34th Annual INCOSE international symposium

hybrid event

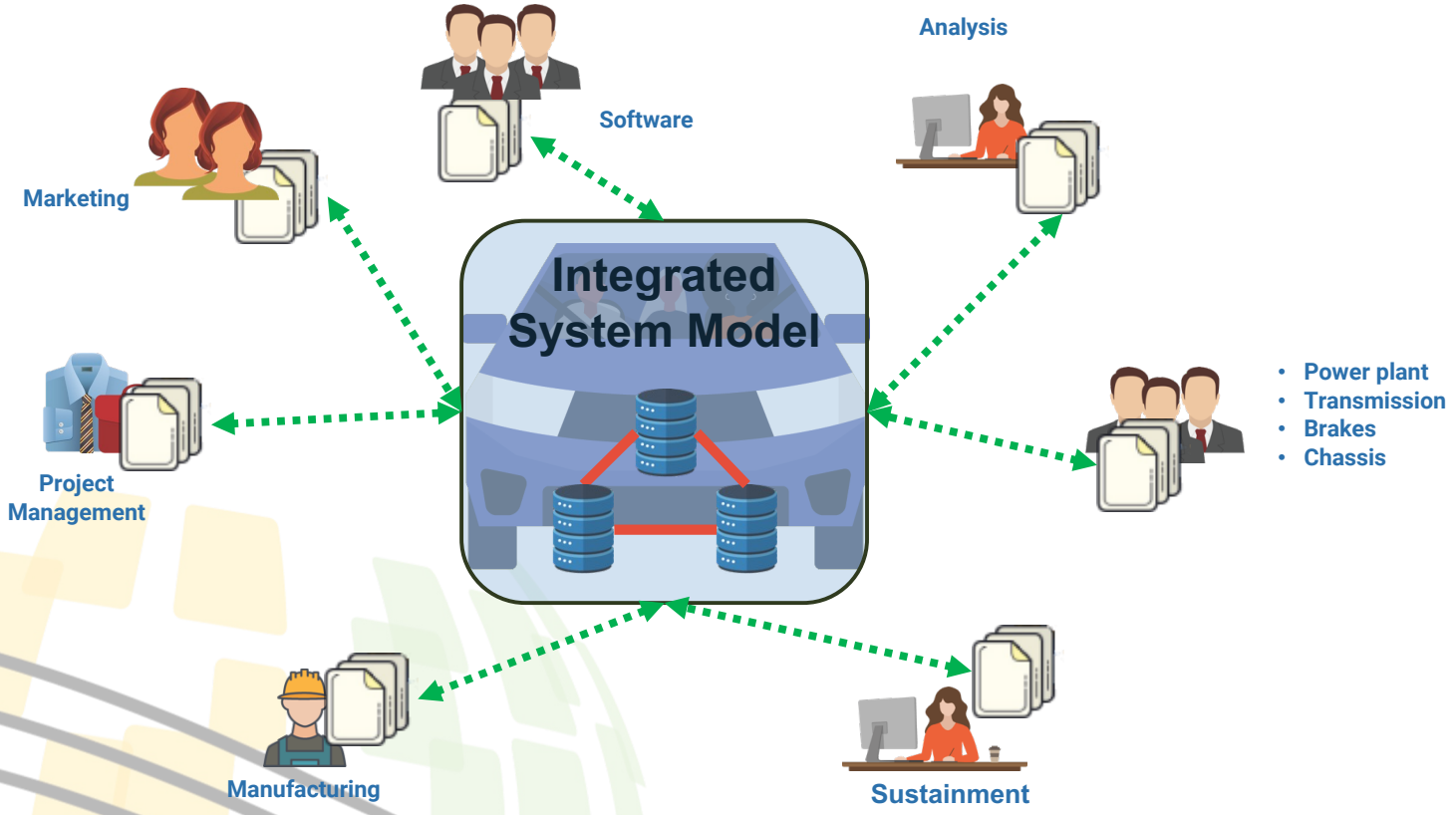
Dublin, Ireland
July 2 - 6, 2024

Empowering MBSE Through Metamodeling

Mr. Richard Wise
Senior Research Engineer
Georgia Tech Research Institute
richard.wise@gtri.gatech.edu

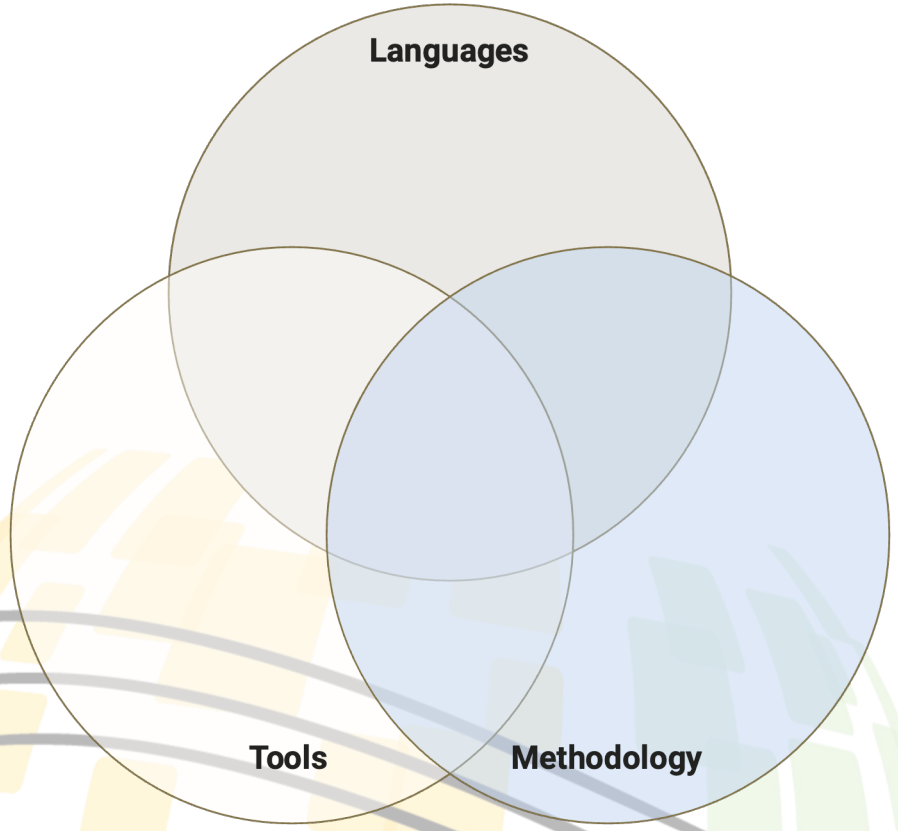
Mr. Rhett Zimmer
Systems Engineer
rhettazimmer@gmail.com

Integrated System Model - Ideal State



Shared understanding through coherent and consistent views

SysML is a MBSE Key Enabler



- SysML
  - Standardized
  - General purpose
  - Domain and methodology agnostic

SysML + Tools + Methods facilitates consistent and coherent representation of system architecture information

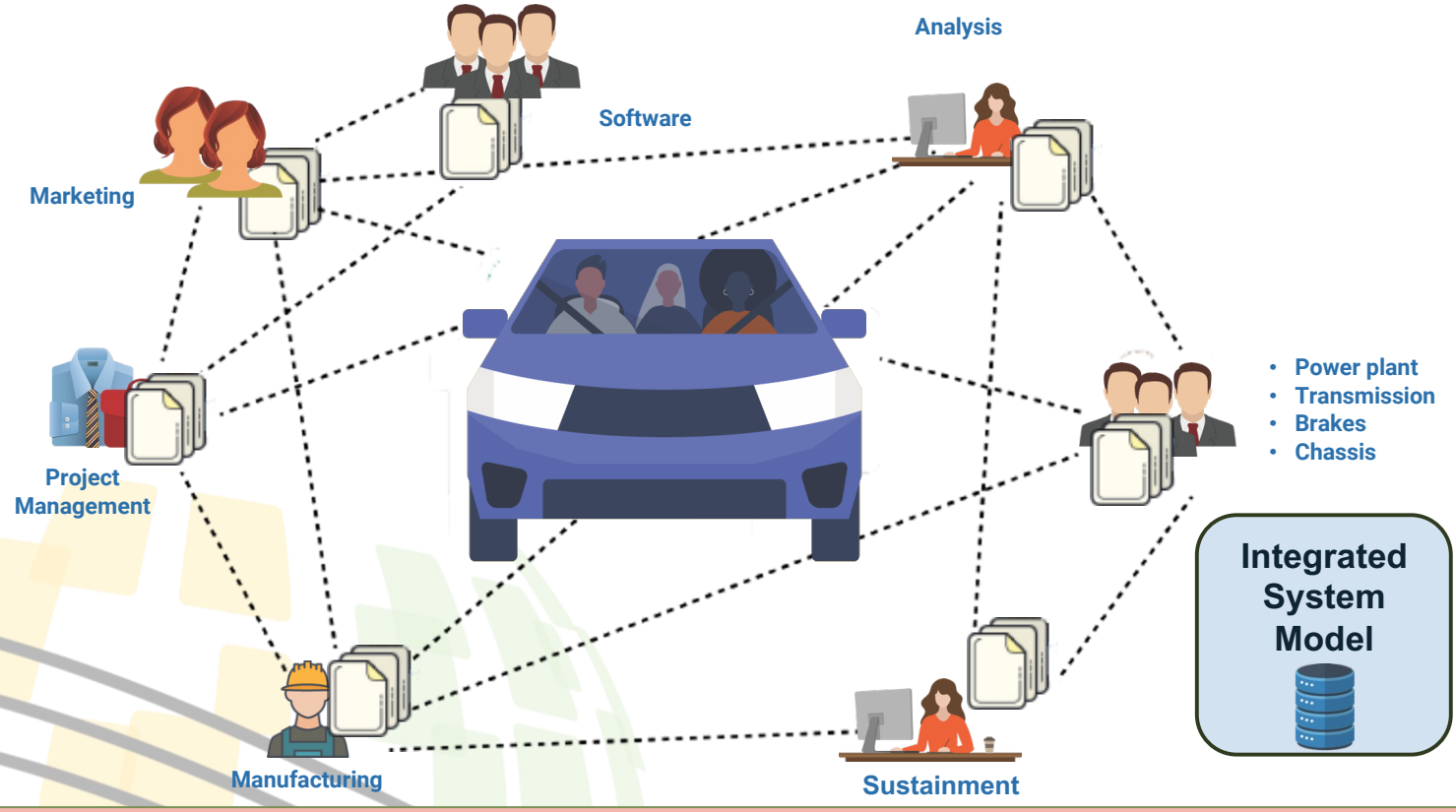
SysML alone leaves modelers with decisions

- How should elements be organized?
- Which types of elements should be presented on a diagram and consequently modeled?
- How is traceability established between architecture perspectives or architecting layers?
  - Behavior to Structure
  - Logical to Physical
- Which naming convention should be used?

SysML Modeler Standoff

(*The Office Three Way Gun Fight Blank Template - Imgflip*, n.d.)

Integrated System Model - Shelfware



Inconsistency and incoherence in modeling approaches leads to:
- **Reduced stakeholder confidence**
- **Impeded machine interrogation**

Common Approaches to Promote Coherency and Consistency in System Modeling

- Domain-specific profiles
- Reference Models
- Style Guides
- Metamodels

Domain-Specific Language (DSL) Profiles

- Extension of a modeling language for a specific domain
- Expresses domain-specific concepts, relationships, and rules

Standard Profiles

- Domain and methodology agnostic

Non-Standard Profiles

Open SE Cookbook: SysML Extensions

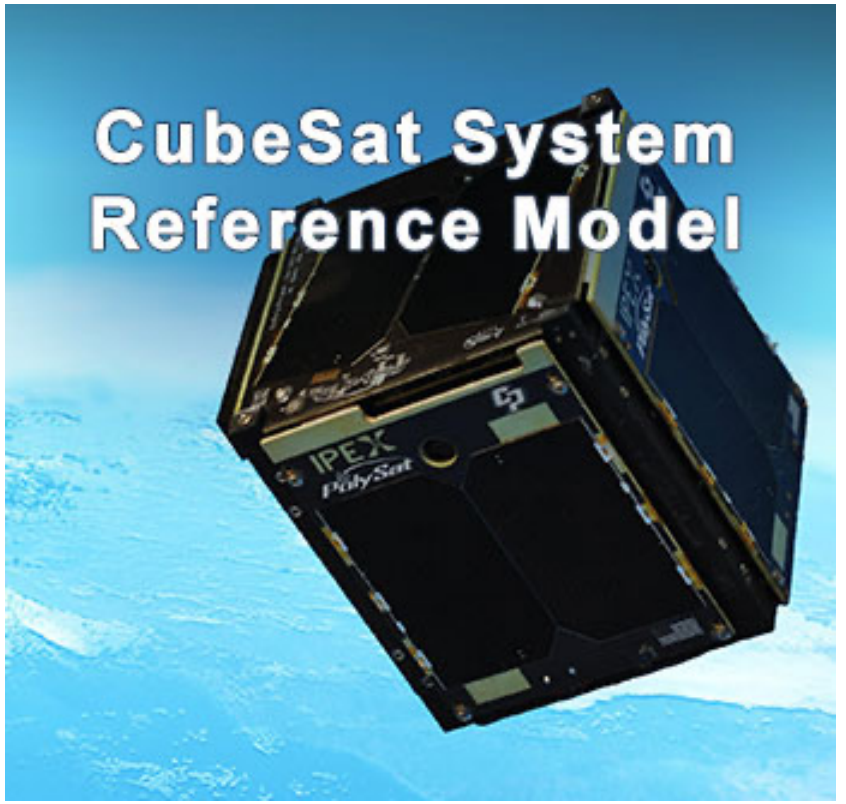
- Domain and/or methodology specific concepts and usage constraints

- **Burden is on the developer to maintain and train users**
- **Proliferation may result in overlapping and competing stereotypes**

(*OpenSE-Cookbook/Models at Master · Open-MBEE/OpenSE-Cookbook · GitHub*, n.d.)

Reference Models

- Representative example of what a referencing system model should look like
- Often ubiquitous, ‘thin’ slice example
- Reference model + style guide = reference implementation



(*CubeSat Reference Model* | *Object Management Group*, n.d.)

Only describes what could be, not what should or should not be

Style Guides

- Embodies modeling rules of an organization
  - Naming and diagramming conventions, element-to-element relationships
- Normally text-based but trending toward model-based

Conformance requires manually generated, modeling-tool specific validation rules
- **Non-standard domain-specific profile for context-specific semantics**
- **‘Broad-brush’ strokes on common stereotypes or metaclasses**

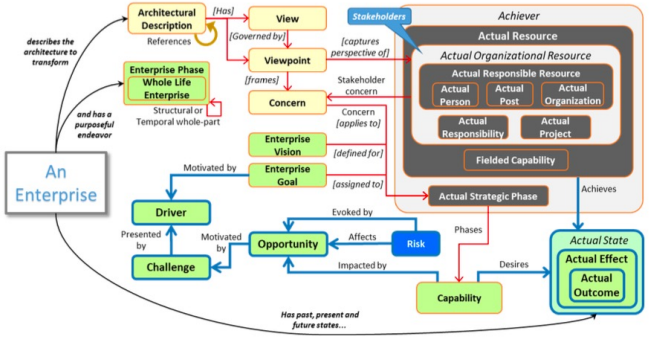
Metamodels

- Simply a model of a model
- *“A metamodel is a model that consists of statements about models. Hence, a metamodel is also a model, but its universe of discourse is a set of models... (Jeusfeld, 2009)”*
- Collection of concepts, attributes owned by the concepts, and relationships between the concepts
- Come in a variety of notations
  - Conceptual schema diagrams
  - UML-based abstract syntax diagrams

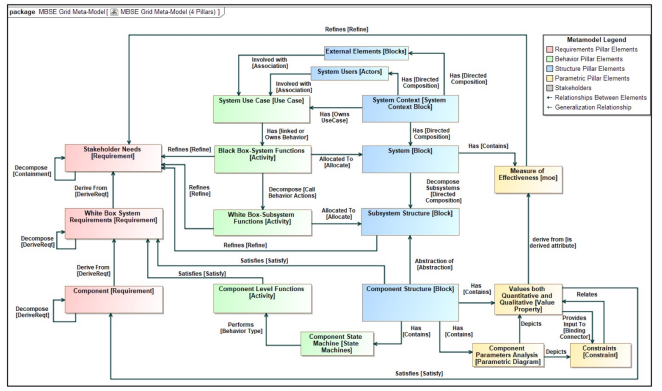
A model can then be considered a valid expression of or conforming to the metamodel if none of the statements in the metamodel are false with respect to the collection of model elements contained within the model *(Seidewitz, 2003)*.

Metamodels - Conceptual Schema Diagrams

- Highlight key concepts and relationships
- Useful visual guides for construction of actual models



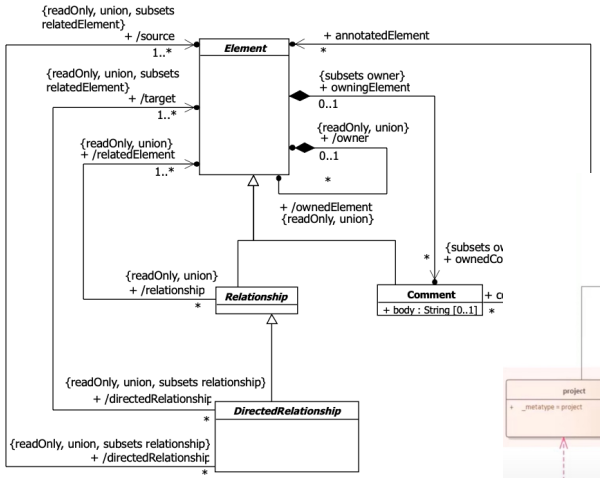
(Object Management Group, 2022)



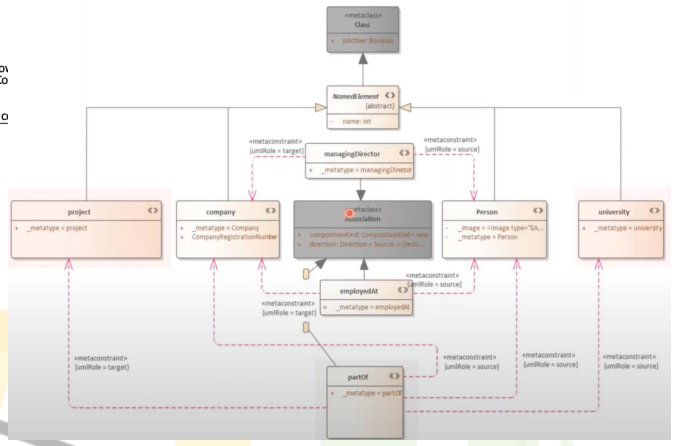
(Office of the NASA Chief Engineer, 2022)

- **Limited in expressiveness**
- **Conformance of actual models is impossible given informal notation**

Metamodels - UML based Abstract Syntax Diagrams



(Object Management Group, 2017)



(Horst, 2020)

- Expressed using a subset of UML
- Constraints declared via textual language, e.g. OCL
- Conformance checking of actual model to metamodel is possible due to formal notation
- Programmatic generation of constraints is progressing

- **UML Class notation is insufficient in specifying all semantics and constraints thus requiring manually, hand-written constraints**
- **Profiles are required for context-specific semantics**
- **Current programmatic constraint generation requires non-standard profiles**

Conclusion of Existing Approaches

- Each approach is valuable in facilitating coherency and consistency in system modeling

- **Insufficient at robust and automated conformance verification of actual models to the ideal model**
- **Domain-specific concepts and semantics rely on non-standard, domain-specific profiles**

The Best Fit (R^2) Metamodeling Approach

- Enables the creation of precise, machine interpretable metamodels with numerous applications

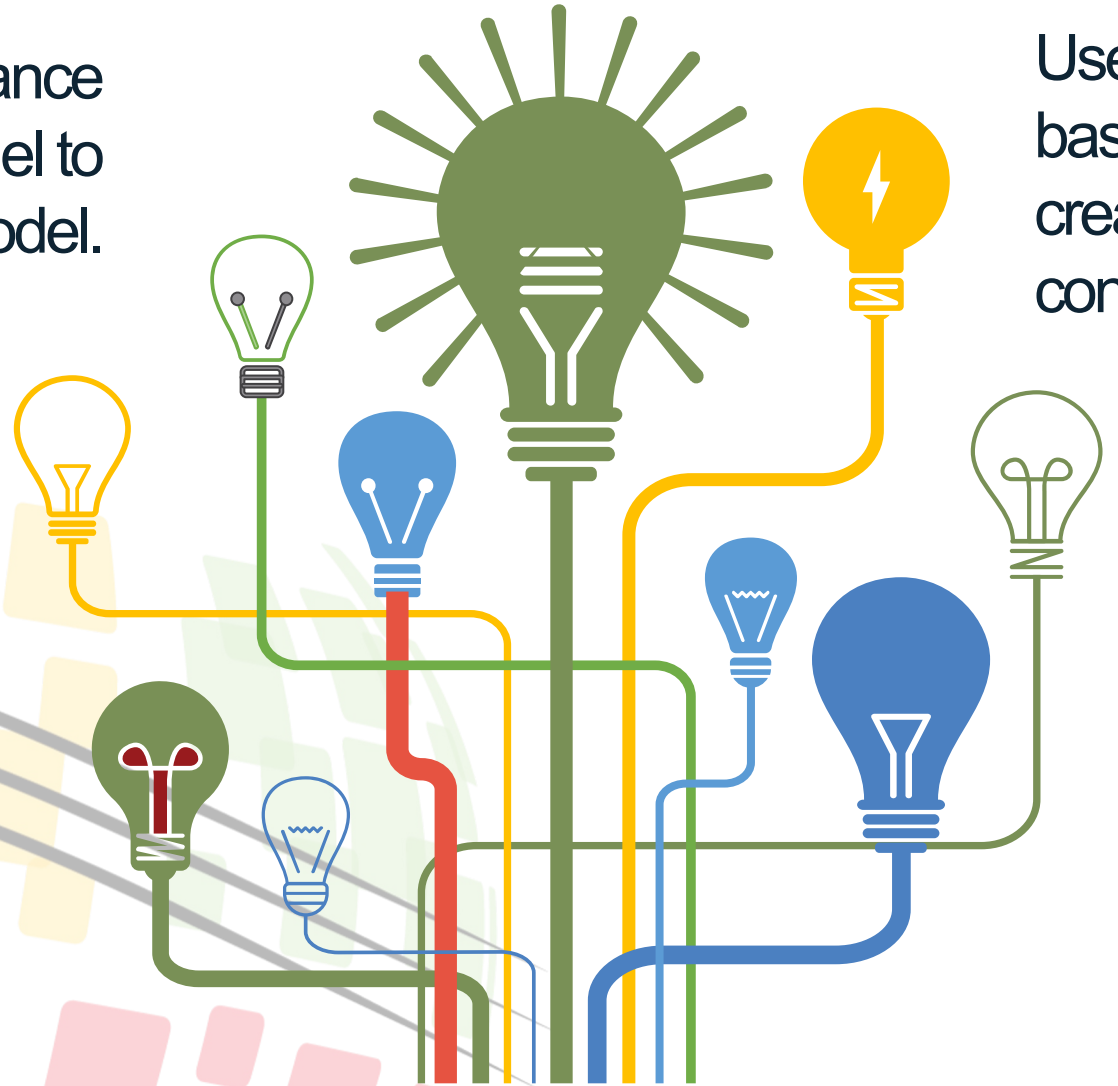
- Applications reduce overall system model development time and maximize system model utility

R^2 Metamodeling Approach Goals

Ensure conformance of the system model to a metamodel.

Specify domain-specific element types, characteristics, and relationships that shall exist.

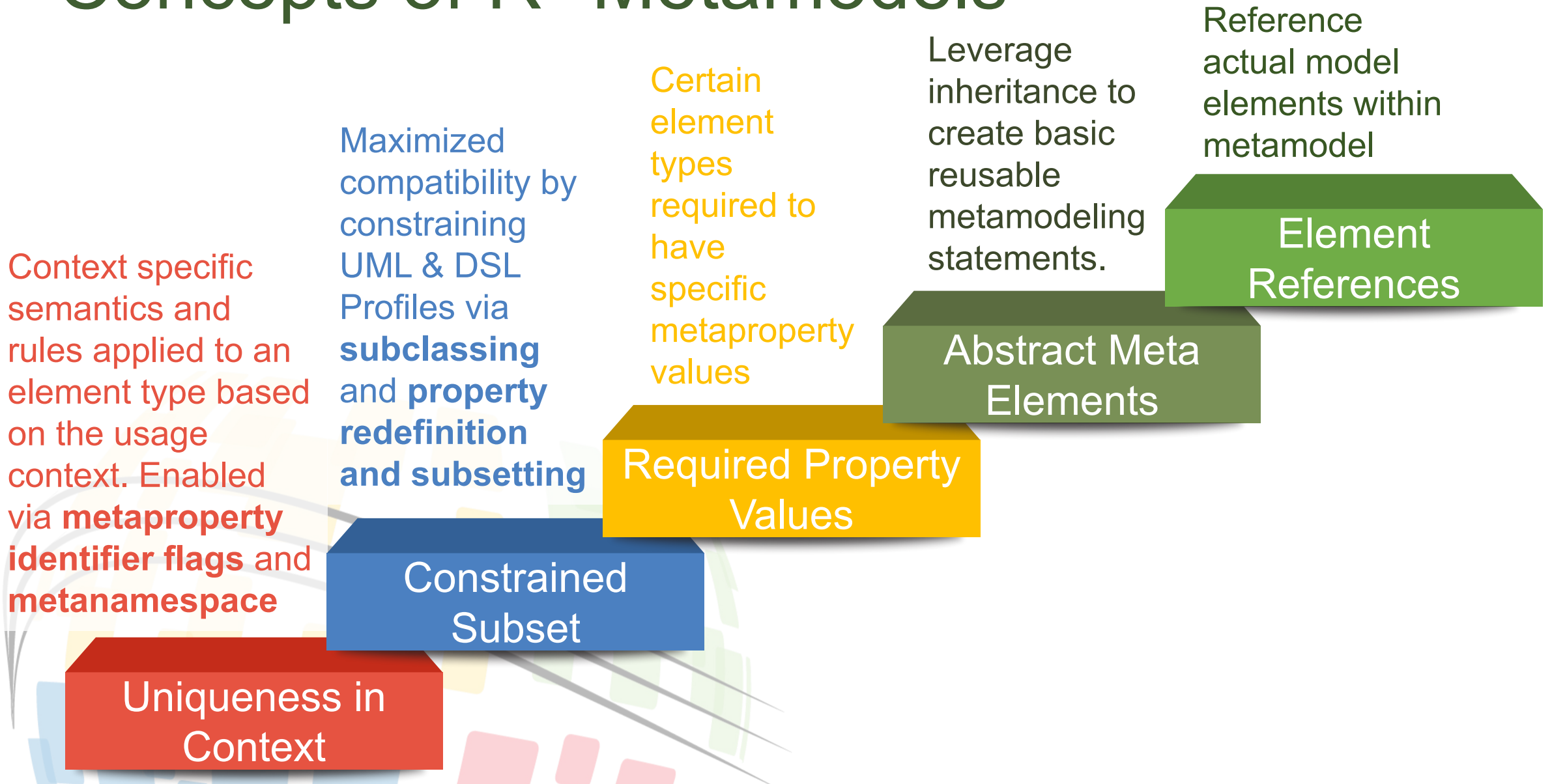
Enable machine interpretability.



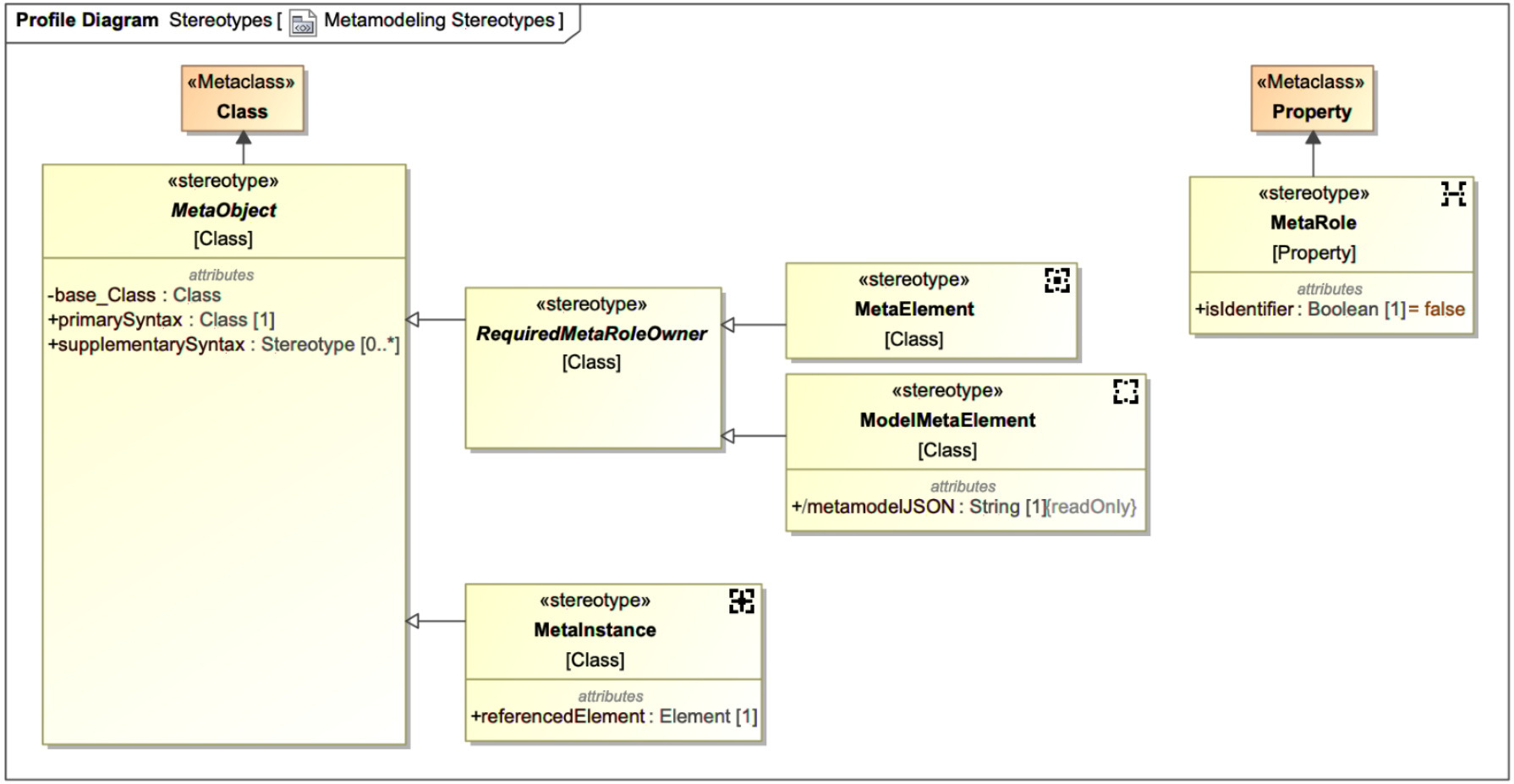
Use standardized UML-based languages to create domain-specific concepts and semantics.

Lower the barrier to creating metamodels by non-language architects.

Concepts of R^2 Metamodels

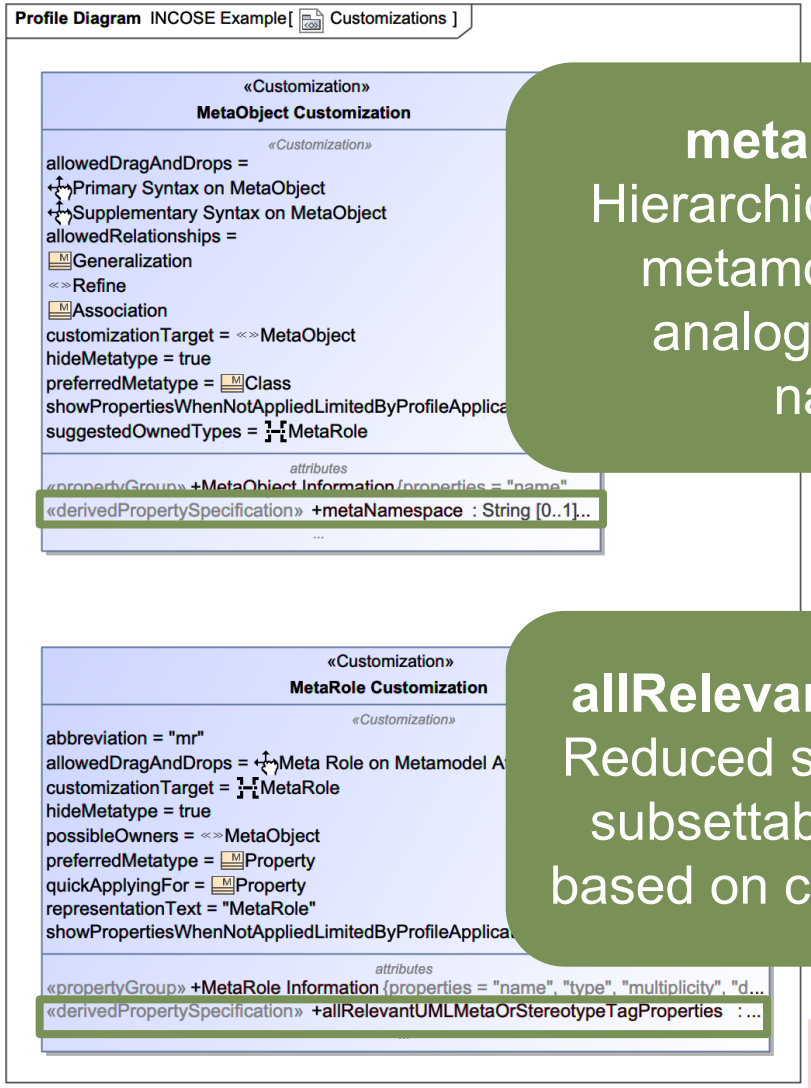


R² Metamodeling Profile



Lightweight UML profile to facilitate creation of R² metamodels

Metamodel Usability - Tool Customizations



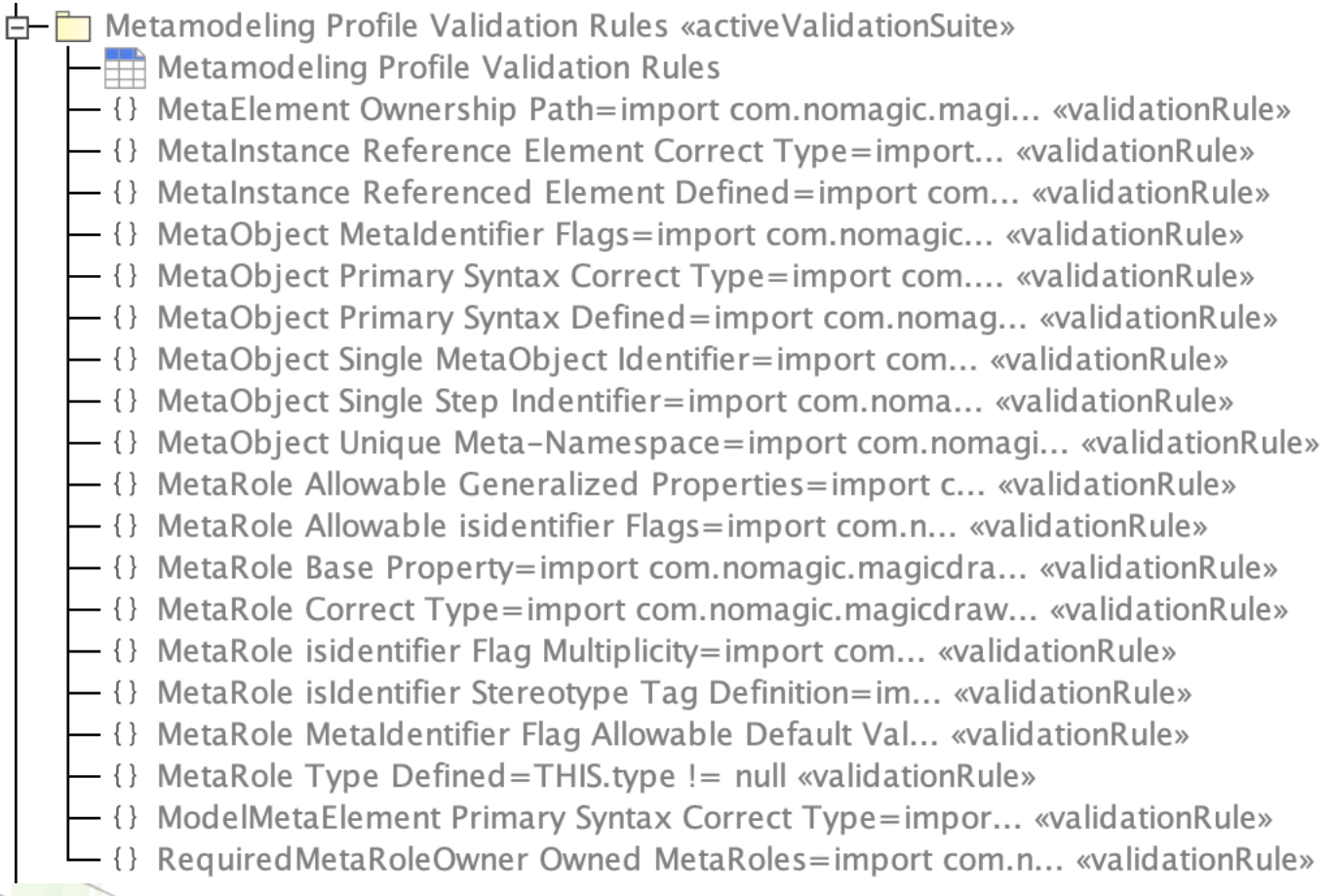
metaNamespace: Hierarchical list of pertinent metamodel information analogous to qualified namespace

allRelevantMetaProperties: Reduced set of redefinable or subsettable metaproperties based on contextual relevance

- Aids modelers in developing R^2 metamodels
- Derived properties developed in No Magic Cameo Systems Modeler™ 2021x R2 by Dassault Systemes

Metamodel Usability - Validation Rules

- 19 validation rules to ensure metamodel consistency and well-formedness
- Reduces ambiguity in metamodels and subsequent models



Metamodel Statements

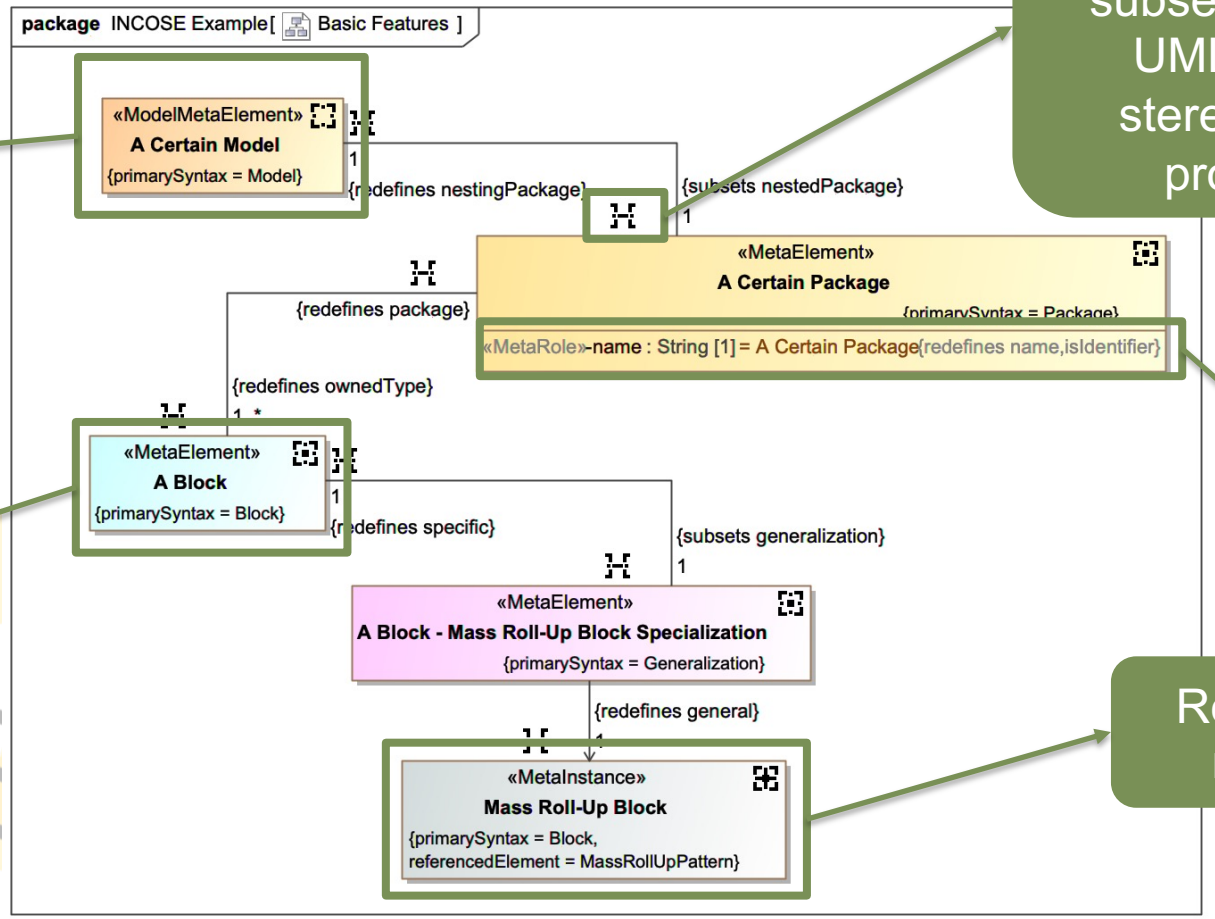
Specifies the root context for all MetaObjects in a metamodel

Definition of an element type in a metamodel; subclasses UML metaclass or stereotype

MetaRole: Redefines or subsets inherited UML meta or stereotype tag properties

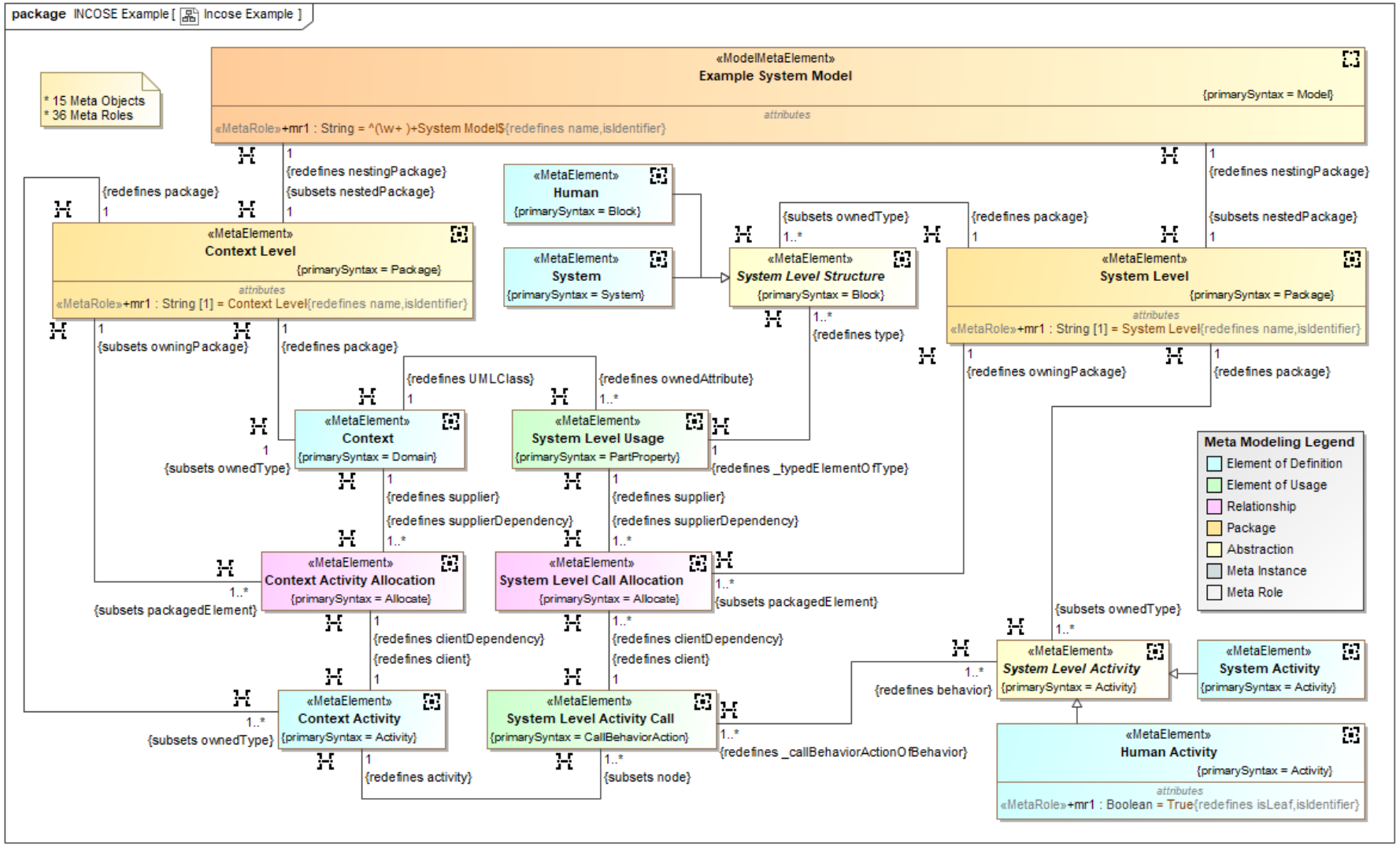
Meta-Identifier Flag: Value of the metaproperty contributes to element uniqueness

References actual model element

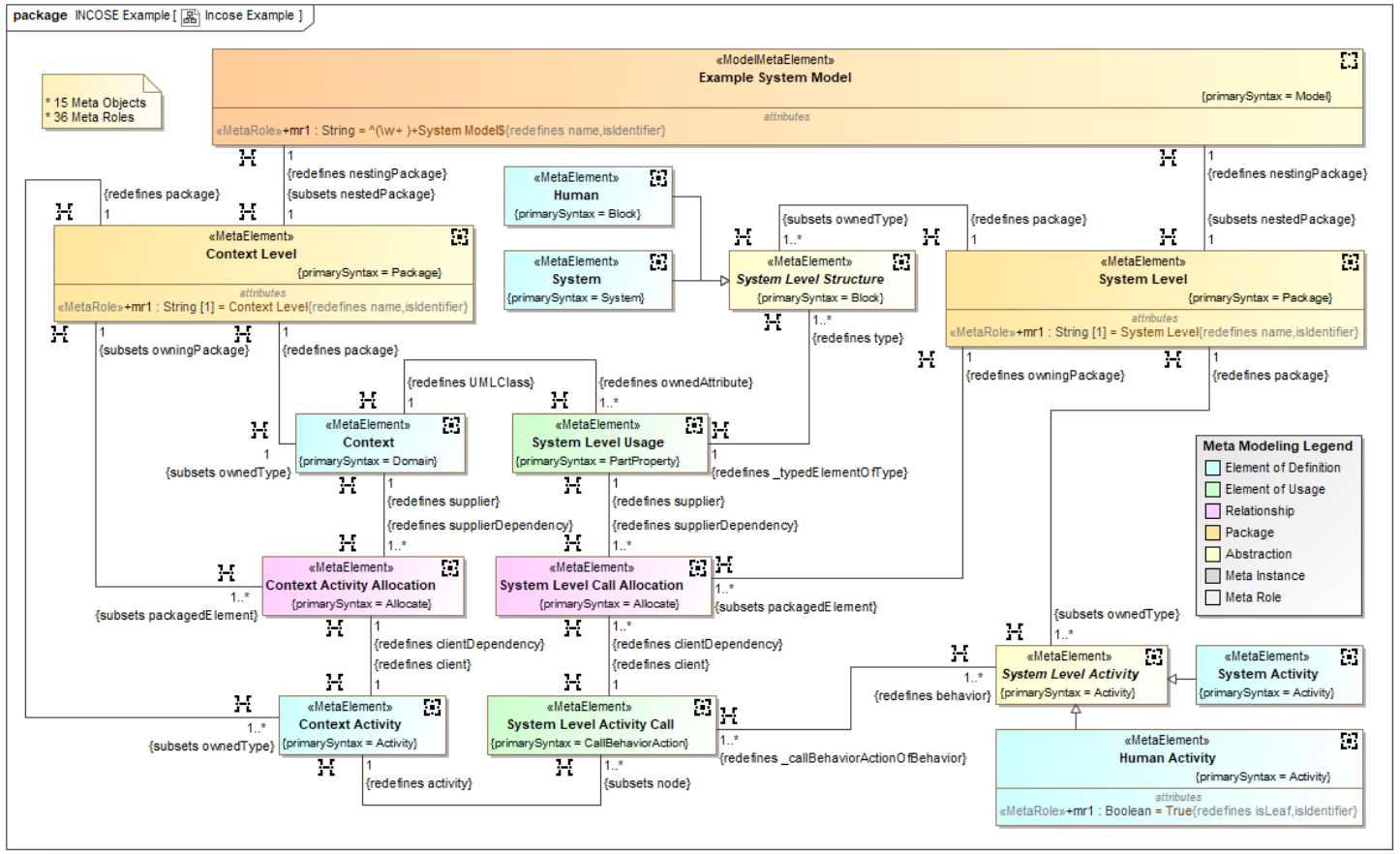


Example metamodel statement: All Blocks in A Certain Package in A Certain Model shall specialize the MassRollUpPattern block in the MD Customization for SysML package

A Simple Metamodel



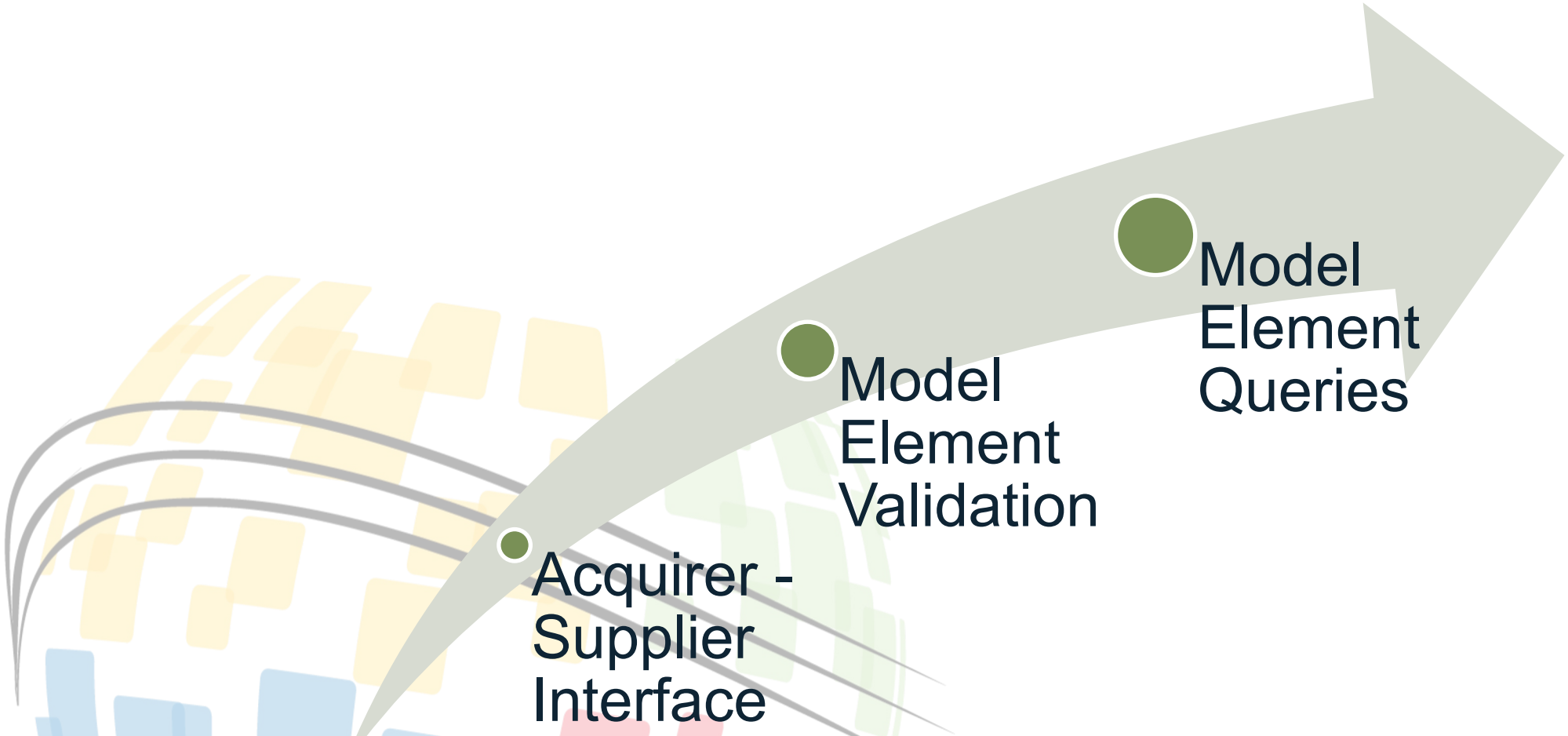
Metamodel Complexity



- 15 MetaObjects and 36 MetaRoles
  - ~108 manually generated validation rules needed to check actual model conformance
  - 15 unique queries needed to find elements in actual model

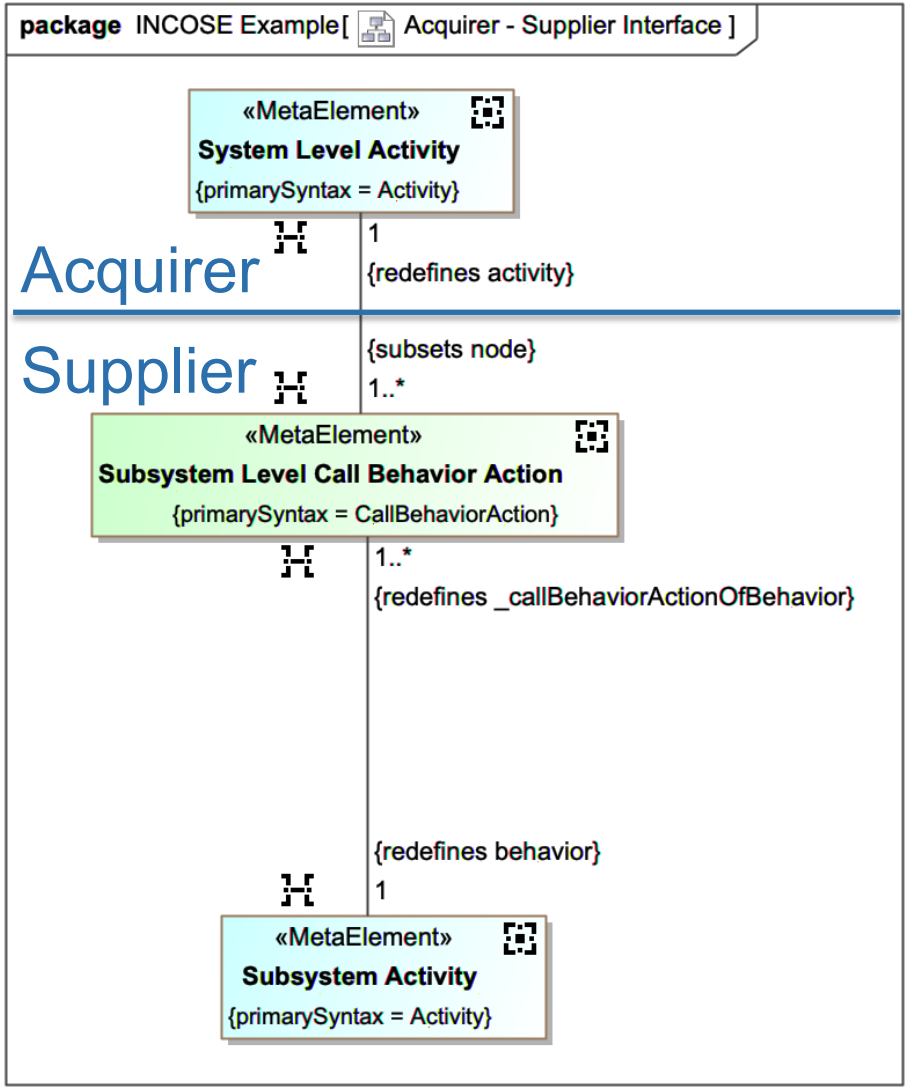
Complexity is directly proportional to the level of interrogation needed or control of actual models required

Metamodel Applications (Implemented)

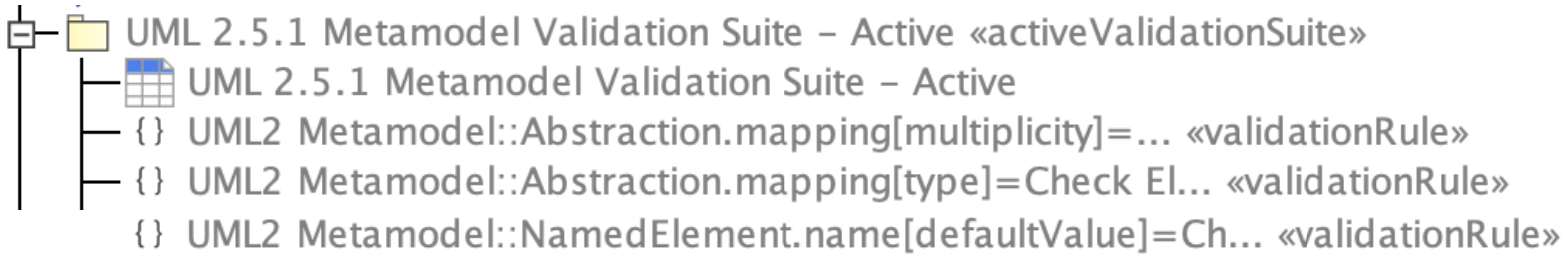


Acquirer - Supplier Interface

- System specification model is primary artifact for acquisition of goods or services in model-based acquisition
- Acquirer may specify form, structure, and relations of the implementation model via a metamodel
- Supplier may propose a metamodel with traceability to the specified elements in the acquirer metamodel

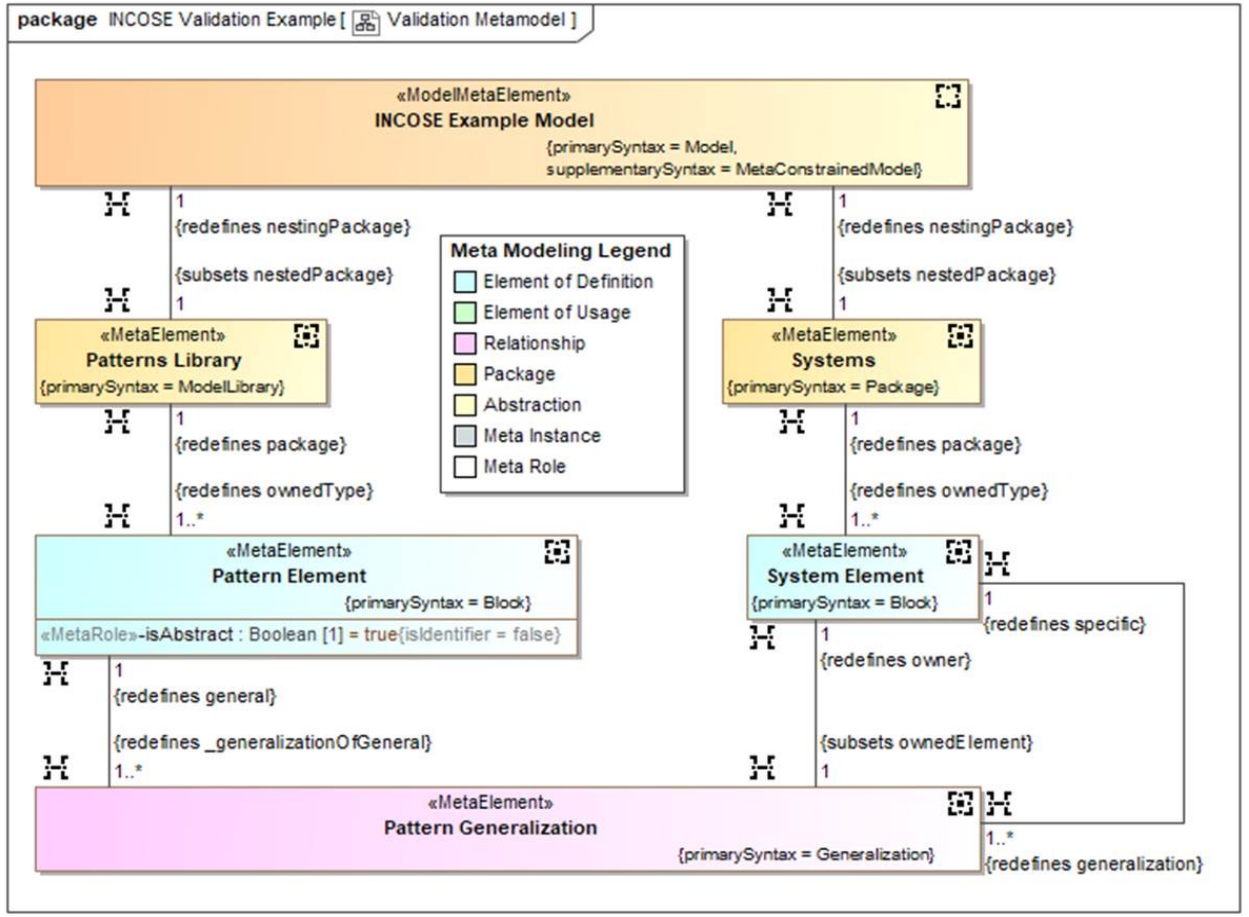


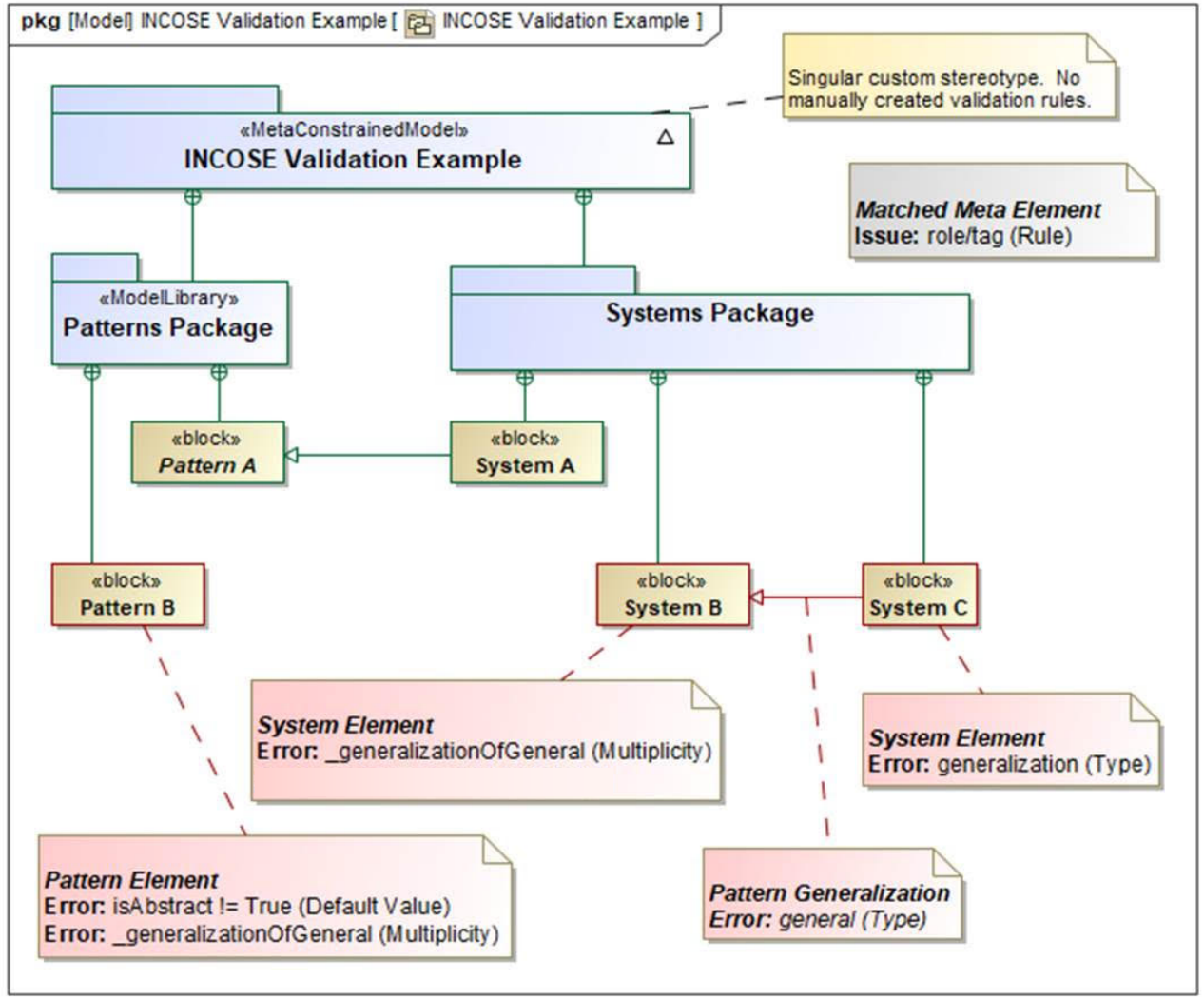
Model Element Validation (1/2)



- Metamodels used as basis for validating actual model elements
- Three inferred constraint kinds:
  - Type: Is the type of the element fulfilling this metaproperty value the same type specified in the metamodel?
  - Multiplicity: Does the number of elements fulfilling this metaproperty value match the multiplicity specified within the metamodel?
  - Default Value: Does the actual value match the specified value from the meta model
- These three rule types can support the bulk of model validation
  ~75% of surveyed validation rules could be automatically generated and contextually applied with current implementation
- 1697 validation rules programmatically generated based on UML 2.5.1 metamodel and inferred constraint kinds
  - Additional validation suites can be programmatically generated for any DSL Profile
- Implemented in No Magic Cameo Systems Modeler™ 2021x R2 by Dassault Systemes

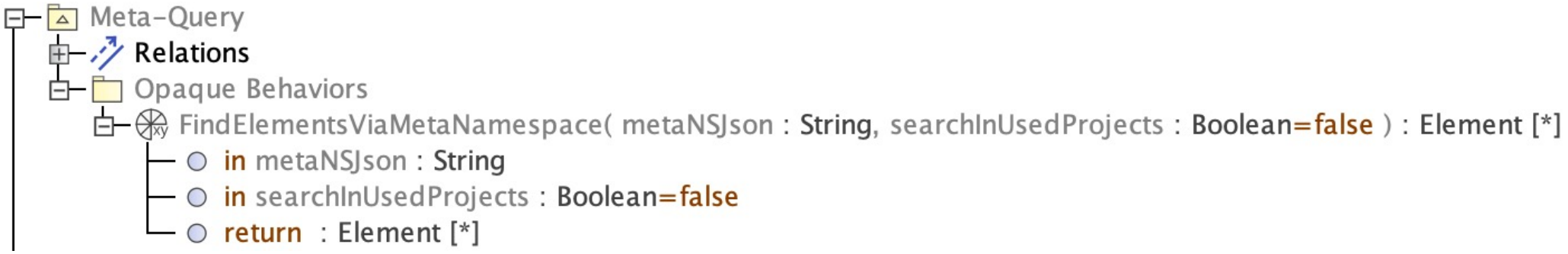
Model Element Validation (2/2)





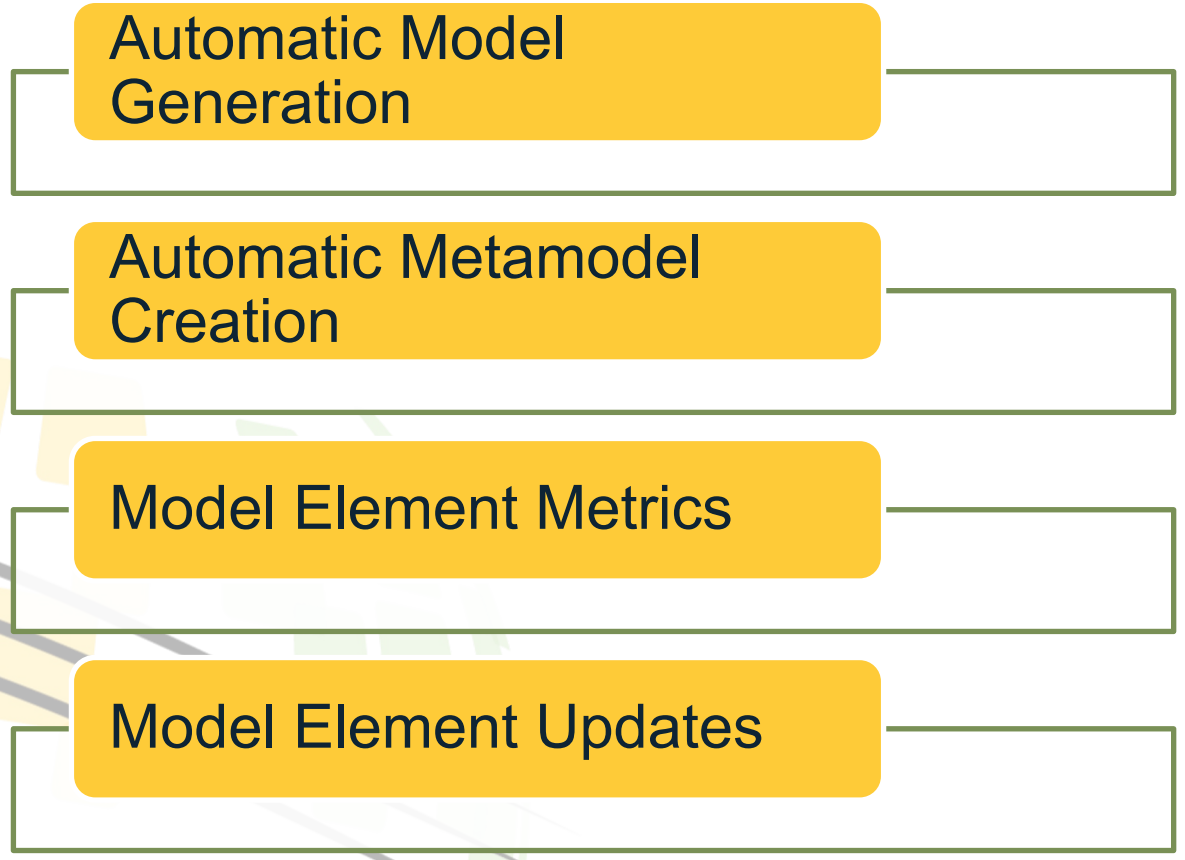
- *'System Element' blocks must specialize at least one 'Pattern Element' block*
- *'Pattern Element' blocks must be specialized and it must be abstract*

Model Element Queries



- Robust query implemented in UML Opaque Behavior
- Returns a list of all model elements matching input metanamespace
- No Magic Cameo Systems Modeler™ 2021x R2 by Dassault Systemes
- Potential improvements to enable metachaining

Metamodel Applications (Potential)



Metamodel Adoption

- Adoption and use of R^2 Metamodeling Approach is a significant investment but **directly proportional to the level of analysis and rigor required from a model**

- Traditional approaches are efforts that must be expended again and again
  1. Creation of the approach
  2. Using the artifacts of the approach in manual model review or in automated model conformance checking
  3. Manual generation of queries
  4. Repeat 2 and 3 when the approaches are updated

- **Creation of a R^2 metamodel is an effort exerted once**

Applications of a R^2 metamodel accommodate metamodel changes with minimal effort

Conclusion

- **Current approaches (domain-specific profiles, style guides, reference models, and metamodels) for ensuring system model coherency and consistency:**
  - **Are insufficient at robust and automated conformance verification of actual models to the ideal model**
  - **Rely on on non-standard, domain-specific profiles to express and enforce domain-specific semantics and rules**
- **The R^2 Metamodeling Approach resolves these issues by enabling a flexible, extensible, and user-friendly approach implemented as a lightweight UML profile**
  - **Constrained subset of UML and standardized profiles**
  - **Leverages contextual uniqueness**
  - **Automatic support of DSL**
- **Three robust applications:**
  - **Acquirer-supplier model interface**
  - **Model element validation**
  - **Model element query**

Use of the R^2 Metamodeling Approach results in reduced system model development and analysis and ensures coherency and consistency of information thus increasing stakeholder use and confidence in the system model

34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024
#INCOSEIS

Abstract

A critical enabler for Model-Based Systems Engineering (MBSE) and Digital Engineering (DE) is the generation of coherent and consistent views of a system-of-interest based on information within a system model. In practice, system model development is facilitated through domain-specific profiles, style guides, reference models, and low-fidelity meta-models to create coherent and consistent system infor- mation. Each of these approaches are useful but are insufficient for robust and automated verification of system models to an ideal. Furthermore, the expression of domain-specific concepts and semantics relies on the proliferation of non-standard, domain-specific profiles as standard system modeling languages like the Systems Modeling Language (SysML) are general purpose. This paper proposes a novel approach to creating precise, machine-interpretable metamodels implemented as a lightweight Unified Modeling Lan- guage (UML) profile. The profile includes numerous features that allow model architects to quickly specify context and domain-specific modeling constructs without creating non-standard stereotypes to apply do- main-specific meaning and usage rules. Three kinds of constraints can be inferred based on the relationships between meta-model elements: type, multiplicity, and default value. Applications of well- formed meta- models include a shared understanding of the intended model format and structure, as well as the one-time programmatic generation of an encompassing suite of validation rules to evaluate a system model against the inferred constraints, thus ensuring consistency. Additional applications include programmatic genera- tion of model analysis metrics, system models from metamodels, metamodels from reference models and element finding queries, and the ability to update a system model based upon the updated metamodel auto- matically. Use of the approach results in reduced time in system model development and analysis and en- sures coherency and consistency of information thus increasing stakeholder use and confidence in the sys- tem model.