INCOSE

International Symposium 2026

A usage-first modelling methodology in SysML v2

Tommie Liddy, Turen Pty Ltd
Matthew Wylie, Shoal Group Pty Ltd

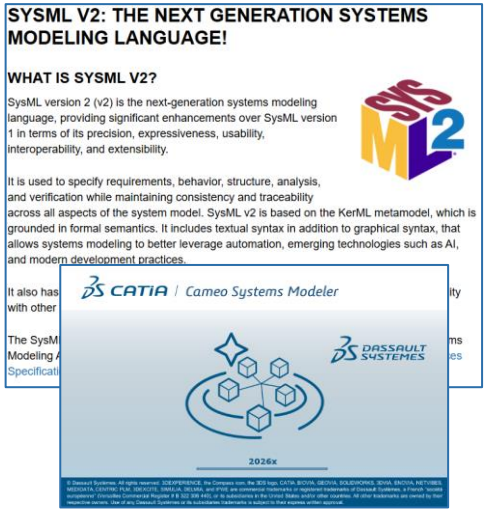
incose.org

Context

- SysML v2 was released (technically in March 2026)
- Pre-release ‘Beta’ versions have been used by tool vendors to implement in their tools
- Practitioners have had access to SysML v2, in tools for the last 12 months (or so)

“If you always do what you’ve always done, you’ll always get what you’ve always got”

- Henry Ford (and also my mum when I repeatedly made the same mistakes!)



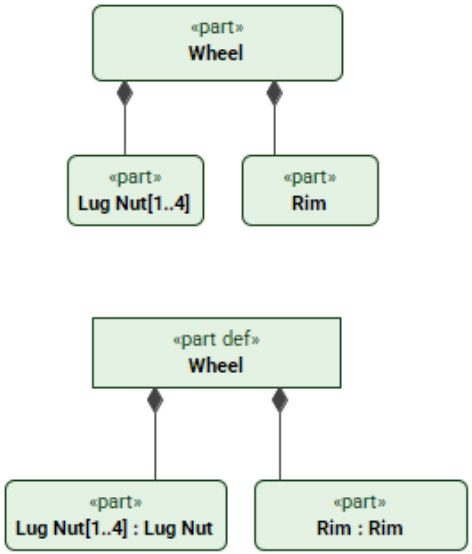
TUREN SHOAL

Reference https://www.omg.org/sysml/sysmlv2/

SysML v2 highlights

- SysML v2 is not just a revision of SysML v1 it changes key modelling mechanics
- Important changes explored in this presentation:
  - Element Features, and **Feature Ownership**
  - Definition, Use and **Subsetting**
- These changes offer practitioners an opportunity to reconsider how we model various systems concepts

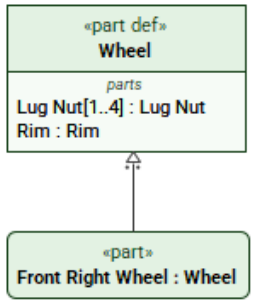
Element Features and Feature Ownership

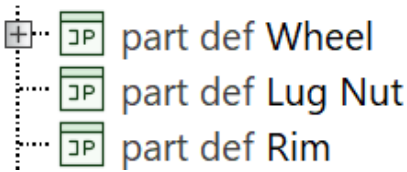


- Features are owned usage elements nested within the parent element (parts, actions, items etc.)
- Features can now be owned by usage element
  - There seems to be no semantic difference when comparing definition owned features with usage owned features
  - This makes the containment representation much more meaningful
- Feature ownership can be done without creating definition elements

Definition, Usage and Subsetting

- The Definition and Usage mechanic (feature typing) allows for reuse of a single definition in multiple contexts
  - Usage elements representing context dependent instances of the definition element (part/part def)
  - Usage elements reference the features of the definition element but do not own them (though they can own additional features)
- Definition elements often sit off to the side in a library

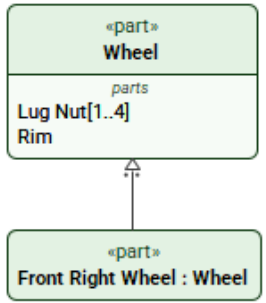




TUREN SHOAL

Definition, Usage and Subsetting

- SysML 2 has introduced Subsetting
  - Allows subclassification (read generalisation) between usage elements
  - Which allows inheritance of features
- Subclassification and feature typing are not the same, however…
  - Specialised elements reference the features of the more generalised element but do not own them (though they can own additional features)
- **Subsetting** can be used when an element needs to reference an existing elements features

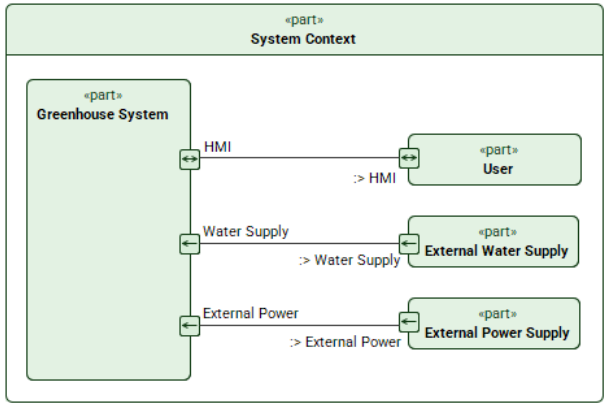


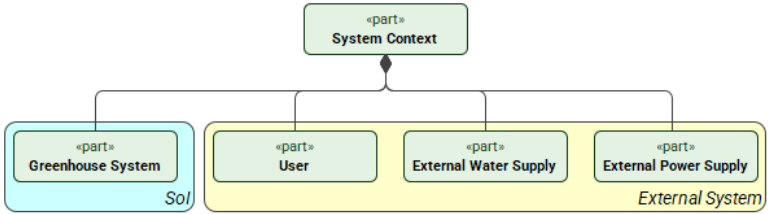
Initial ‘Usage-first’ Approach

- Capture and relate the system design information with usage elements
  - Aim for usage-only, can these concepts be developed with no definition elements
  - Move to Definition and Usage if absolutely necessary
  - Adhere to the SysML v2 standard but focus on output, not modelling purity
- Go through a system design lifecycle and investigate the following
  - Can various system concepts be represented?
  - Does this approach provide modelling efficacy?
  - What are the inherent limitations of applying it?

System Context

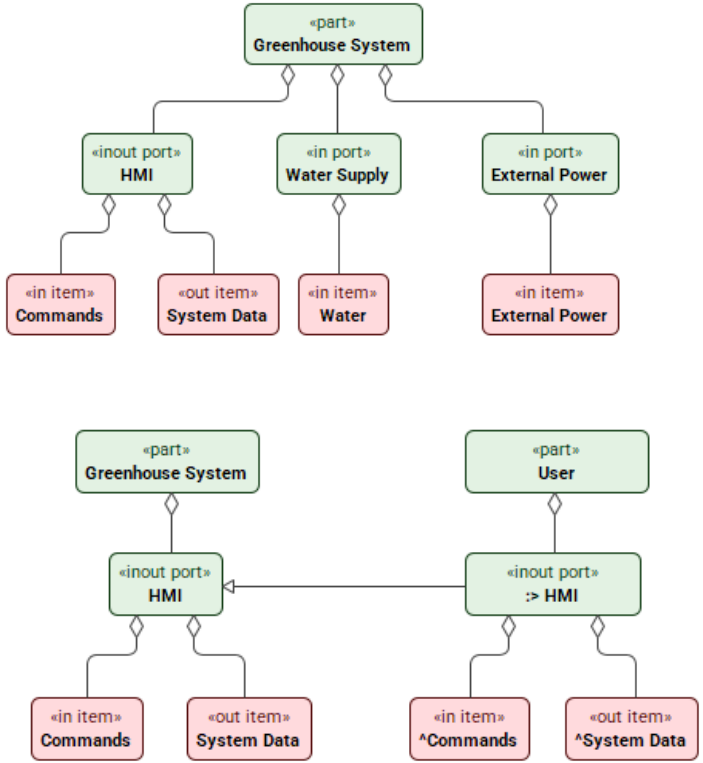
- Example system: **The deployable greenhouse**
- The system context, Greenhouse system, and external systems all modelled with part elements and feature membership





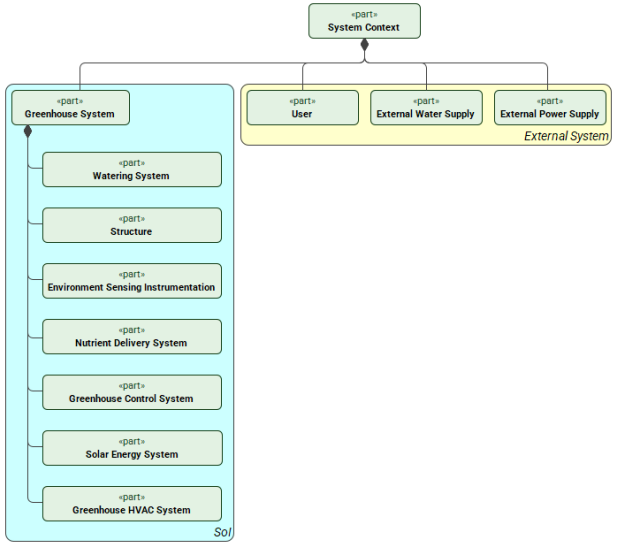
Interactions/Interfaces

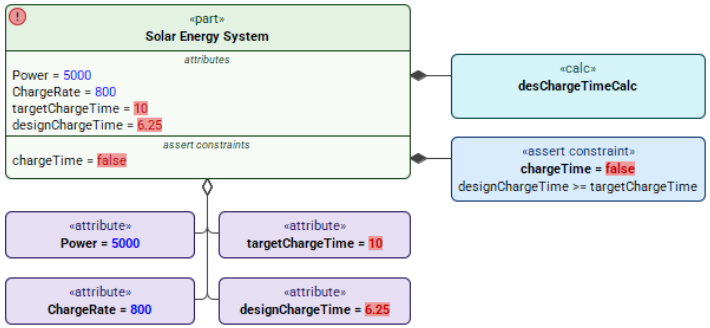
- Defining ports, without a port def elements
  - The part owns ports as features, the ports own items as features
  - The items have a direction (one of their properties)
- To flow the definition to other ports
  - One port is defined, Other ports inherit the features through subsetting
  - Port item directions cannot be conjugated when using subsetting



System Hierarchy

- Continued building the hierarchy with part elements
- Added attributes, calculations, constraints
  - Usage-only just works (especially in Text Notation)
  - No opportunity for reuse, each calc and constraint is its own element in a single modelling context



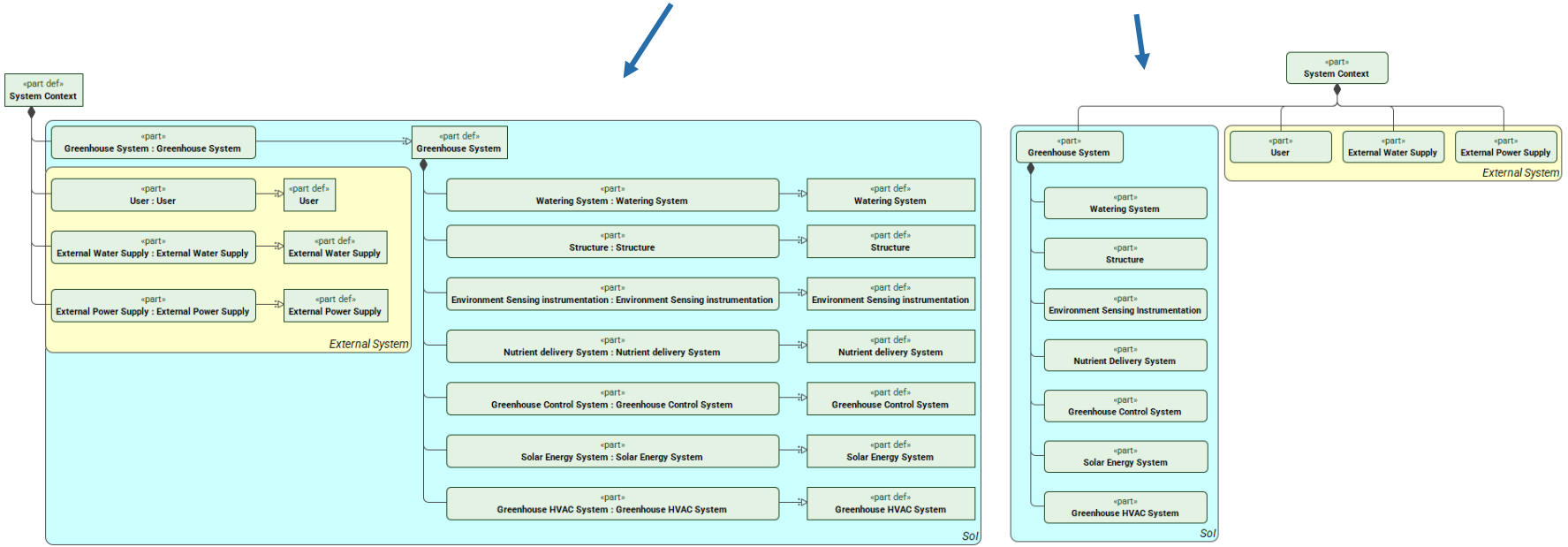



```
part [$115] 'Solar Energy System' {
    part [$116] 'Solar Cells';
    part [$117] 'Battery System';
    part [$118] 'Power Distribution System';
    satisfy [$119] Requirements::'Solar Energy Sub-System Requirement 1';
    satisfy [$120] Requirements::'Solar Energy Sub-System Requirement 2';
    attribute [$121] Power = 5000;
    attribute [$122] ChargeRate = 800;
    attribute [$123] targetChargeTime = 10;
    attribute [$124] designChargeTime = desChargeTimeCalc(Power, ChargeRate);
    calc [$125] desChargeTimeCalc {
        in [$126] capacity;
        in [$127] chargeRate;
        return [$128] result = capacity / chargeRate;
    }
    assert constraint [$129] chargeTime {
        designChargeTime >= targetChargeTime
    }
}
```

TUREN SHOAL

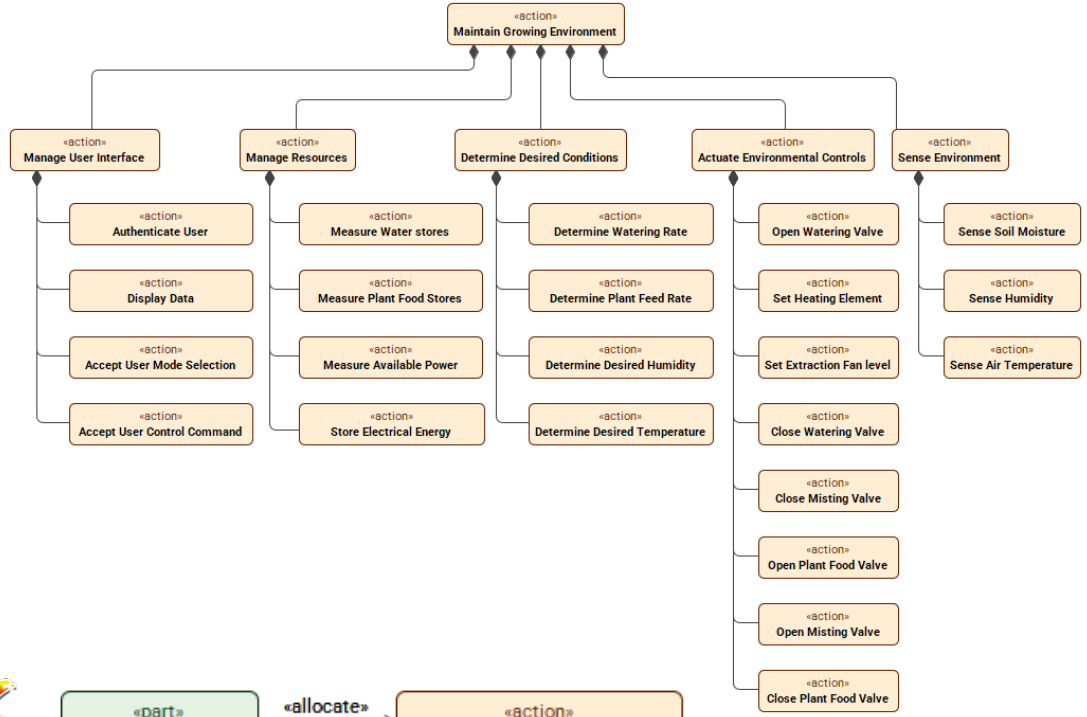
Comparison of approaches (Hierarchy)

- Direct comparison between definition-and-usage and usage-only patterns



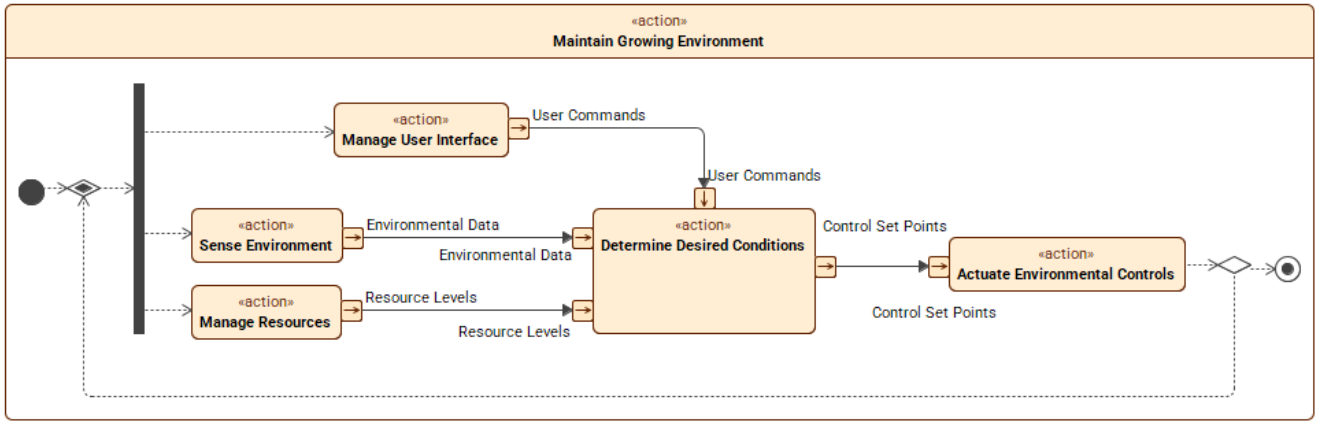
Functional Hierarchy

- Action elements used to define the functional hierarchy (this is just brilliant!!!)
- Functional allocation
  - Subsetting a perform actions as a direct features of the part
  - Using the allocation element and remain usage-only



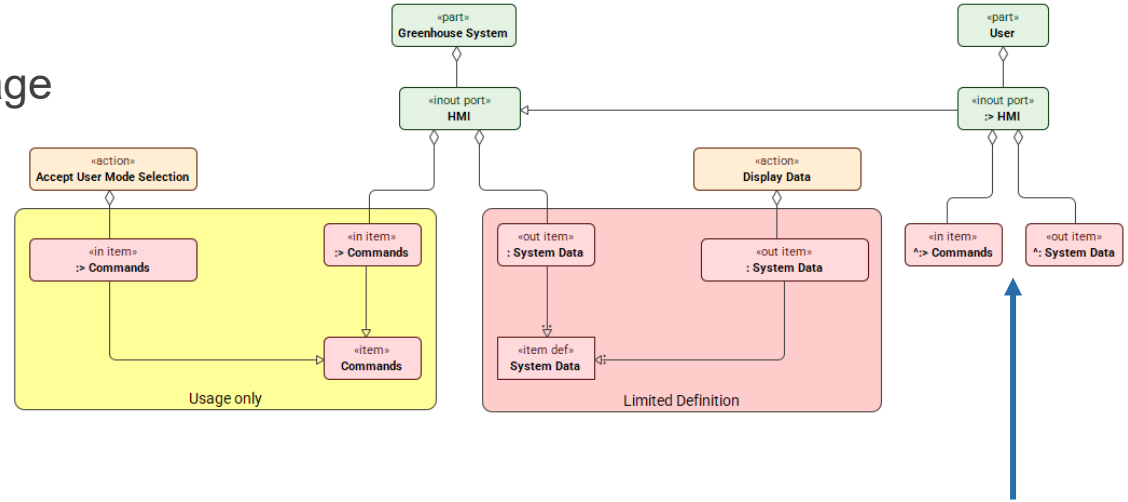
Functional Behaviour

- Behaviour was easy enough to define…it's the data flows that were a challenge
- Actions own item with directions (as features), these **should** be the same items as those owned by the ports



Item ‘Definition’

- Usage only approach
  - Create the item in a library package
  - Subset the port or action feature, and add direction
- Limited definition approach
  - Create the item def in a library package
  - Feature type a usage in the port or action feature, and add direction

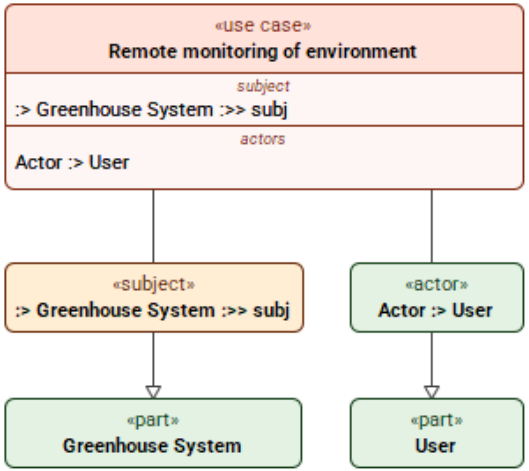


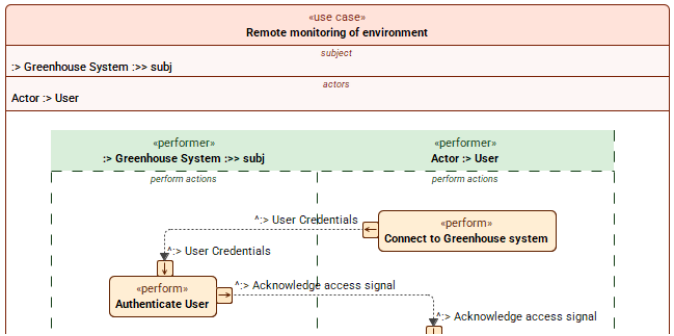
Note that usage only port-to-port subsetting remains the same

TUREN SHOAL

Use Cases (and lots of subsetting)

- Had to use subsetting for the ‘Subject’ and ‘Actor’ elements of the use case
  - Default structures of the use case
  - Wanted those parts to act in the use case
- Relationships between parts and actions did not consolidate from the use case to the part

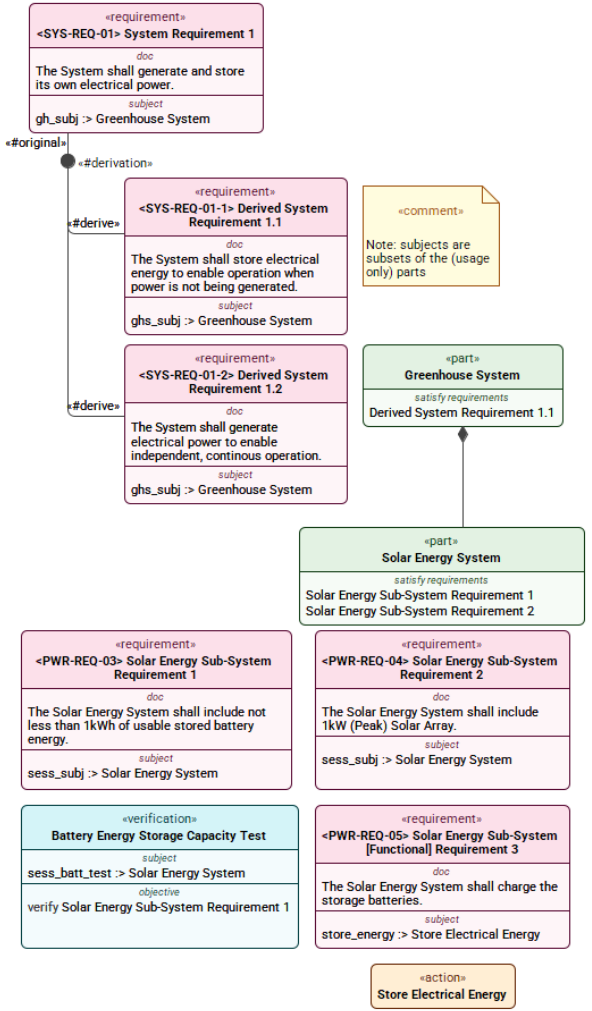




TUREN SHOAL

Requirements and Verification Cases

- Usage only had no impact on requirement derivation and dependency relationships
- Used subsetting to use Parts or Actions as a Requirement and Verification Subjects
- No issues with ‘Part’ ‘Satisfy Requirement’ properties
- No issues specifying ‘Requirements’ [usage] as ‘Verification’ ‘Objectives’



TUREN SHOAL

Conclusion

Can various system concepts be represented?	• Yes (not really in doubt), context layer, hierarchy, behaviour, interfaces, requirements, and design traceability

Does this approach provide modelling efficacy?	• Yes, fewer elements need to be created
	  • Example: Generating a system hierarchy takes approximately half the elements
	• (Subjectively) Model comprehension seems easier:
	  • Modelling is done within the context of the concept being explored
	  • Containment structure (thank you SysML v2) is far more digestible

What are the inherent limitations of applying it?	• Subsetting and feature typing are not the same and do not get treated in the same way by the tool • Some things are better as a library of elements, such as items, as opposed to a single instance

TUREN SHOAL

Questions

Tommie Liddy

Turen Pty Ltd

Matthew Wylie

Shoal Group Pty Ltd

TUREN SHOAL