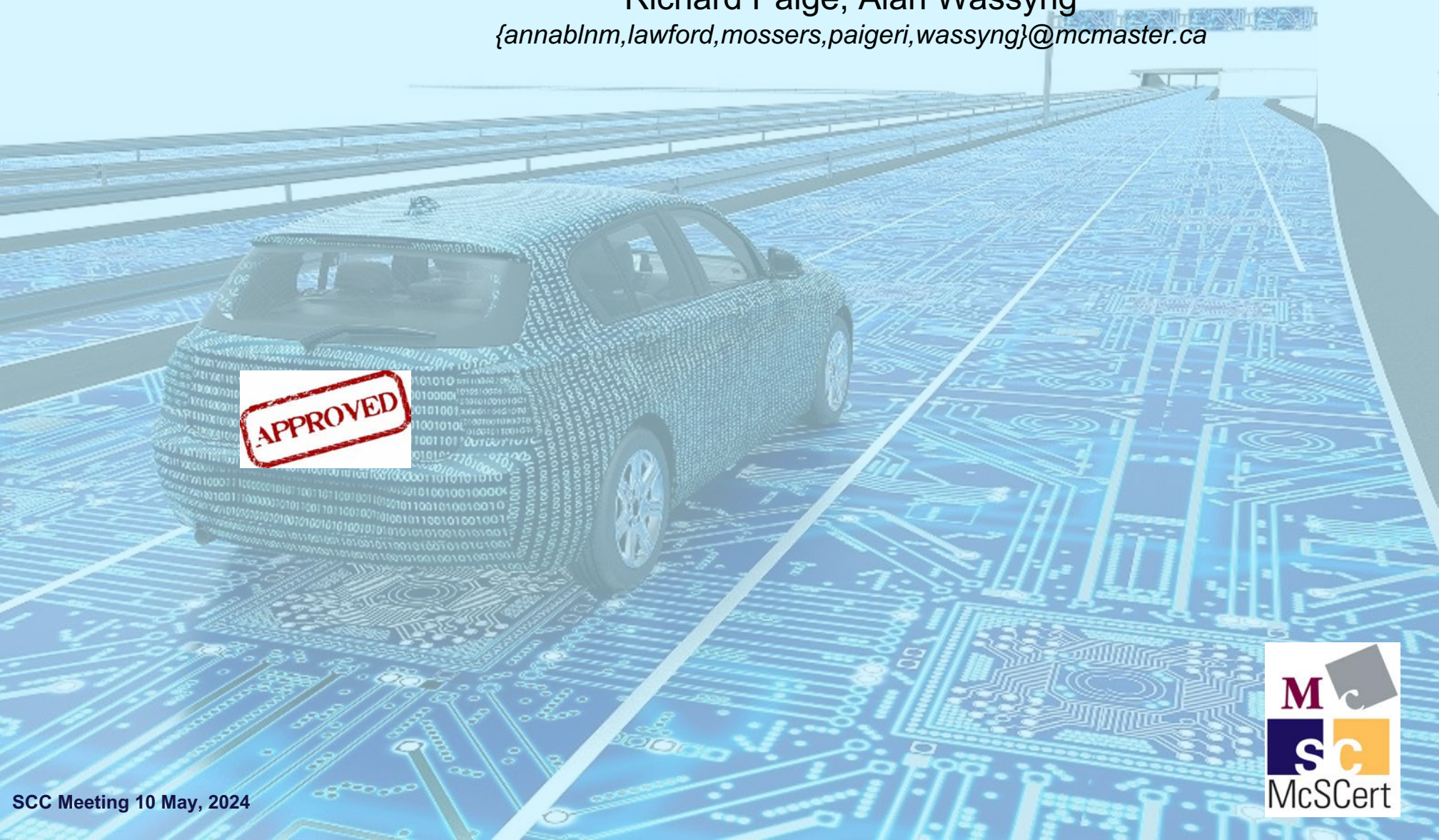
McMaster University

Inspiring Innovation and Discovery

Using Workflow+ to Assure Safety of Automotive Over-the-Air Software Updates

Nicholas Annable, Mark Lawford, Sébastien Mosser, Richard Paige, Alan Wassyng

{annablnm,lawford,mossers,paigeri,wassyng}@mcmaster.ca



McSCert

SCC Meeting 10 May, 2024

Updating Automotive Systems Over The Air

Automakers have been updating their cars for years …

…Delivering updates over the air is the direction many OEMs are going in & this poses new challenges

Benefits

Convenience

Cost

Reach

- Vehicle can be contacted directly

Challenges

Reliability & Safety & Security

- Scalable methods
- Installed without supervision
- Dealing with product variants

Updating Automotive Systems Over The Air

Challenges

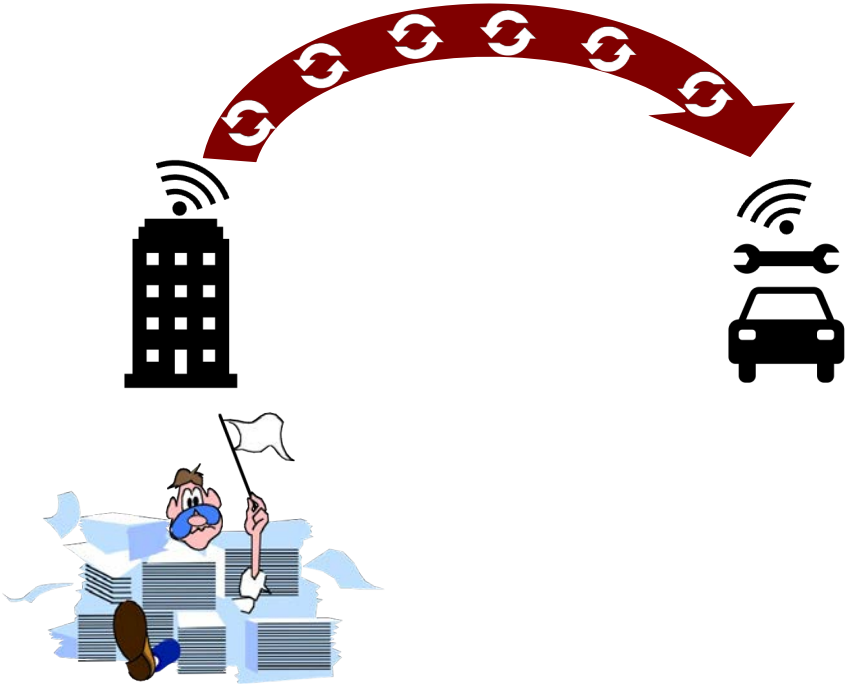
Assuring Reliability, Safety & Security – we need to:

- **Develop scalable methods for more frequent updates**
- **Update & verify updates automatically without skilled supervision**
- **Deal with many product variants each with thousands of components**

Much more frequent updates

Existing methods will not be able to keep up with demand

What can we do so that updates can be planned and assured to be reliable, safe & secure with low enough effort?



Updating Automotive Systems Over The Air

Challenges

Assuring Reliability, Safety & Security – we need to:

- **Develop scalable methods for more frequent updates**
- **Update & verify updates automatically without skilled supervision**
- **Deal with many product variants each with thousands of components**

Suppliers of components, third-party apps running in vehicle may want to update their own apps

Need mechanisms to ensure safety when updating code from suppliers that may be black-box

Updating Automotive Systems Over The Air

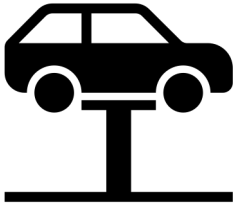
Challenges

Assuring Reliability, Safety & Security – we need to:

- **Develop scalable methods for more frequent updates**
- **Update & verify updates automatically without skilled supervision**
- **Deal with many product variants each with thousands of components**

Updates used to be deployed in a controlled environment, with skilled workers performing the process

- Success/failure of update could be checked
- Manual roll back if there were issues
- Worker could physically check status of vehicle before/after update



Updating Automotive Systems Over The Air

Challenges

Assuring Safety & Security – we need to:

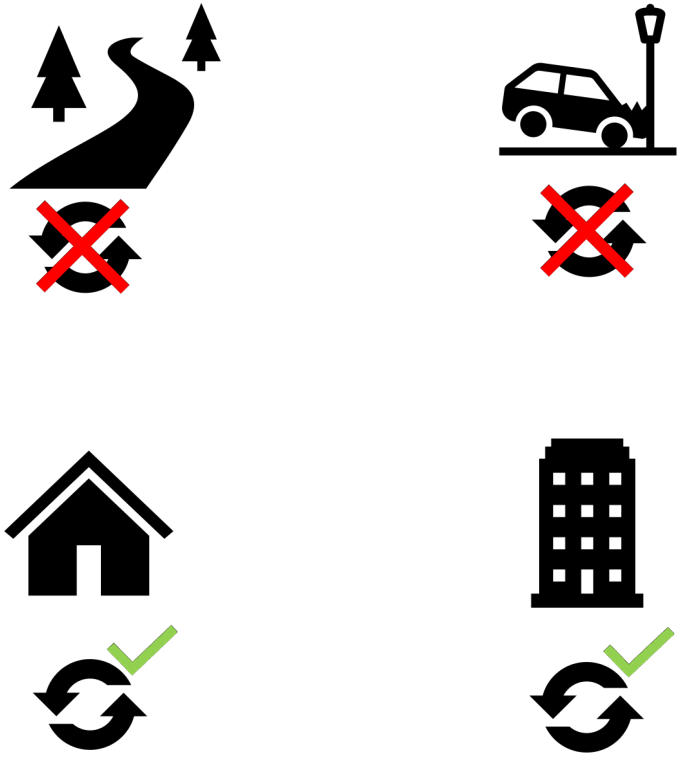
- Develop scalable methods for more frequent updates
- Update & verify updates automatically without skilled supervision
- Deal with many product variants each with thousands of components

Now we need to make sure it is safe to update entirely automatically

Environment

Current condition of vehicle, components present, maintenance history, etc.

Everything must be checkable through software



Updating Automotive Systems Over The Air

Challenges

Assuring Safety & Security – we need to:

- Develop scalable methods for more frequent updates
- Update & verify updates automatically without skilled supervision
- Deal with many product variants each with thousands of components

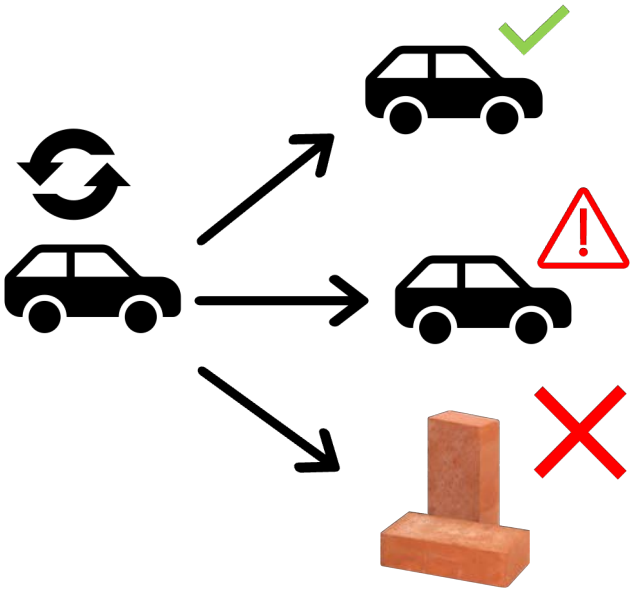
Now we need to make sure it is safe to update entirely automatically

Automatic checks must be able to identify if vehicle is in a safe state

- User might not know what intended behaviour is after update

Failed updates must be able to be rolled back automatically, or user may be stuck

- Could be hazardous depending on where the update was started



Updating Automotive Systems Over The Air

Challenges

Assuring Safety & Security – we need to:

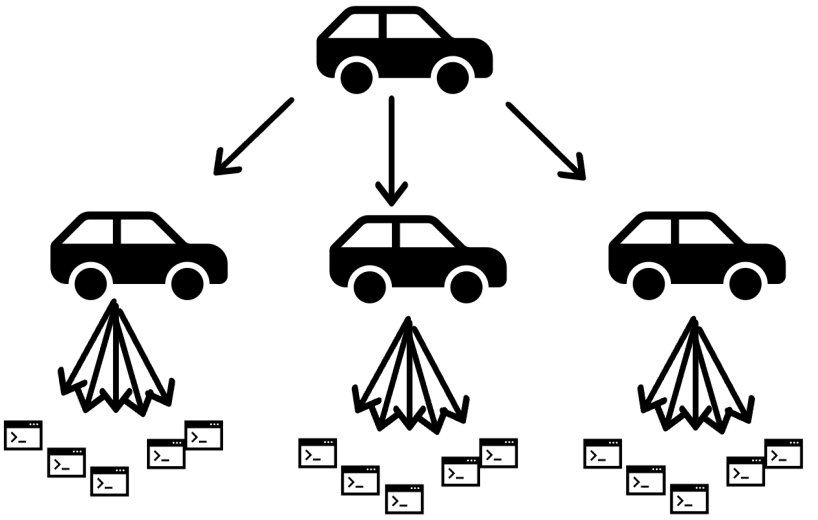
- **Develop scalable methods for more frequent updates**
- **Update & verify updates automatically without skilled supervision**
- **Deal with many product variants each with thousands of components**

Incremental assurance is difficult on a 'small scale' with several variants

When an update is applied, a new variant is created

Need to ensure hardware is compatible with software and meets its assumptions

How can we manage this complexity and support reuse across product families?



How Can We Address These Challenges?

Lots of difficult challenges… but we have a plan

McSCert has been developing the WorkFlow+ modelling framework in collaboration with GM

The main goals are:

- Support impact analysis & incremental assurance through extensive traceability
- Support more systematic, rigorous and traceable generation of safety & security assurance arguments

We believe an extended version of our framework will address these problems

McSCert's Approach: WorkFlow+ (WF+)

WF+ is a formal, model-based framework to model information necessary for (incremental) safety assurance

Models processes, data (work products) and constraints

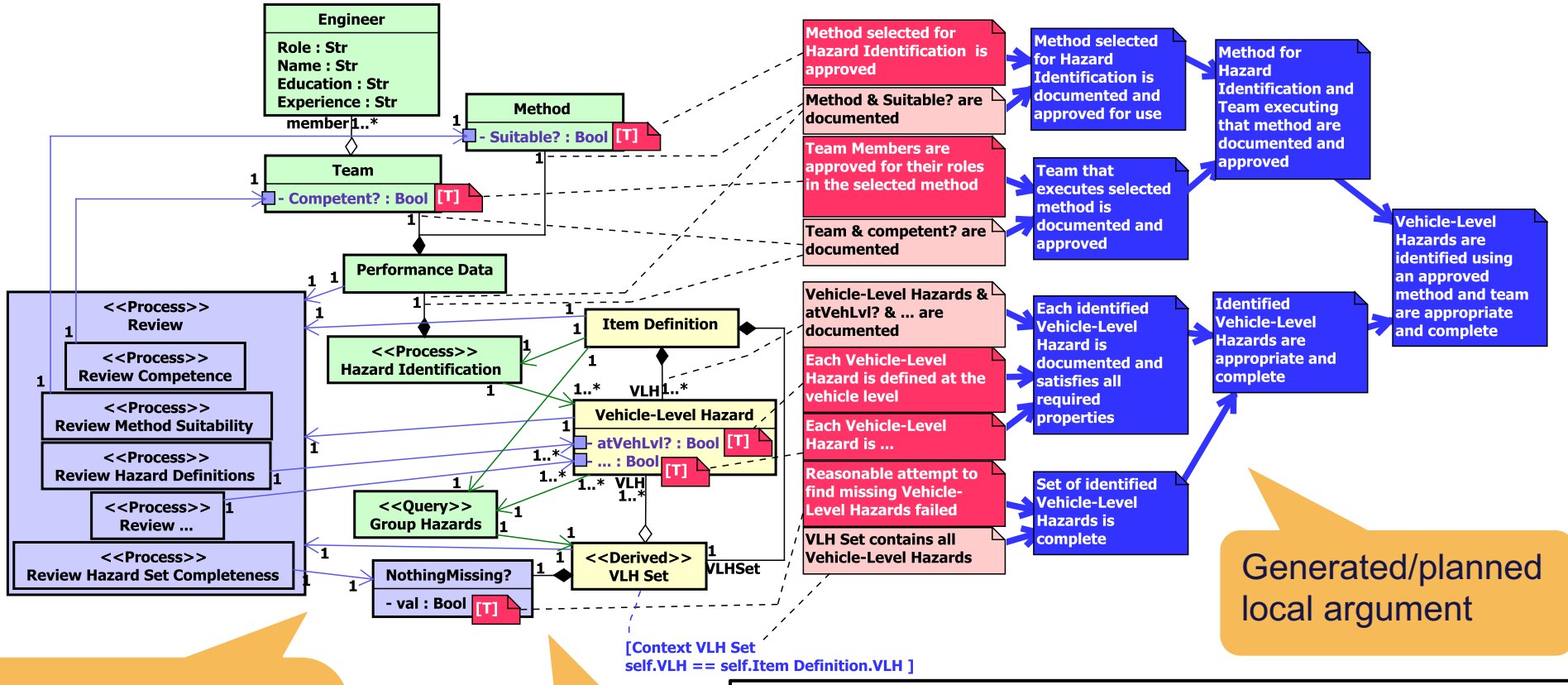
Experts encode knowledge on what must be done to ensure system safety in WF+ metamodels before development (similar to a safety plan and/or an assurance case template)

Exports encode safety and other attributes into the model by inserting relevant constraints that they require to be true. The constraints can be syntactic (multiplicity constraints for example), or semantic (specific review results for example).

Arguments are based on these constraints and what they guarantee about instances – we use this to generate the segment of the assurance case related to that activity

When a workflow is executed, the WF+ metamodel is instantiated to document the execution

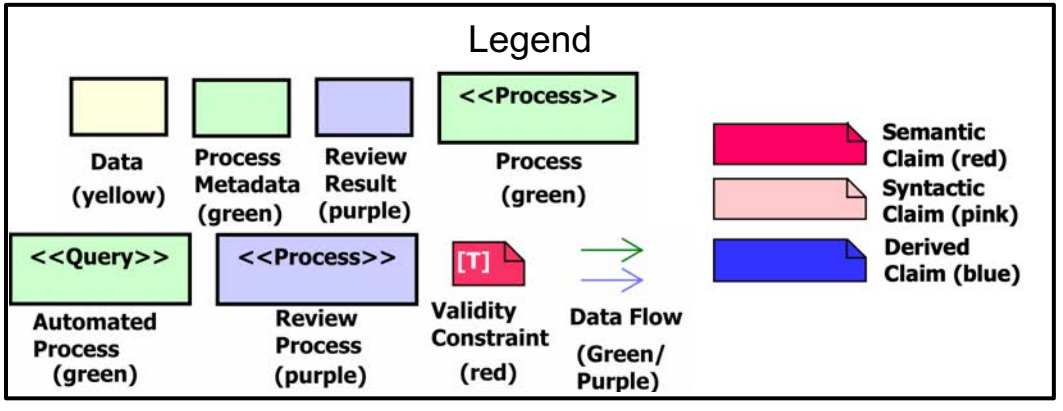
WF+ Example: Metamodel



Example WF+ metamodel [2], based on automotive functional safety standard ISO 26262 [1]

Template of evidence in WF+ metamodel, including product data

Generated/planned local argument

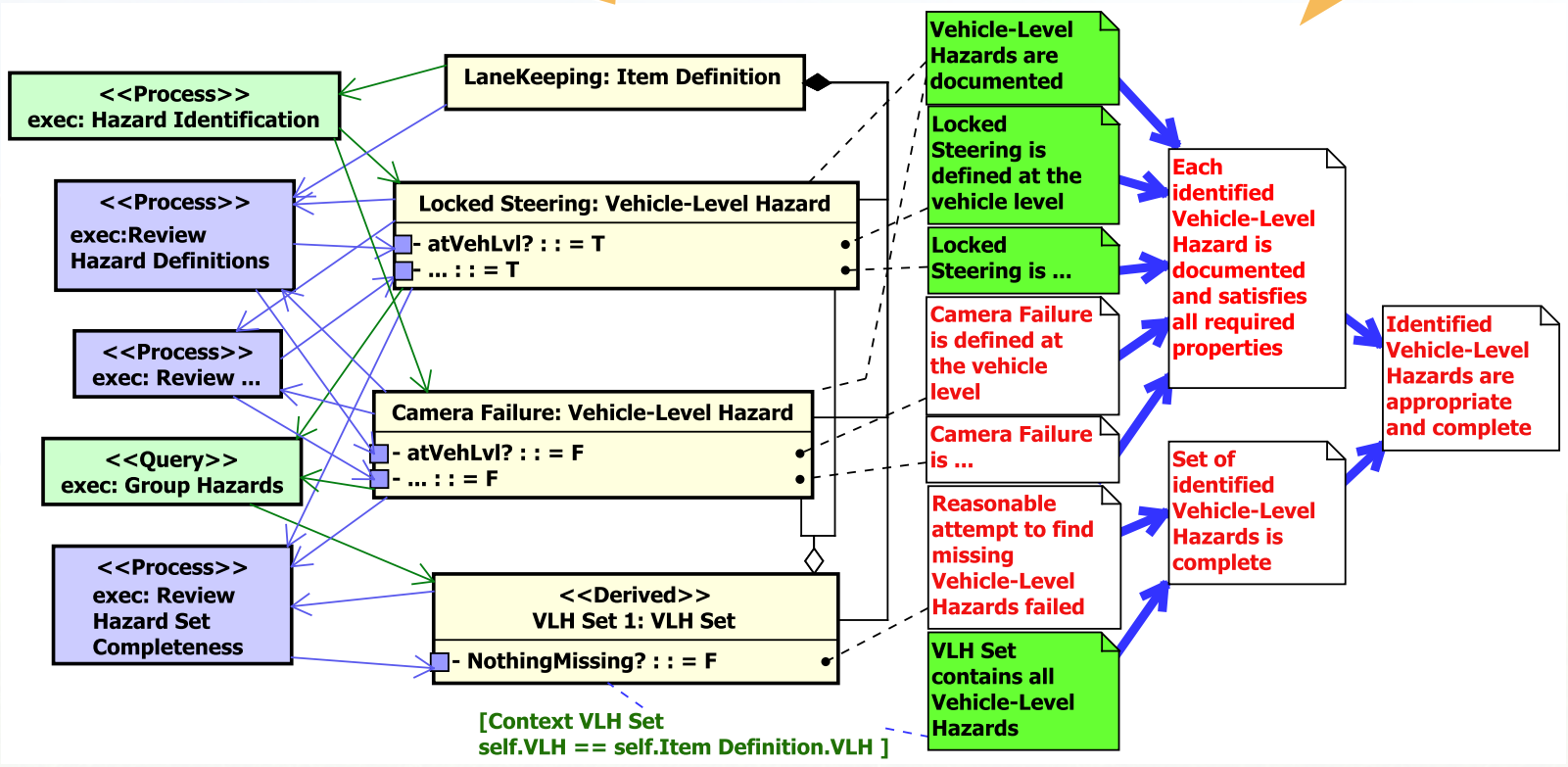


WF+ Example: Instance

Execution is documented in instances (evidence)

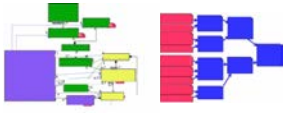
Instances can be in models, spreadsheets, Simulink, Medini Analyze, other tools

Argument in instances directly follows from template in metamodel –all we do is check constraints to see if they hold

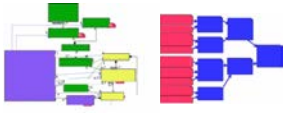


How We Use It

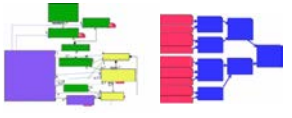
We have several processes & the related argument fragments modelled…



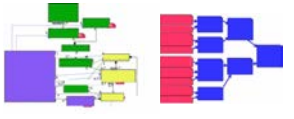
Process A



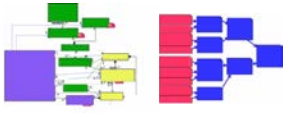
Process B



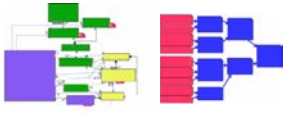
Process C



Process D



Process E



Process F

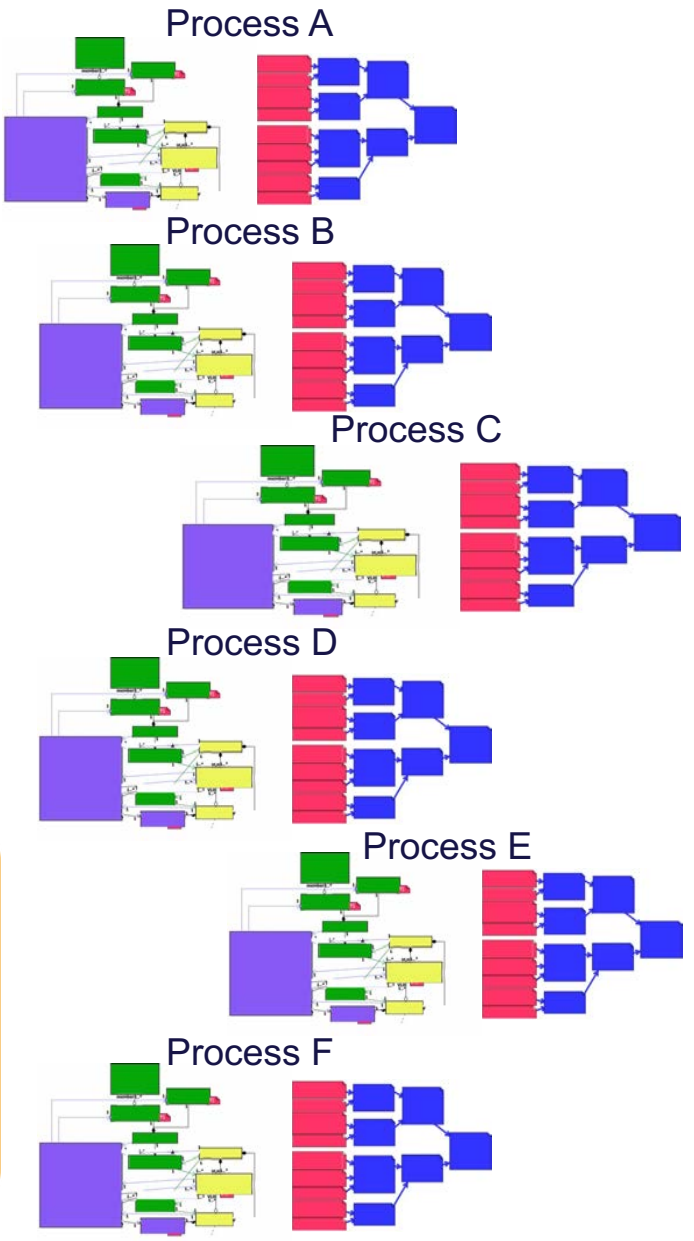
These fragments are tightly integrated & highly traceable

Later processes can build on or modify earlier ones

Each represents a step taken to ensure safety. Need to integrate the arguments to reveal overall safety argument/reasoning

Completing The Argument

Process
Review
Data
Terminal Claim
Derived Claim



Framework is still very generic, allowing for complete freedom in how they are integrated

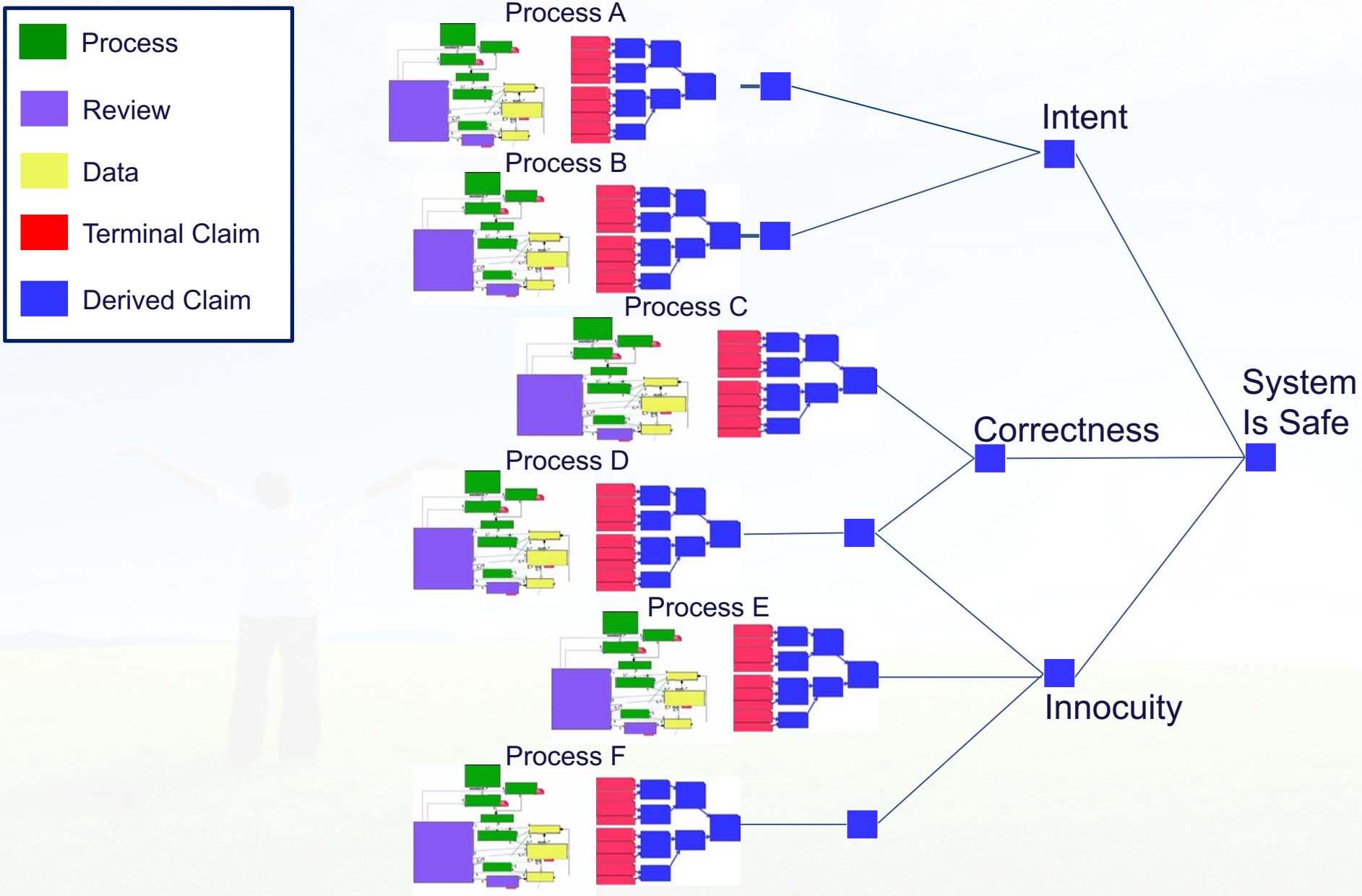
Currently have not implemented anything specific in terms of the approach to safety, e.g. what is necessary, how different tasks are related, etc.

We could (and are working on) using ideas such as the Overarching Properties [4,5] to achieve this

Important to realize that

1. The entire process is highly iterative
2. The individual processes are planned to deliver specific results

Completing The Argument



Creating Reusable And Understandable Arguments

A fundamental idea in our framework is that the **metamodel is the plan** and the **instance is the execution**

We found that including details of the system in the argument cluttered the argument and made reuse more difficult

Example 1

Detail on the mitigation of hazards belongs in the hazard analysis as evidence

The results, properties that were checked etc. can be recorded in the argument

and

Elements typically found in a GSN argument, such as context, assumptions and defeasible reasoning, we have moved back into local arguments (Toulmin style) between the evidence and the terminal claims in the evidence.

Example 2

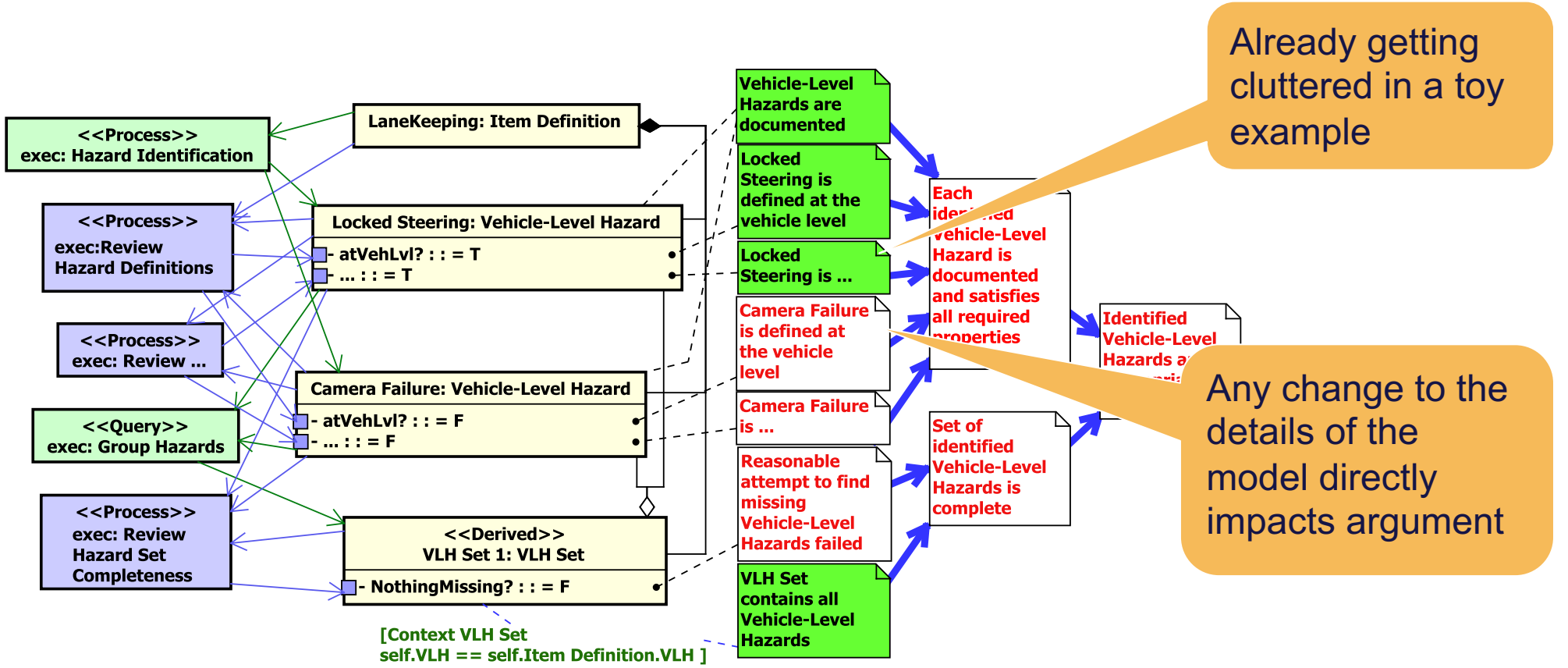
Plan for a test and its results should be in documentation

Aggregated results (i.e. every test passed, each was adequately planned, etc.) can go in the argument

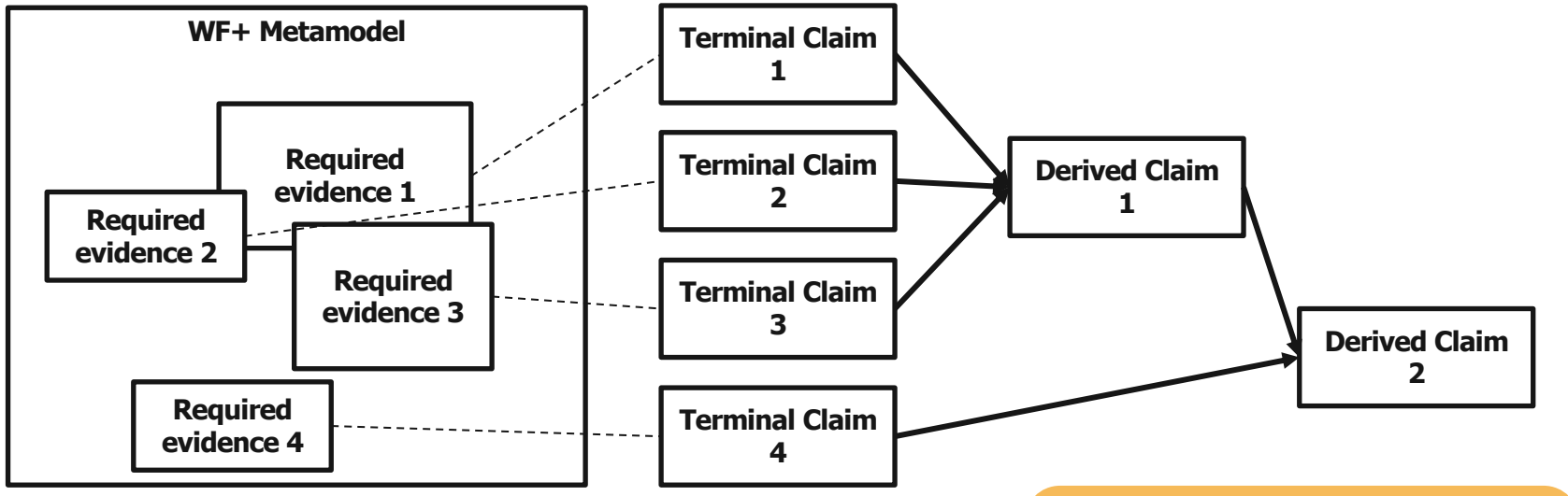
Creating Reusable And Understandable Arguments

A fundamental idea in our framework is that the **metamodel is the plan** and the **instance is the execution**

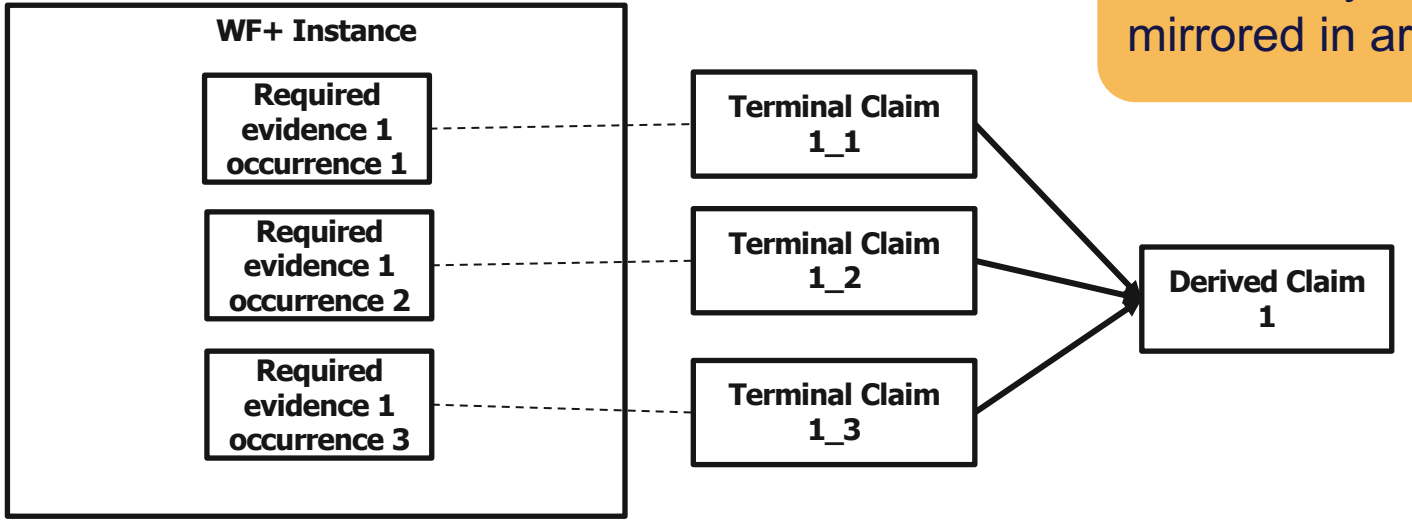
We found that including details of the system in the argument cluttered the argument and made reuse more difficult



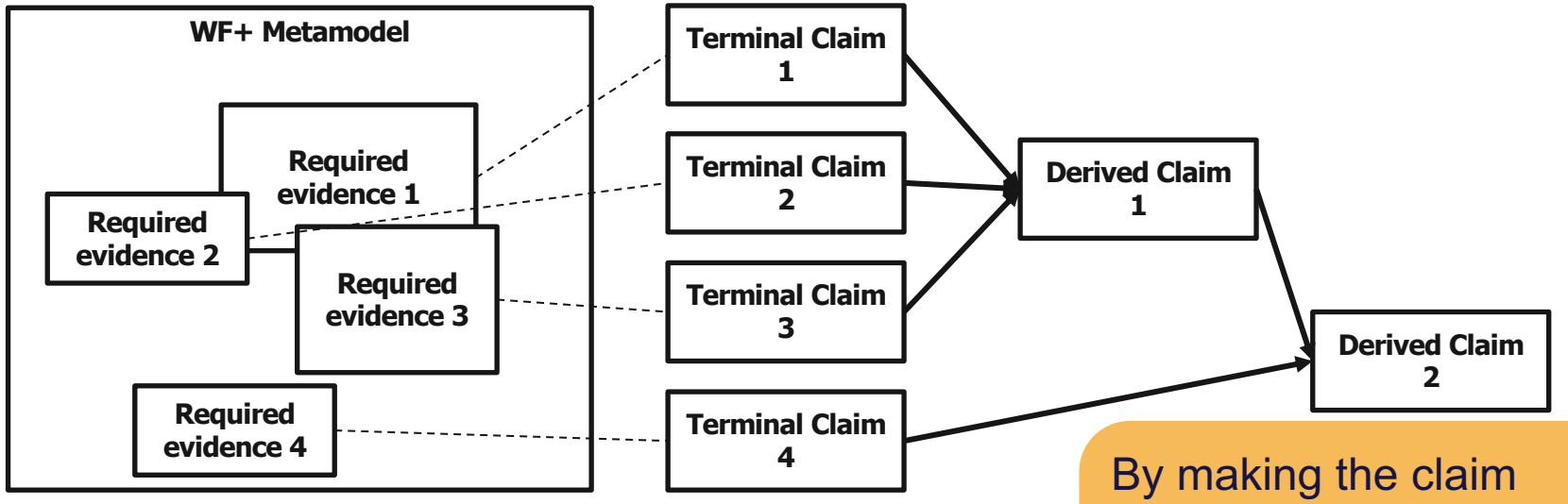
Creating Reusable And Understandable Arguments



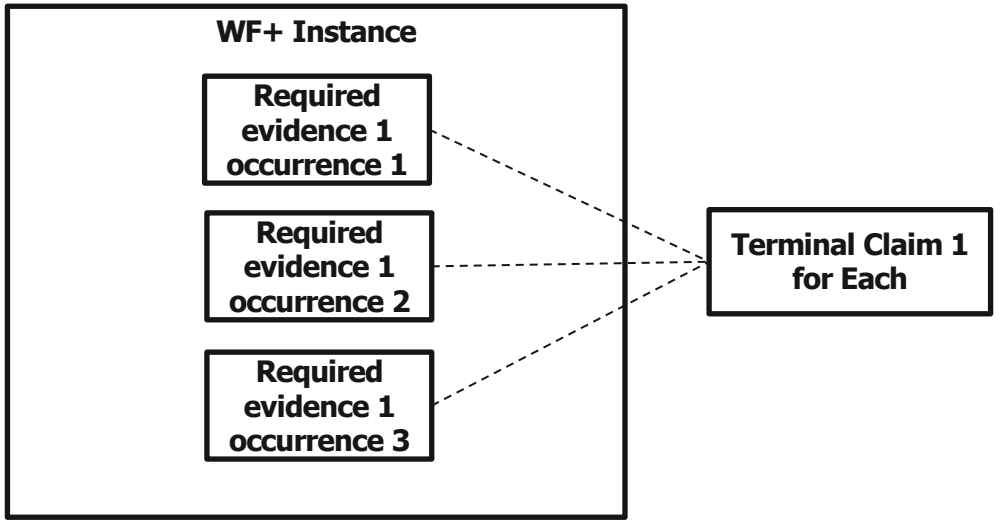
Complexity of instance/system is mirrored in argument



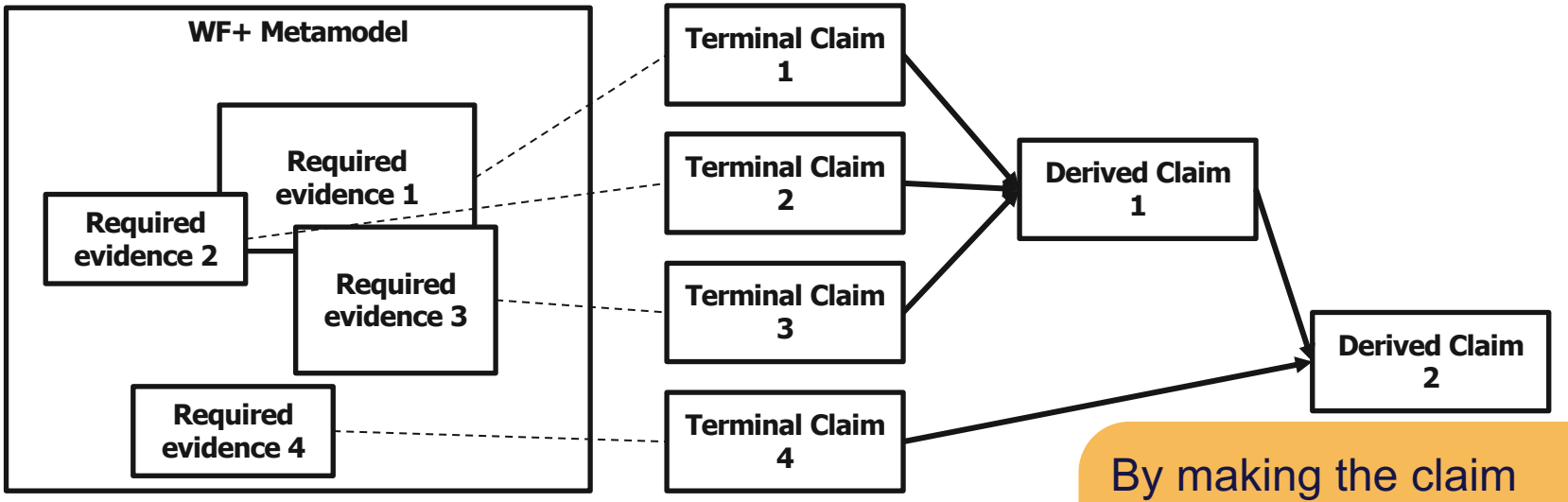
Creating Reusable And Understandable Arguments

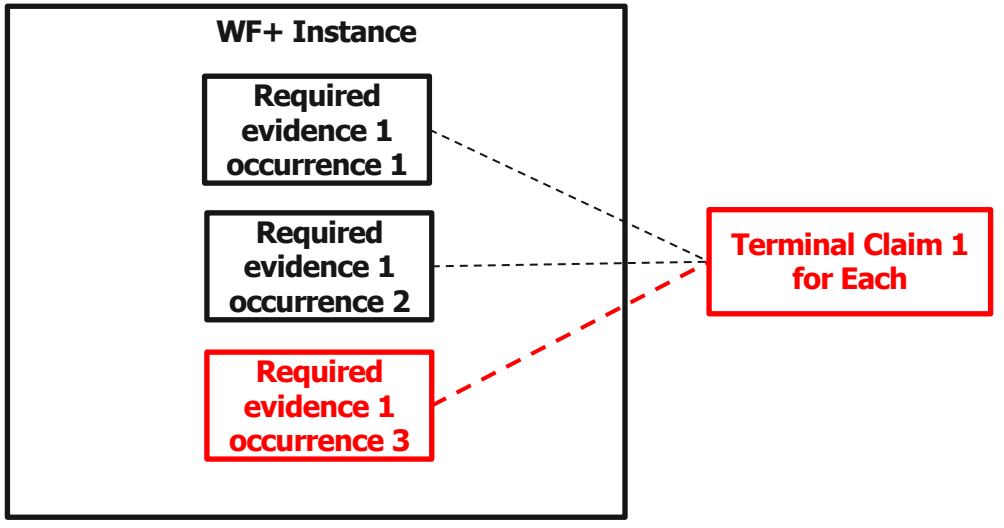


By making the claim over each occurrence in the instance the argument is more concise



Creating Reusable And Understandable Arguments





By making the claim over each occurrence in the instance the argument is more concise

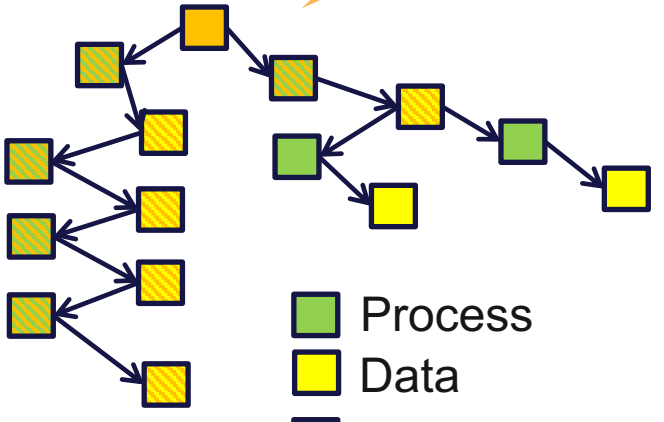
We do not actually lose anything – traceability allows us to understand the details that are now excluded from the argument

Effective & Efficient Change Impact Analysis

Absolutely essential in supporting incremental safety & security assurance – a primary capability of WF+ that will facilitate producing safe & secure OTASUs

Avoiding false positives is essential for efficiency, detailed traceability helps avoid false propagations

Crucial for any approach to avoid false negatives – detailed traceability helps ensure we do not miss anything



Process
Data
Initial change
Flagged by impact analysis

Engineers can be guided to processes producing/consuming impacted data to begin resolving impacts

McSCert

How WF+ Can Help With OTASUs

Supports scalability
Extensive traceability enabling model management, automation to handle variability, support for generating understandable, more reusable arguments

Supports increased confidence
Basing assurance arguments on detailed models with the necessary traceability can help give confidence required for remote updates

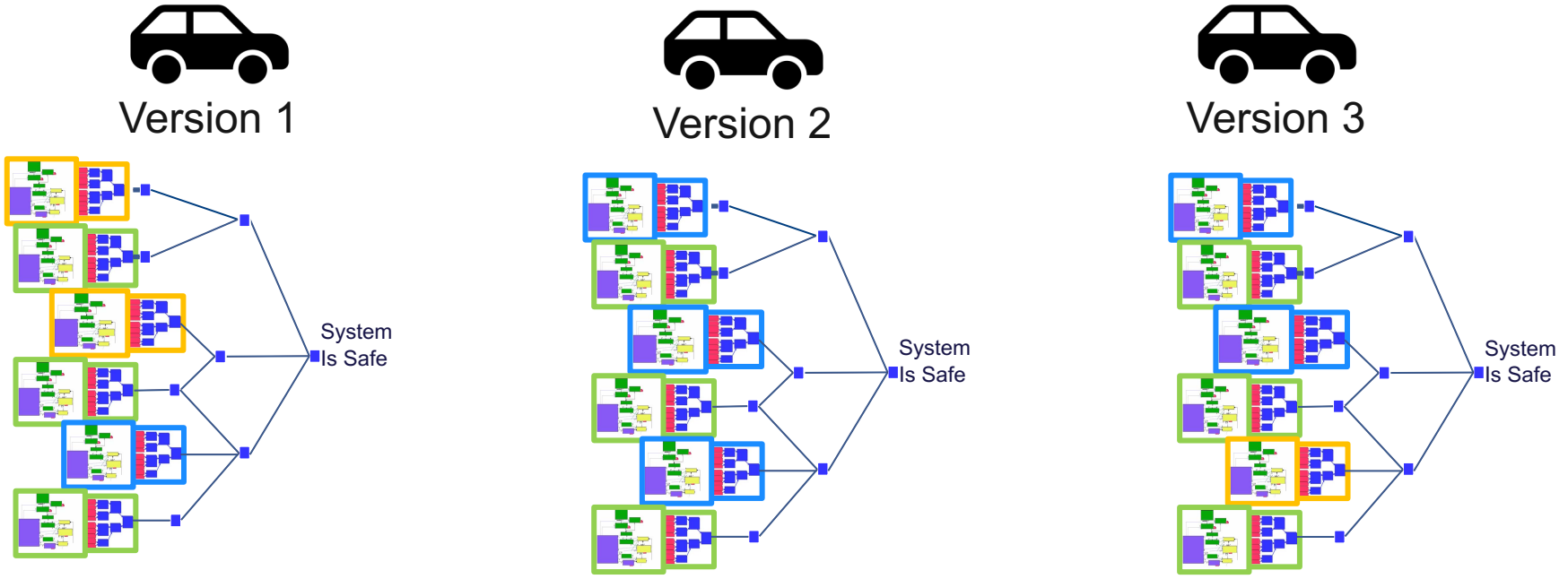
Capable of supporting product families
Traceability enables change impact analysis and incremental assurance required to efficiently assure safety of product families in the time available

What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Product family support

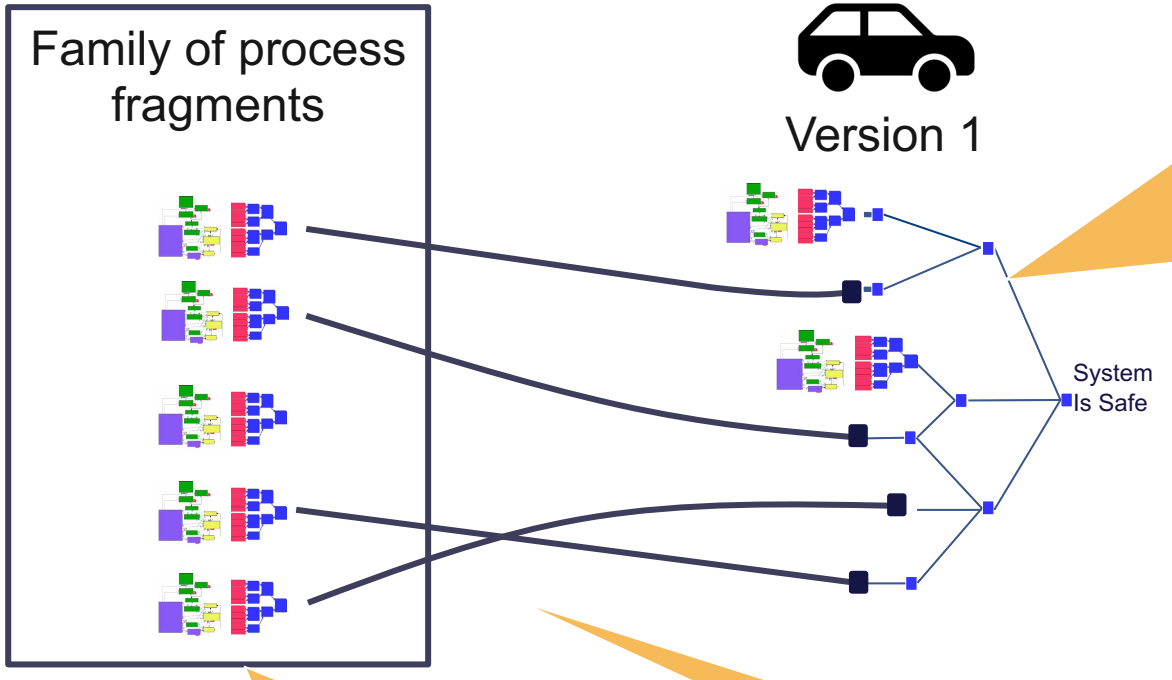
We need a way to track, aggregate and update reusable process/work product fragments between different versions of systems



What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Product family support



Need to generate argument for variants based on shared fragments. This involves significantly less work than manually duplicating and maintaining

Less duplicated effort by enabling reusable fragments

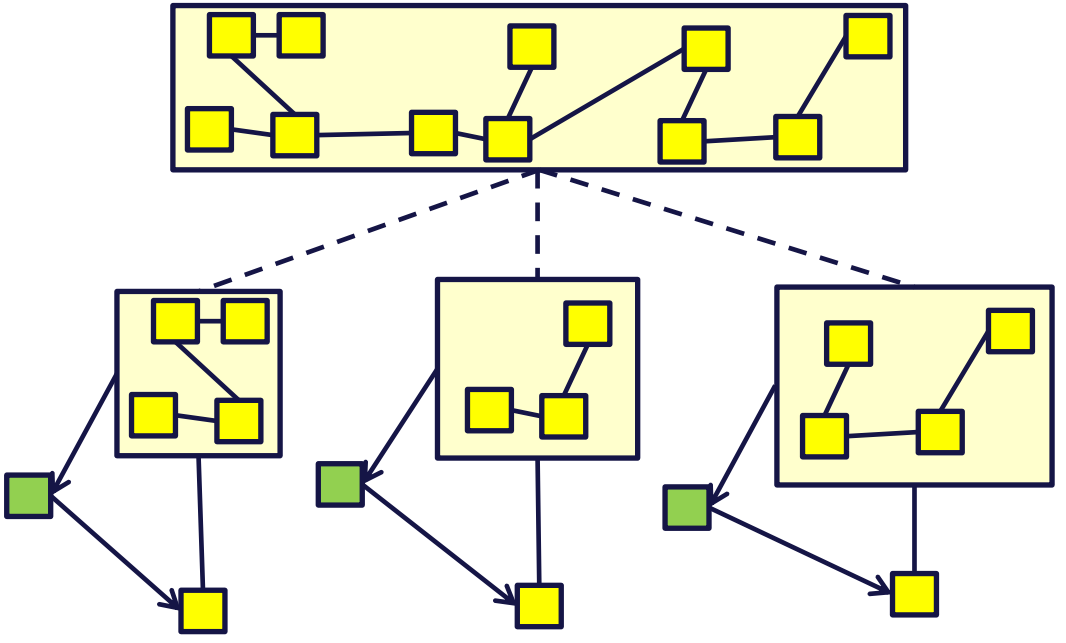
Need to support & track usage of reusable components so impact analysis, other queries can be traced to affected variants

What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Packaging data

Data must be processed in granular chunks to enable traceability for impact analysis & specific assurance arguments



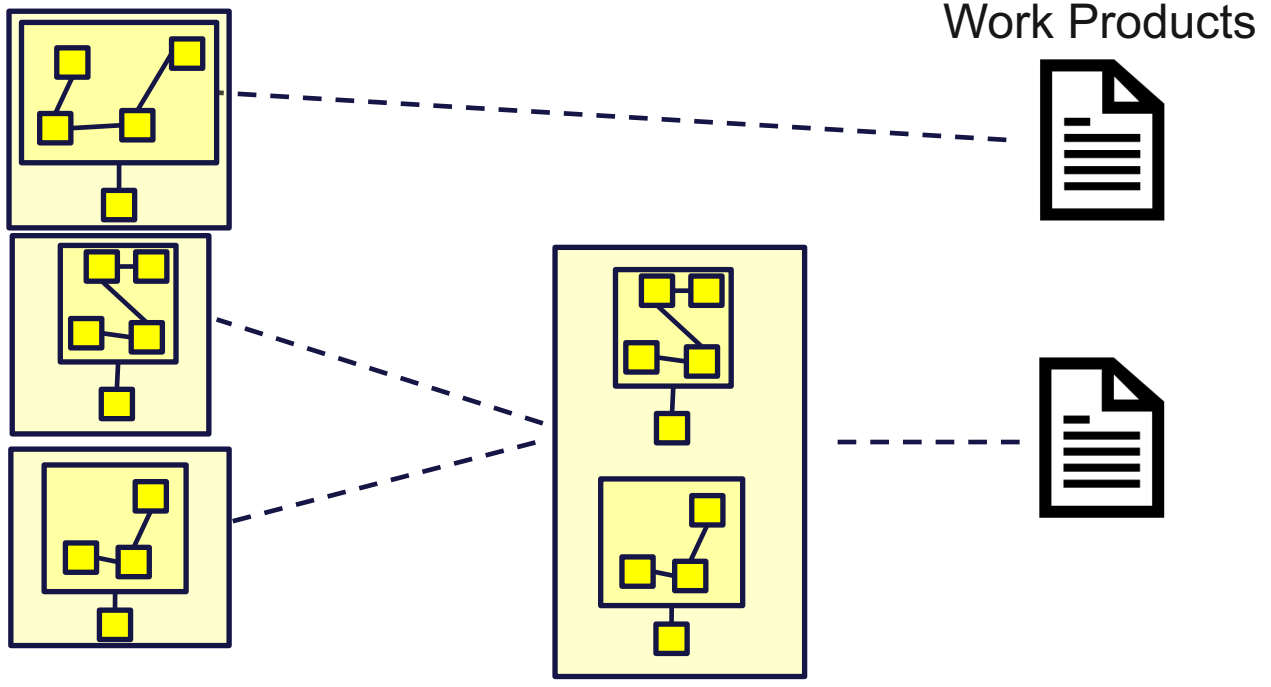
What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Packaging data

Data must be processed in granular chunks to enable traceability for impact analysis & specific assurance arguments

At some point it may need to be packaged/aggregated



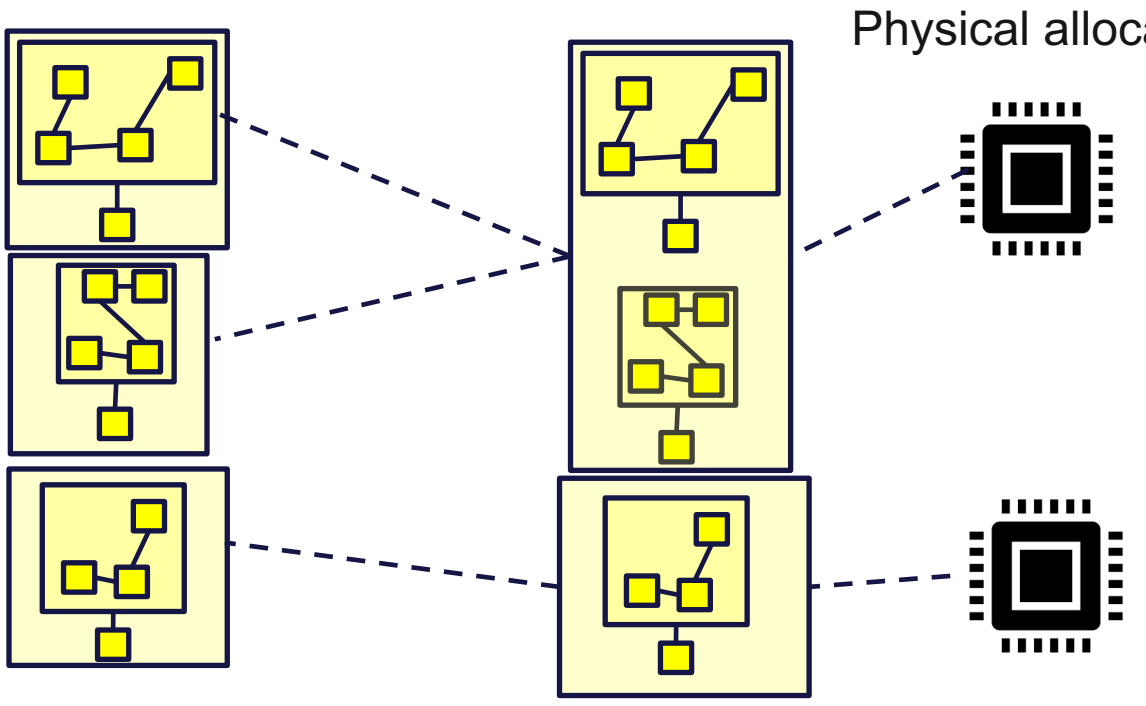
What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Packaging data

Data must be processed in granular chunks to enable traceability for impact analysis & specific assurance arguments

At some point it may need to be packaged/aggregated



This type of packaging will be important for analyzing vehicle variants & performing impact analysis

Some components will be OEM and some from suppliers. Important for OTASU to track allocation to components

What We Are Adding To WF+

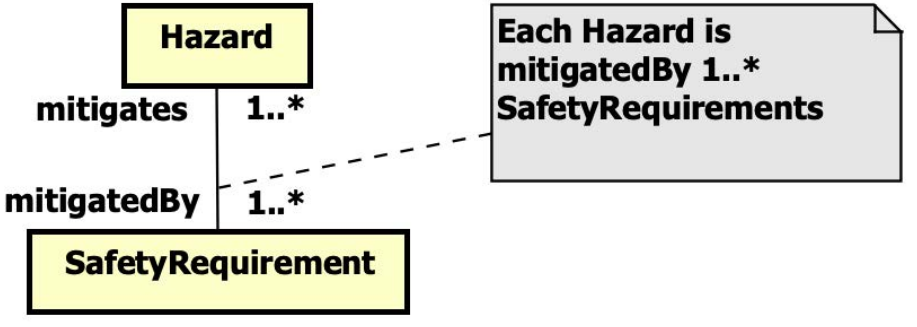
We are not quite there yet… Some things we are working on adding:

Method for (semi-automatic) argument generation

Having such frequent updates will all but require automation of aspects of argument generation

Reliable automation can also improve accuracy and confidence in arguments

Lots of 'low hanging fruit' to easily automate & reduce effort required



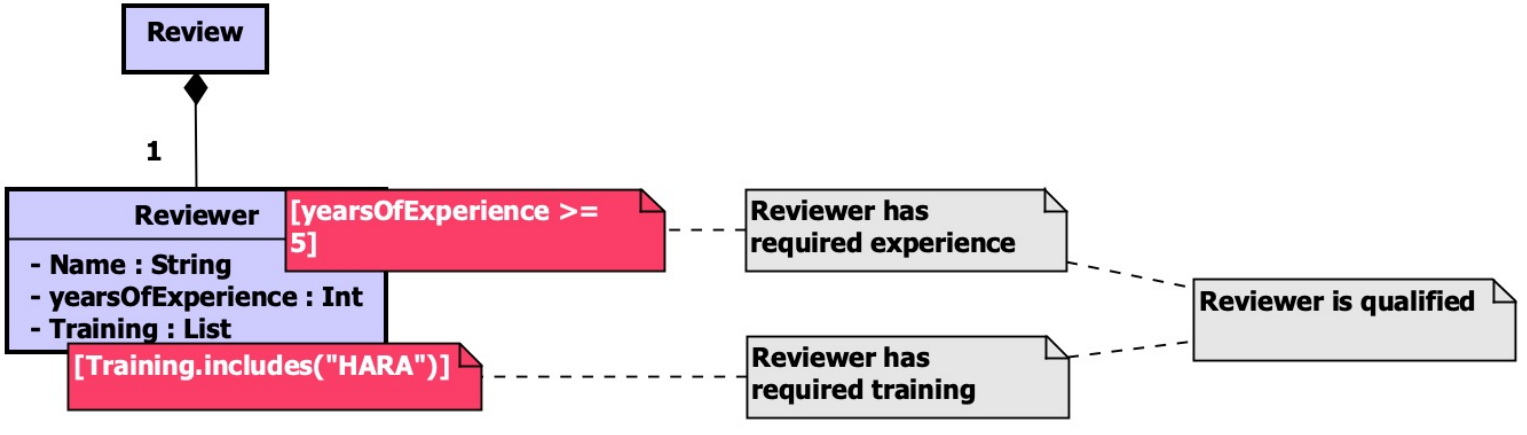
What We Are Adding To WF+

We are not quite there yet… Some things we are working on adding:

Method for (semi-automatic) argument generation

Process of developing metamodels is currently flexible

With a bit of regularity, we can enable automation of more argument generation steps, composition of arguments



Acknowledgements

This work reflects collaboration in a larger group

We thank everyone in this group who has collaborated to this approach re over-the-air software updates

Mehrnoosh Askarpour
Thomas Chiang
Spencer Deevy
Sahar Kokaly
Lucian Patcas
Galen Ressler
Horacio Hoyos Rodriguez
Ramesh S
Anthony Shenouda
Lindsay White

McSCert

References

[1] ISO 26262: Road vehicles – Functional safety. Int. Organization for Standardization, Geneva, Switzerland (2018)

[2] Annable, N., Chiang, T., Lawford, M., Paige, R.F., Wassyng, A.: Generating Assurance Cases using WorkFlow+ Models. In: SAFECOMP 2022, Munich, Germany, September 6–9, 2022, Proceedings. pp. 97–110. Springer (2022)

[3] Annable, N., Bayzat, A., Diskin, Z., Lawford, M., Paige, R., Wassyng, A.: Model-driven Safety of Autonomous Vehicles. In: proc. of CSER (2020)

[4] Holloway, C. Michael. Understanding the overarching properties. No. NF1676L-33745. 2019.

[5] Z. Daw, S. Beecher, M. Holloway and M. Graydon, "Overarching Properties as means of compliance: An industrial case study," 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 2021, pp. 1-10, doi: 10.1109/DASC52595.2021.9594298.