SysML v2 Release Notes

2020-09 Release

Language features

1. *Dependencies.* Dependencies can now be defined between one or more client elements and one more supplier elements in a model.
2. *Human IDs.* A human-specified ID can now be given to any model element (previously such IDs were only supported on requirements). Unlike an element name, the human ID of an element is inherent to that element rather than being relative to some namespace. The human ID of an element can be used as an alias name for it in a qualified name.
3. *Comment Bodies.* Comment bodies are now processed to removed surrounding `/*` and `*/` markers and line-initial `*`s, as specified in the initial submission specification.
4. *Documentation Comments.* Comments using the **doc** keyword now identify special documentation comments directly owned by their annotated element. Requirement text is now specified using documentation comments for the requirement, rather than just being the first regular comment.
5. *Textual Representation.* Any element can now have a textual representation annotation. In particular, this can be used to give the text for an "opaque" element in a given language.
6. *Subject Parameters.* Subject parameters for requirements and cases are now specified as features in the bodies of those elements using the **subject** keyword, rather than as the first parameter.
7. *Parameter flow-feature syntax.* Parameters in general may now be declared inside the body of behavioral elements (actions, states, calculations, constraints, requirements, cases) in the same way that flow features are declared on structural elements, by prefixing the declaration with the appropriate direction (**in**, **out**, **inout**). For calculations and cases, the result parameter can also be so declared by using the keyword **return** in place of a direction.
8. *Verification Cases.* An initial concrete syntax has been added for verification cases, but this is simply a placeholder and not yet fully supported in the language.
9. *View and Viewpoint Modeling.* An initial concrete syntax has been added for views, viewpoints and renderings, but this is preliminary and not yet complete.
10. *Quantities and Units.* The Quantities and Units model library has been significantly extended to include many quantity kinds and units defined in additional parts of the ISO 80000 standard, including Atomic and Nuclear, Chemistry and Molecular, Electromagnetism, Light, Mechanics, and Thermodynamics (Space and Time was already included in 2020-06).

Backward incompatibilities

1. *Keywords.* The following new keywords have been added as reserved words:
   all, **dependency**, **doc**, **expose**, **istype**, **language**, **rendering**, **rep**, **subject**, **verification**, **view**, **viewpoint**

2. *Action Syntax.* The syntax of named perform, accept and send action usages, and named exhibit state usages, has been changed to be more consistent.

   - `perform performingAction as performedAction`
     becomes
     `perform action performingAction :> performedAction`

   - `exhibit exhibitingState as exhibitedState`
     becomes
     `exhibit state exhibitingState :> exhibitedState`

   - `accept acceptingAction (sig : Signal)`
     becomes
     `action acceptingAction accept sig : Signal`
     (note, in particular, the parentheses are no longer required)

   - `send sendingAction of Signal() to target`
     becomes
     `action sendingAction send Signal() to target`

3. *Distinguishability Validation.* A warning validation has been added that checks whether all owned members in a namespace have names that are distinguishable from each other, from the names of inherited members and the human IDs of any owned elements. This check can fail, in particular, if an element is defined with the same name as an inherited element, without explicitly redefining the inherited element.

API

1. *Validation.* The API now validates to the 2020-09 baseline of the abstract syntax.
2. *Query Service.* Endpoints have been added to support the definition and execution of simple model queries that allow the specification of specific element attributes to be returned and constraints on what is selected.
3. *Commits.* There has been a small backward incompatible change to the model for a Commit, where the Commit::changes attribute has been renamed to Commit::change.

Visualization

1. *Tom Sawyer*
   - Structural modeling has been improved to reflect planned standard visualization for SysML v2.
   - Action compartments are now shown on parts and "performed by" compartments on actions.
   - Small arrows are now shown in ports to reflect flow feature direction.
   - Behavioral modeling has been improved to render control flows and control nodes.

2. *PlantUML*
   - Action/activity modeling has been added.
   - Guards and effects are now shown on transitions in state machine diagrams. c. Alias names are now properly rendered for elements.

2020-06 Release

Language features

1. *New definition/usage keywords*. New keywords have been introduced corresponding to the consistent definition/usage terminology in the 2020-06 baseline of the abstract syntax. The old keywords still work as synonyms for the new keywords.
    - **block → part def**
    - **assoc block → connection def**
    - **link → connection**
    - **value type → attribute def**
    - **value → attribute**
    - **activity → action def**
2. *Item definition and usage.* An *item* is a model of something that exists in space and time but is not considered to necessarily itself be a system or part of a system. An item may be spatially discrete (like a part) or continuous (like water or fuel), and it may flow through, be stored in or be acted upon by a system.
3. *Case definition and usage*. A *case* is a model of the steps necessary to produce a desired result regarding some subject, satisfying a specific objective. The objective of a case is modeled as a requirement.
4. *Analysis case definition and usage.* An *analysis case* is a case carrying out an analysis on its subject. The steps of an analysis case may be drawn from a set of analysis actions that abstract typical capabilities to be applied by analytical tools during an analysis.
5. *Variability modeling.* Any kind of definition or usage may be identified as a *variation* that models a point at which a selection can be made between a specified set of choices during the configuration of a system. All the members of a variation must be variant usages that model the allowable choices for configuration of the variation.
6. *Succession shorthand.* The **then** shorthand can now be used to show succession between individual usages (particularly snapshots and time slices).
7. *Behavioral usages with direction.* It is now possible to declare a direction on behavioral usages (e.g, "**in action** …").
8. *Nested behavioral usages*. It is now possible to have behavioral usages generally nested in action and state usages (this allows, in particular, the nesting of constraint usages in action and state usages).
9. *Space and time quantities and units*. The new package `ISQSpaceTime` in the `Quantities and Units` domain library covers all quantities and units defined in ISO 80000 Part 3 Space and Time.

Backward incompatibilities

1. *Keywords.* The following new keywords have been added as reserved words:
attribute, **analysis**, **calc**, **case**, **connection**, **item**, **objective**, **return**, variation, variant
2. *Direction and end keyword placement*. The keywords **in**, **out**, **inout** and **end** now always come before other keywords. For example, the previously allowed syntax `part in x;` and `part end x;` must now be written `in part x;` and `end part x;`, respectively.
3. *Behavioral definition and usage specialization.* The specialization parts for the declaration of behavioral definitions and usages (actions, states, calculations, constraints, requirements, cases and analysis cases) must now come before the parameter parts. For example, the previously allowed syntax `action def A (in x) :> B;` and `action a: A (in x = y) :>> b;` must now be written `action def A :> B (in x);` and `action a: A :>> b (in x = y);`, respectively.
4. *Measurement units*. Measurement units are now attributes, rather than parts. This is consistent with the ability to perform arithmetic operations on measurement units (e.g., `kg*m/s`). This mostly affects user models that define new measurement units. Models that simply use standard units from the SI or US Customary Units libraries should generally be unaffected.
5. *Public imports in quantities and units models.* The use of public import in the `Quantities and Units` packages has been limited to the cases listed below. This may result in name resolution errors when reprocessing existing models that import `Quantities and Units` library models. Such errors can be fixed by either explicitly qualifying the names in question or by importing them directly from the appropriate package.
    - The `SI` package re-exports names from the `ISQ` package (e.g., quantity value types, such as `LengthValue`, etc.) and `SIPrefixes` (e.g., `centi`, `milli`, `kilo`, etc.).
    - The `USCustomaryUnits` package re-exports names from the `ISQ` package.

Validations

1. *Default validation.* Previous default metamodel validation checking has been disabled (which greatly reduces the cascade of unclear messages due to small errors).
2. *Typing errors.* Validation checks have been implementing for the typing of specific SysML elements (e.g., a port usage must be typed by a port definition, etc.).
3. *Warnings*. The following warning validations have been implemented (some of which were previously errors). Unlike errors, warnings do not prevent a model from being fully syntactically processed, though it may not be semantically valid.
    - *Connected features should have a common context.* This warning indicates that there is no common containing context from which a connector and each of its connected features can be reach by valid ownership paths.
    - *Bound features should have conforming types*. This warning indicates that the types of the features connected by a binding connector do not actually allow for the possibility of the bound features having common values.
    - *Subsetting/redefining feature should not be nonunique if subsetted/redefined feature is unique* (previously an error).
    - *Subsetting/redefining feature should not have larger multiplicity upper bound* (previously an error). This is only checked if the upper bounds are given as literal values.
    - *Redefining feature should not have smaller multiplicity lower bound* (previously an error). This is only checked if the lower bounds are given as literal values.
    - *Owner of redefining feature should be a specialization of owner of redefined feature.* A redefinition is supposed to be of a feature that would otherwise be inherited, and inherited features come from the generalizations of the owner of the redefining feature.
    - *Owner of redefining feature should not be the same as owner of redefined feature.* This also follows from the requirement that a redefinition be of a feature that would otherwise be inherited.
    - *A package-level feature should not be redefined (by another package-level feature).* A feature that is declared in a package that is not a type is effectively considered to be a feature of the most general type `Anything`. Therefore, redefining a feature at "package-level" with another feature at package-level would be like redefining a feature with the same effective owner, `Anything`.

Visualization

1. *Tom Sawyer*
    - Tree and interconnection views have been significantly updated based on user feedback.
    - A basic action model view has been added.
2. *PlantUML*
    - PlantUML visualization is now available in Jupyter Notebook using the '%viz' magic command.
    - An interconnection view has been added.
    - Errors in state visualization have been corrected.

API

1. *JSON-LD.* The API payloads now uses JSON-LD for type and other object references. Previously, the type field in JSON responses was the name of the relevant metaclass, and the identifier (UUID) was used for object references.
2. *Roots endpoint.* A `GET ../roots` endpoint has been added to get root elements by project and commit.
3. *Relationships endpoint.* The `GET …/relationships` endpoint functionality has been expanded to get relationships for the given element as source, target or both.

2020-03 Release

Language features

1. Added constraints
2. Added requirements
3. Added individuals and snapshots
4. Updated alias notation
5. Updated conditional successions
6. Added Base Functions
7. Added Scalar Functions
8. Updated Quantities and Units model

API

1. Added support for element versioning

Visualization

1. *Tom Sawyer.* Updated.
2. *PlantUML.* Initial release of tree and state machine views for Eclipse editors.

2019-12 Release

Language features

1. Added States
2. Added conjugated port definitions
3. Added condition successions
4. Updated send and receive actions
5. Made minor keyword changes
6. Added validation checks for subsetting and redefinition.

Visualization

1. *Tom Sawyer.* Updated.

API

1. Added validation against abstract syntax

2020-09 Release

Language features

Initial release:

1. Packages
2. Blocks and value types
3. Parts
4. Connectors
5. Ports
6. Interfaces
7. Binding connectors
8. Item flows
9. Activities and actions
10. Expressions
11. Scalar values and functions
12. Quantities and units

Visualization

1. *Tom Sawyer.* Initial release for structural diagrams.

API

Initial release:

1. `GET` and `POST` elements
2. `GET` and `POST` relationships