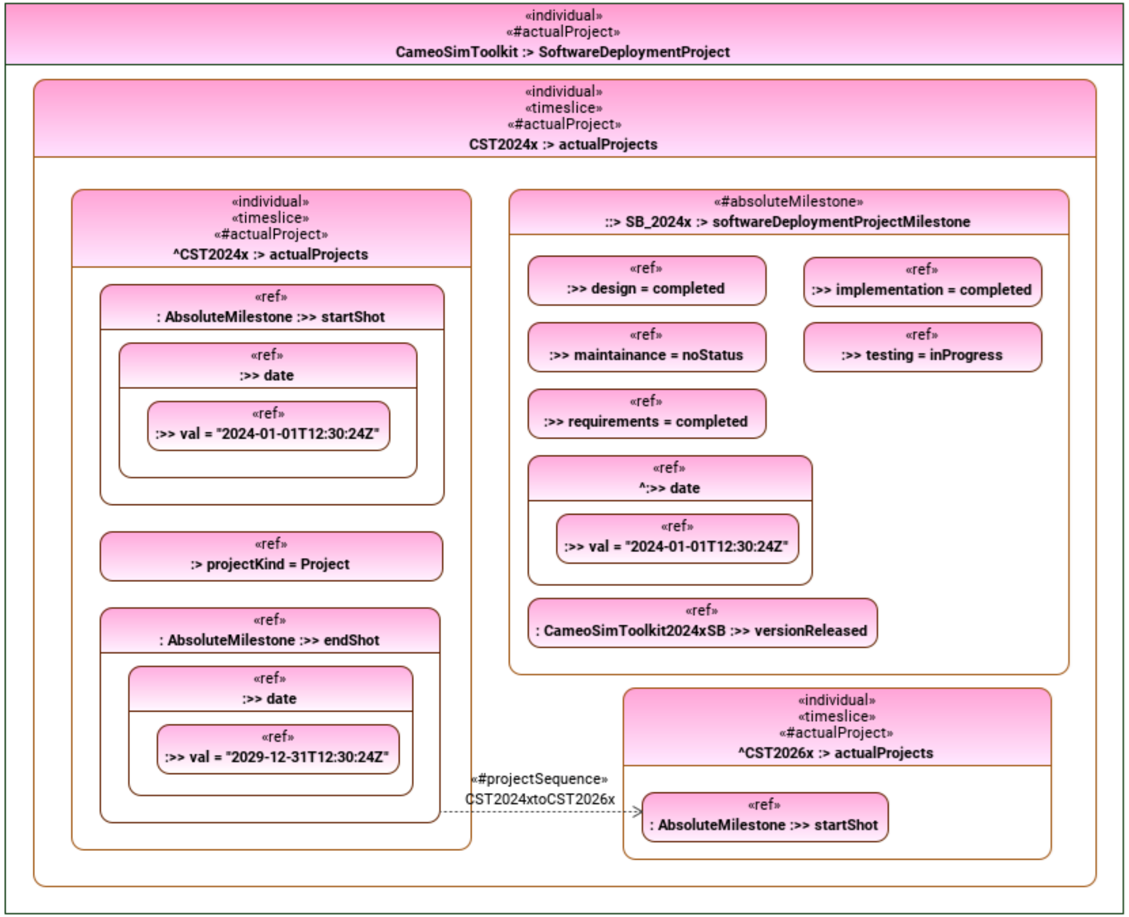
Time Slices and Snapshots

- The Time slice and Snapshot concepts introduce a very different method of handling individuals than in SysML V1.
- It perfectly fits UAF needs for enterprise and project planning.

```
individual #project part def CameoSimToolkit :> SoftwareDeploymentProject{
            timeslice #project CST2024x {
                        :>> startShot : AbsoluteMilestone {:>> date { :>> val = "2024-01-01T12:30:24Z";}} :>
            softwareDeploymentProjectMilestone;
                        :>> endShot : AbsoluteMilestone {:>> date {:>> val = "2029-12-31T12:30:24Z";}} :>
            softwareDeploymentProjectMilestone;
                        :> projectKind = projectKind::Project;
                        snapshot #absoluteMilestone SB_2024x :> softwareDeploymentProjectMilestone {
                                    :>> date = "2024-12-07T10:30:24Z";
                                    :>> versionReleased : CameoSimToolkit2024xSB;
                                    :>> requirements = SoftwareDevStatus::completed;
                                    :>> design = SoftwareDevStatus::completed;
                                    :>> implementation = SoftwareDevStatus::completed;
                                    :>> testing = SoftwareDevStatus::inProgress;
                                    :>> maintainance = SoftwareDevStatus::noStatus;
                                                                          }
                        }
```

Time Slices and Snapshots Example

Graphical Notation


«individual»
«#actualProject»
CameoSimToolkit :> SoftwareDeploymentProject
«individual»
«timeslice»
«#actualProject»
CST2024x :> actualProjects
«individual»
«timeslice»
«#actualProject»
^CST2024x :> actualProjects
«ref»
: AbsoluteMilestone :>> startShot
«ref»
:>> date
«ref»
:>> val = "2024-01-01T12:30:24Z"
«ref»
:> projectKind = Project
«ref»
: AbsoluteMilestone :>> endShot
«ref»
:>> date
«ref»
:>> val = "2029-12-31T12:30:24Z"
«#absoluteMilestone»
::> SB_2024x :> softwareDeploymentProjectMilestone
«ref»
:>> design = completed
«ref»
:>> implementation = completed
«ref»
:>> maintainance = noStatus
«ref»
:>> testing = inProgress
«ref»
:>> requirements = completed
«ref»
^:>> date
«ref»
:>> val = "2024-01-01T12:30:24Z"
«ref»
: CameoSimToolkit2024xSB :>> versionReleased
«#projectSequence»
CST2024xtoCST2026x
«individual»
«timeslice»
«#actualProject»
^CST2026x :> actualProjects
«ref»
: AbsoluteMilestone :>> startShot

Library Constraints

- Libraries are powerful tool to define domain specific extensions. However, there are some things to be aware of:
  - Library constraints are up to the tool vendor. Beware that instead of warning about incorrect relation end SysML V2 would compile and apply missing type to the incorrect end type **making the wrong model right!**

```
allocation def Presents :> PropertySet {
		end presentingElement : Driver :>> source ;
		end presentedElement : Challenge :>> target;
}
abstract item def MotivationalElement :> PropertySet{
		doc /*
			* An abstract kind of element in the model that provides the 				reason or reasons one has for acting or behaving in a particular way.
		*/
		ref item motivationalRoles [*] : MotivationalElement;
}
```

Best Practices of Using Metadatas

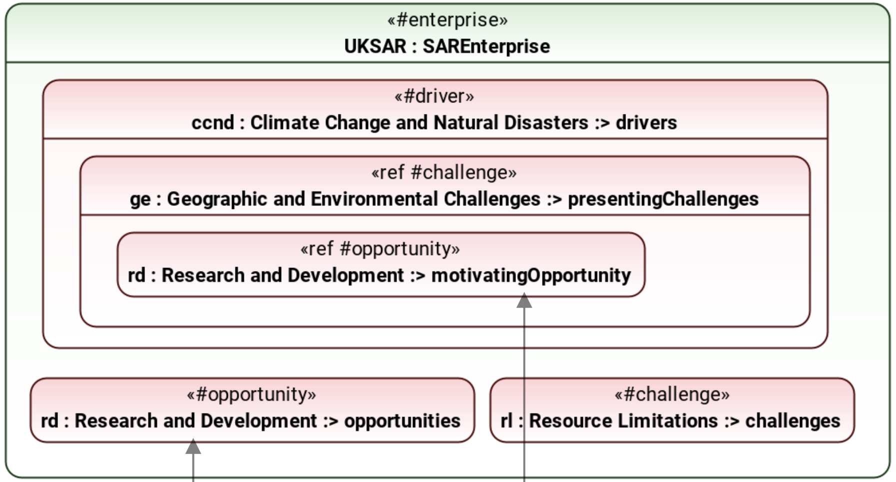
- You cannot avoid using Metadatas -> use it for your benefit.
  - Metedata constraints
  - Graphical annotation with keywords
  - Searching model
  - Annotations

```
metadata def <operationalPerformer> OperationalPerformerMetadata :> SemanticMetadata{
        :> annotatedElement : SysML::PartDefinition;
        :> annotatedElement : SysML::PartUsage;
        :>> baseType =
                if (annotatedElement istype SysML::PartDefinition)?
                OperationalPerformer meta SysML::PartDefinition else
                OperationalAgent::operationalRoles meta SysML::PartUsage;
}
```

Metadatas (2)

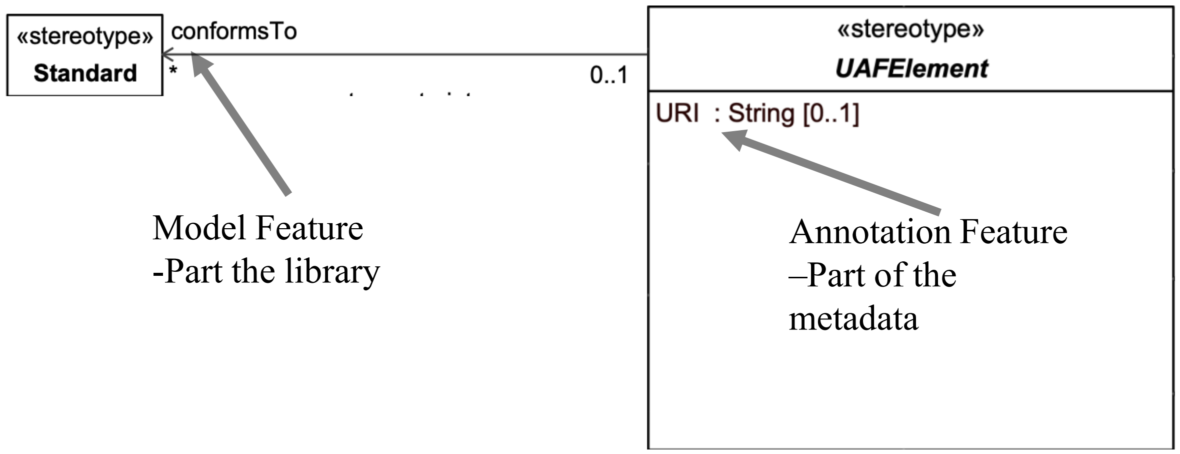
- Operations defined on the metatypes in the KerML specification, are not available(accessible) for the end user.
  - Of these the most important for the extensibility are KerML::Type::allSupertypes() , KerML::Type::specializes()
  - Cannot evaluate baseType expression:

```
metadata def <opportunity> OpportunityMetadata1 :> SemanticMetadata {
    :> annotatedElement : SysML::ItemDefinition;
    :> annotatedElement : SysML::ItemUsage;
    :>> baseType =
       if (annotatedElement istype SysML::ItemDefinition)
        ? Opportunity meta SysML::ItemDefinition
        else if (annotatedElement.owner as KerML::Feature).specializes.{in f : Expression; f(Initiative meta SysML::Definition)}
             ? Initiative::opportunities metaSysML::ItemUsage
               else if (annotatedElement.owner as KerML::Feature).specializes.{in f : Expression; f(Challenge meta SysML::Definition)}
                                                       ? Challenge::motivatingOpportunity meta SysML::ItemUsage
                  else opportunities metaSysML::ItemUsage;
 }
```


«#enterprise»
UKSAR : SAREnterprise
«#driver»
ccnd : Climate Change and Natural Disasters :> drivers
«ref #challenge»
ge : Geographic and Environmental Challenges :> presentingChallenges
«ref #opportunity»
rd : Research and Development :> motivatingOpportunity
«#opportunity»
rd : Research and Development :> opportunities
«#challenge»
rl : Resource Limitations :> challenges

Tag Definitions

- Tags are a part of the library
  - annotation features -> part of metadatas



Summary

Conclusions

- A few significant improvements, such as time bound individuals support and a powerful libraries mechanism to define extensions, promise UAF V2 to be more precise and expressive.
- Some outstanding issues need to be fixed to fully support our needs. We are working closely with SysML V2 to make these fixes priority to keep us on the schedule.
- Identified differences between V1 and V2 introduce some concerns, including how new ways of modeling will be perceived by the community and how tool vendors will be able to hide the increased complexity of the language.
- To ensure that all UAF tools handle some situations, like Operational Exchanges (ItemFlows) the same way, patterns must be defined as part of the UAF specification
- No migrators will handle domain specific extensions. Mapping of existing extensions in V1 to V2 is manual work. Perhaps AI could do it?.

Thank You!

?

aurelijus.morkevicius@3ds.com

INCOSE

35th Annual **INCOSE** international symposium

hybrid event

Ottawa, Canada
July 26 - 31, 2025