PROPERTY-DRIVEN CONTINUOUS ASSURANCE OF SOFTWARE DESIGNS

HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE (HCSS 2024)
Annapolis, Maryland

SRIVATSAN VARADARAJAN
HONEYWELL INTERNATIONAL
SRIVATSAN.VARADARAJAN@HONEYWELL.COM

May 6, 2024

Honeywell

Design for Certification (DesCert) Project

DARPA Automated Rapid Certification of Software (ARCOS) Program

TEAM:
DEVESH BHATT, HAO REN, ANITHA MURUGESAN, SRIVATSAN VARADARAJAN
SHANKAR NATARAJAN, MINYOUNG KIM,
MICHAEL ERNST

supported by

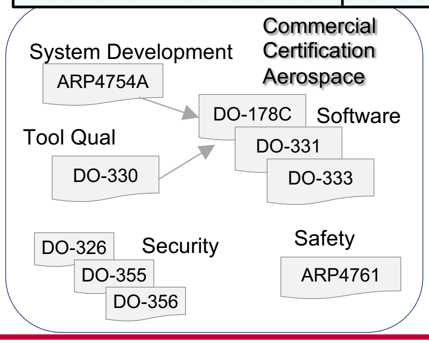
SRI International

UNIVERSITY *of* WASHINGTON

DARPA

WHY PROPERY DRIVEN CONTINUOUS ASSURANCE?

Traditional Means of Compliance



Artifacts are just the tip of the iceberg
A large part of <u>assurance</u> lies within the hidden activities that surround the artifact production
Hard to Judge: Quality of Compliance $\overset{?}{=}$ Degree of Confidence
Implicit Prescription Rationale to Designers vs Dearth of Design Insights for Regulators

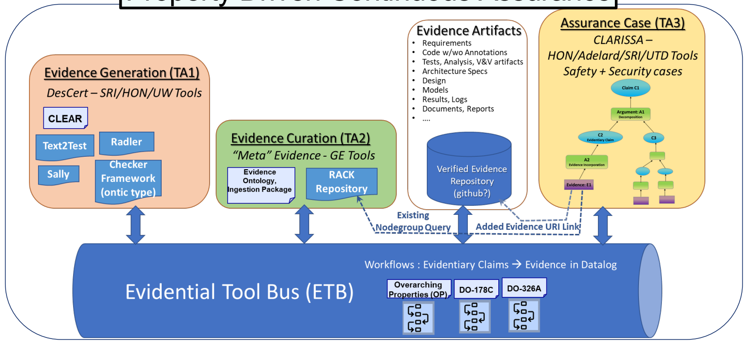
Process
Human Centric View
Development Metrics
Audits

People
Expertise, Competency
Domain knowledge
Credentials, Reputation

Governance
Oversight
Regulations

DesCert Vision:
Explicate Hidden Iceberg

Property Driven Continuous Assurance



Evidence Generation (TA1)
DesCert – SRI/HON/UW Tools

Evidence Curation (TA2)
"Meta" Evidence - GE Tools

Evidence Artifacts

- Requirements
- Code w/wo Annotations
- Tests, Analysis, V&V artifacts
- Architecture Specs
- Design
- Models
- Results, Logs
- Documents, Reports
- ….

Assurance Case (TA3)
CLARISSA – HON/Adelard/SRI/UTD Tools
Safety + Security cases

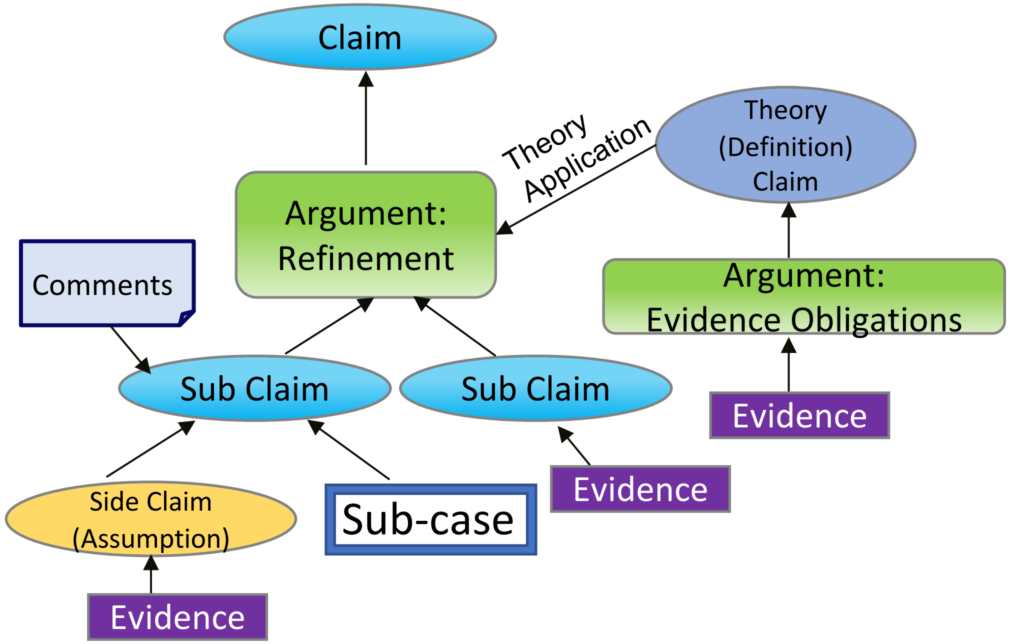
Evidential Tool Bus (ETB)

Workflows : Evidentiary Claims → Evidence in Datalog

CLEAR
Text2Test
Radler
Sally
Checker Framework (ontic type)
Evidence Ontology, Ingestion Package
RACK Repository
Verified Evidence Repository (github?)
Claim C1
Argument: A1 Decomposition
C2 Evidentiary Claim
C3
A2 Evidence Incorporation
Evidence: E1
Existing Nodegroup Query
Added Evidence URI Link
Overarching Properties (OP)
DO-178C
DO-326A

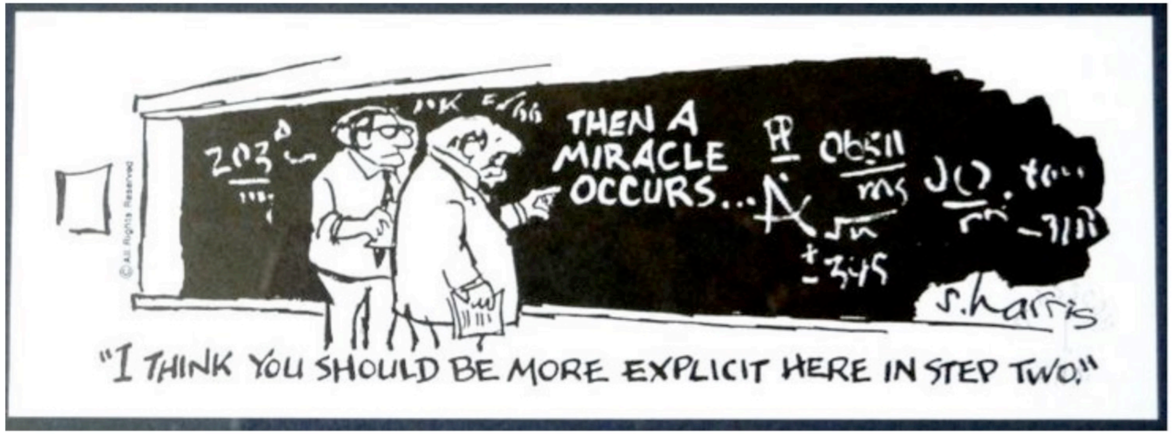
- ➢ Making assurance <u>more</u> ***objective (i.e. property-driven)***, ***evidence based, explicit rationale***, ***automated***, and ***systematic***
- ➢ Making assurance <u>less</u> ***process/compliance-driven, prescriptive*** and ***implicit rationale***
- ➢ <u>Less</u> ***documentary artifact production*** & <u>More</u> ***rigorous digital engineering***
- ➢ Encourage ***development, regulatory innovations*** that ***lowers cost, time and errors***
- ➢ ***Incremental*** Certification of changes, ***Continuous Assurances*** for CI/CD Pipelines

SOFTWARE DESIGN FOR ***EFFICIENT ARGUMENTS***



Evidence-based Assurance

- *Arguments:* parent-claims *refinements* to sub-claims, & side-claims backed by supporting *evidence* that demonstrates that *software faithfully implements the intended behavior*
- *Repeatable argumentation* backed by reusable assurance sub-cases called *Theories* with own *supporting evidentiary obligations*
- Good argument should make it easy to identify and *fix fallacious reasoning* steps

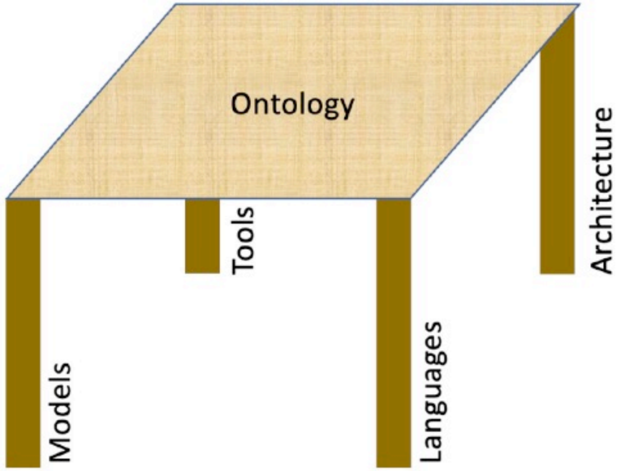


Making Arguments Efficient

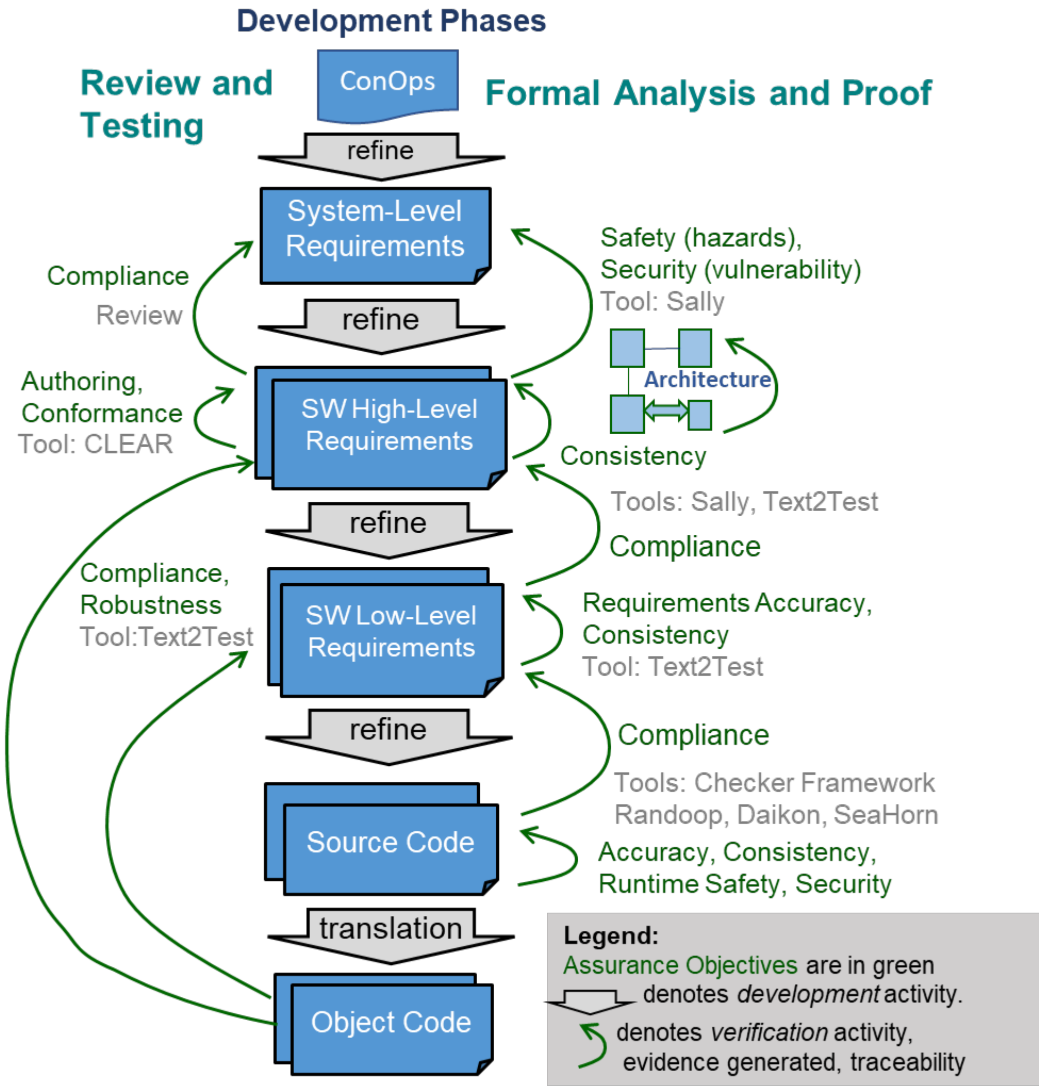
- *Efficient argument* is one whose flaws, if any, can be easily identified by a skeptic
- *Good designs* expands the falsification space for the skeptic
- *Efficiency* is measured by the *amortized* cost of falsification e.g. *Partitioned* RTOS, using *memory-safe* hardware and *type-safe* languages
- *Inefficient arguments* due to imprecise claims, flawed/irrelevant evidence, complex arguments, unfalsifiable assumptions, invalid reasoning….

EVIDENCE GENERATION TOOLS FOR ASSURANCE

Software Design for Efficient Argument



- Precise Claims based on *Ontologies*
- Valid *models* and assumptions
- Reusable design *tools*, ”Safe” Languages
- *Architectural* separation of concerns
- Rigorous chains of reasoning and evidence



Development Phases

Review and Testing

Formal Analysis and Proof

ConOps

refine

System-Level Requirements

refine

SW High-Level Requirements

refine

SW Low-Level Requirements

refine

Source Code

translation

Object Code

Compliance
Review

Authoring,
Conformance
Tool: CLEAR

Compliance,
Robustness
Tool:Text2Test

Safety (hazards),
Security (vulnerability)
Tool: Sally

Architecture

Consistency

Tools: Sally, Text2Test
Compliance

Requirements Accuracy,
Consistency
Tool: Text2Test

Compliance
Tools: Checker Framework
Randoop, Daikon, SeaHorn

Accuracy, Consistency,
Runtime Safety, Security

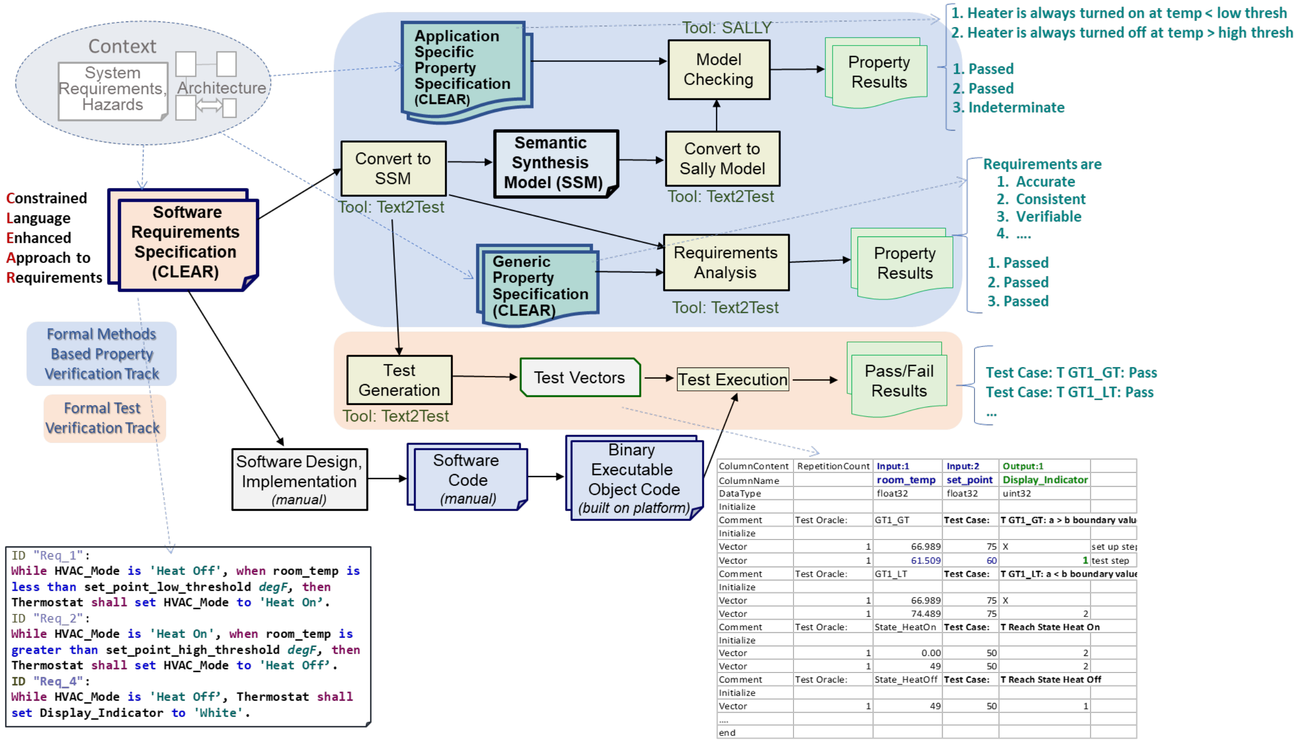
Legend:
Assurance Objectives are in green
denotes *development* activity.
denotes *verification* activity,
evidence generated, traceability

PROPERTY DRIVEN SOFTWARE ASSURANCE

	Requirements: Specific, individual functional behaviors the system shall do	Properties: What the system ought to do/not to do		
		Safety	Liveness	Invariants
Purpose	Specification for design and implementation	Something bad will never happen	Something good will eventually (bounded time) occur	Desired system constraints
Verification Approach	Testing	Model Checking	Testing and Model Checking	Testing and Model Checking
Exemplars	If the remaining battery power is critically low, the system shall initiate emergency landing	Once the system is in insufficient battery state, then system shall never transition back to normal battery state	The system shall reach its destination in normal battery state (within x secs)	Emergency landing is always initiated when/after systems reached insufficient battery state

Derive Tests to execute on Implementation

Model-check Properties on Requirements Model (proxy for checking on implementation)



Context
System Requirements, Hazards
Architecture
Constrained **L**anguage **E**nhanced **A**pproach to **R**equirements
Software Requirements Specification (CLEAR)
Formal Methods Based Property Verification Track
Formal Test Verification Track
Application Specific Property Specification (CLEAR)
Tool: SALLY
Model Checking
Property Results
1. Heater is always turned on at temp < low thresh
2. Heater is always turned off at temp > high thresh
1. Passed
2. Passed
3. Indeterminate
Convert to SSM
Semantic Synthesis Model (SSM)
Convert to Sally Model
Tool: Text2Test
Tool: Text2Test
Generic Property Specification (CLEAR)
Requirements Analysis
Tool: Text2Test
Property Results
Requirements are
1. Accurate
2. Consistent
3. Verifiable
4. ....
1. Passed
2. Passed
3. Passed
Test Generation
Tool: Text2Test
Test Vectors
Test Execution
Pass/Fail Results
Test Case: T GT1_GT: Pass
Test Case: T GT1_LT: Pass
...
Software Design, Implementation (manual)
Software Code (manual)
Binary Executable Object Code (built on platform)

```
ID "Req_1":
While HVAC_Mode is 'Heat Off', when room_temp is
less than set_point_low_threshold degF, then
Thermostat shall set HVAC_Mode to 'Heat On’.
ID "Req_2":
While HVAC_Mode is 'Heat On', when room_temp is
greater than set_point_high_threshold degF, then
Thermostat shall set HVAC_Mode to 'Heat Off’.
ID "Req_4":
While HVAC_Mode is 'Heat Off’, Thermostat shall
set Display_Indicator to 'White'.
```

ColumnContent	RepetitionCount	Input:1	Input:2	Output:1	
ColumnName		room_temp	set_point	Display_Indicator	
DataType		float32	float32	uint32	
Initialize					
Comment	Test Oracle:	GT1_GT	**Test Case:**	**T GT1_GT: a > b boundary valu**	
Initialize					
Vector	1	66.989	75	X	set up step
Vector	1	61.509	60	1	test step
Comment	Test Oracle:	GT1_LT	**Test Case:**	**T GT1_LT: a < b boundary value**	
Initialize					
Vector	1	66.989	75	X	
Vector	1	74.489	75	2	
Comment	Test Oracle:	State_HeatOn	**Test Case:**	**T Reach State Heat On**	
Initialize					
Vector	1	0.00	50	2	
Vector	1	49	50	2	
Comment	Test Oracle:	State_HeatOff	**Test Case:**	**T Reach State Heat Off**	
Initialize					
Vector	1	49	50	1	
....					
end					

Capture both Requirements and Properties

- Properties have broader scope and context than individual requirements
- Capturing both increases confidence in the validity of requirements
- Property holds on the aggregated behavior of individualized requirements

NASA Formal Methods (NFM) symposium 2022 paper: “*Requirements-Driven Model Checking and Test Generation for Comprehensive Verification*”

Belt and Suspender Hybrid Verification Approach: Testing & Formal Methods

SMART REQUIREMENTS ENGINEERING USING GEN-AI

Need to address Gen-AI issues:

☑ Lack of system and domain understanding.

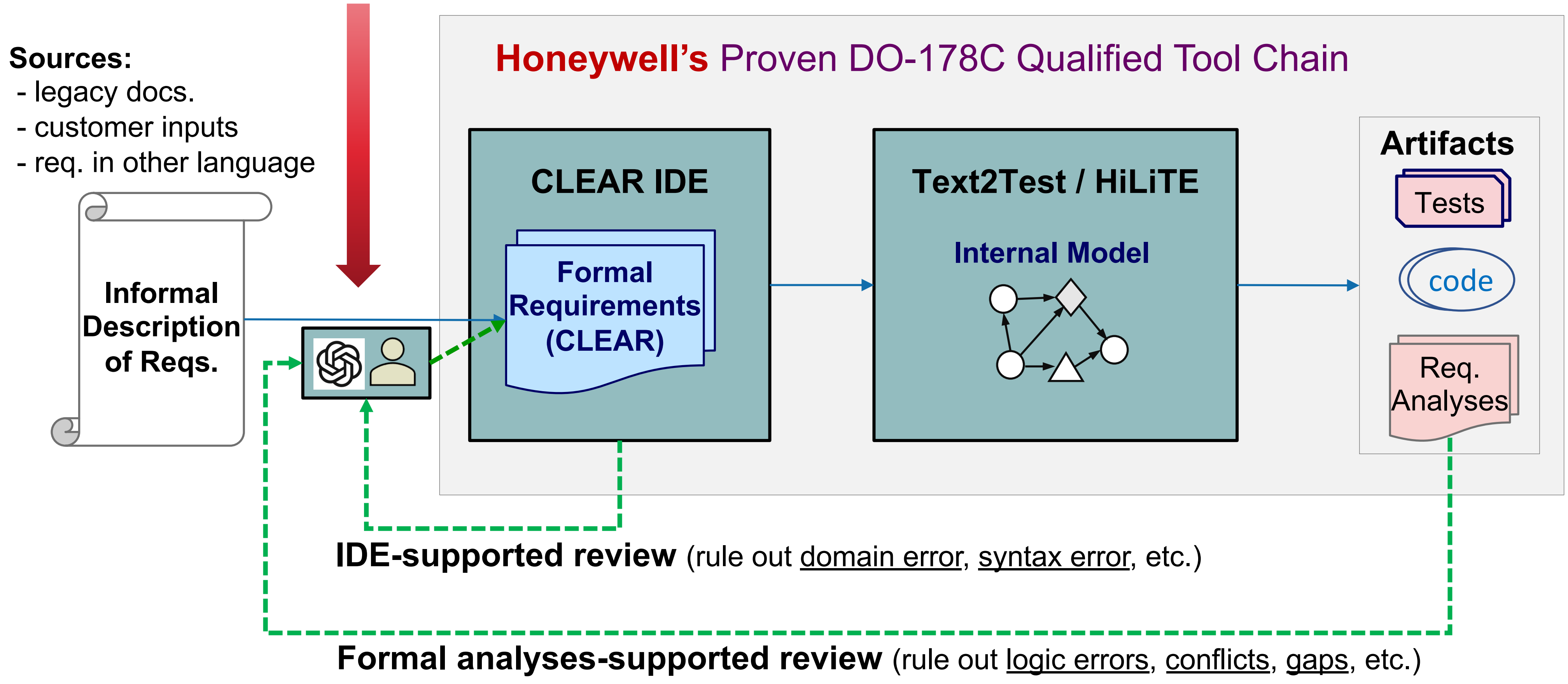
☑ Outputs are not always reliable.

☑ Need human review.

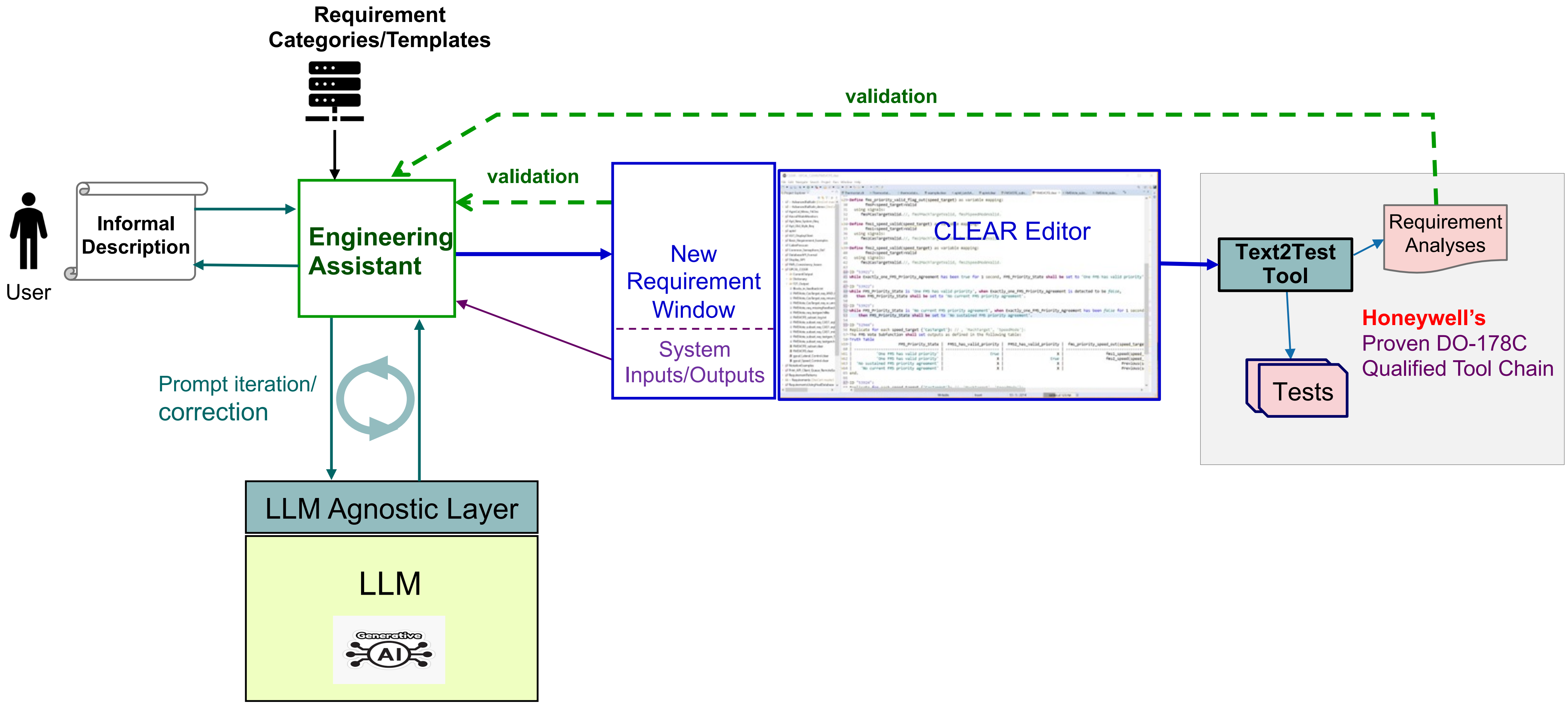
☑ Cost and usability

Gen-AI Assisted Req. Creation

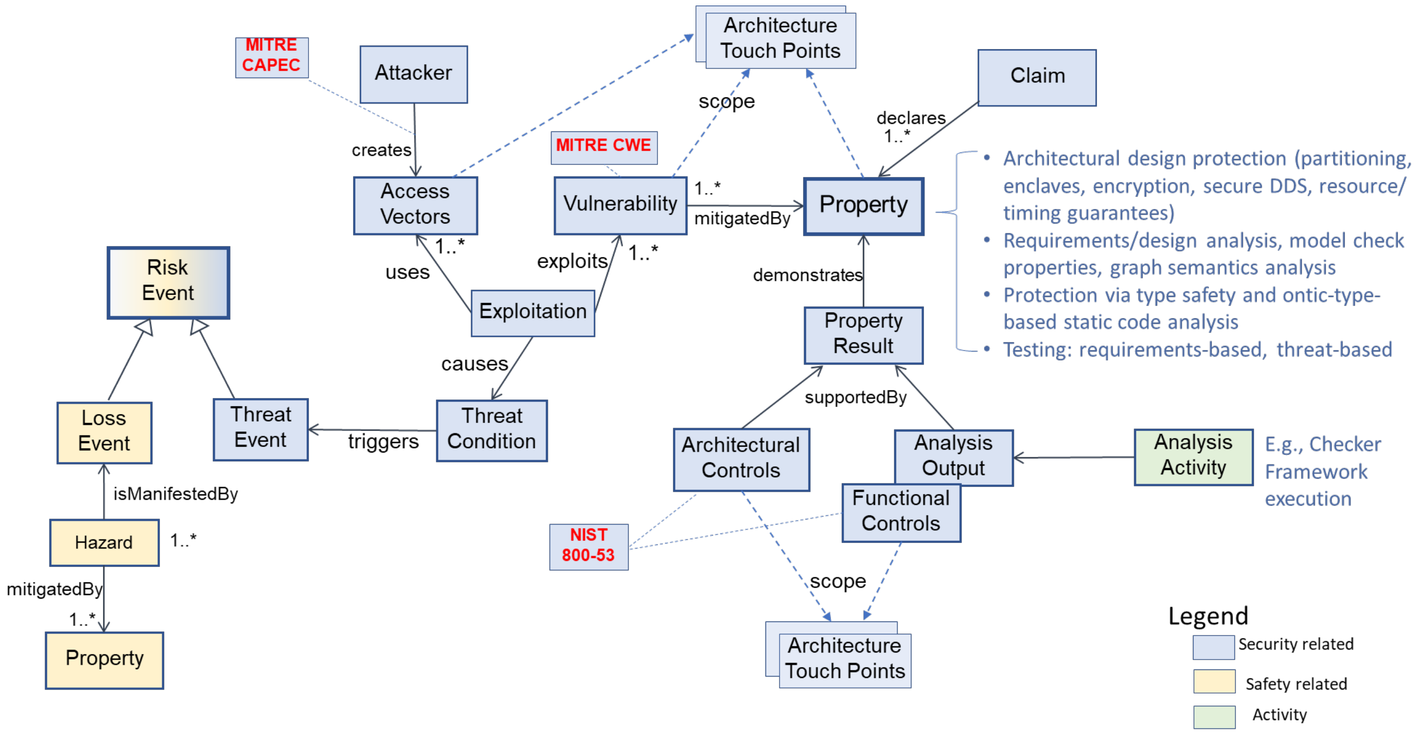
Low-cost Few-Shot Learning of Sys./Domain


Sources:
- legacy docs.
- customer inputs
- req. in other language
Informal Description of Reqs.
Honeywell's Proven DO-178C Qualified Tool Chain
CLEAR IDE
Formal Requirements (CLEAR)
Text2Test / HiLiTE
Internal Model
Artifacts
Tests
code
Req. Analyses
IDE-supported review (rule out domain error, syntax error, etc.)
Formal analyses-supported review (rule out logic errors, conflicts, gaps, etc.)

SMART REQUIREMENTS ENGINEERING USING GEN-AI



ONTOLOGIES SYTEMATIZATION: SAFETY & SECURITY

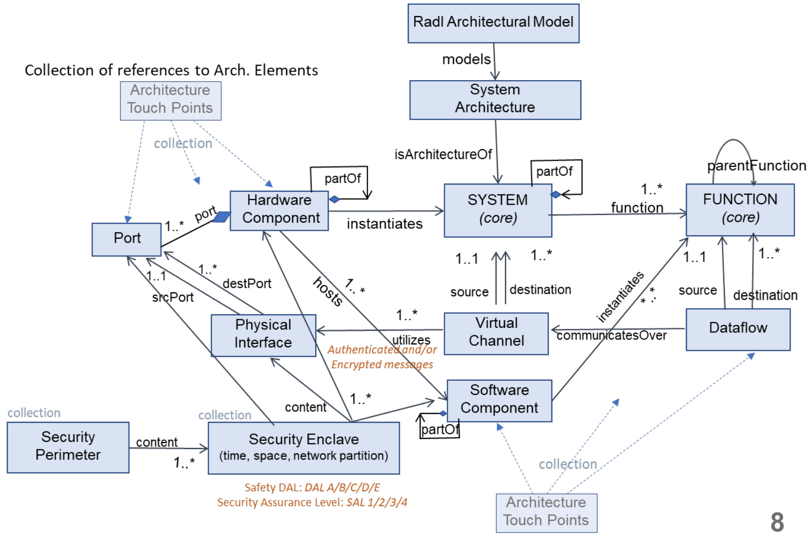


- Architectural design protection (partitioning, enclaves, encryption, secure DDS, resource/ timing guarantees)
- Requirements/design analysis, model check properties, graph semantics analysis
- Protection via type safety and ontic-type-based static code analysis
- Testing: requirements-based, threat-based

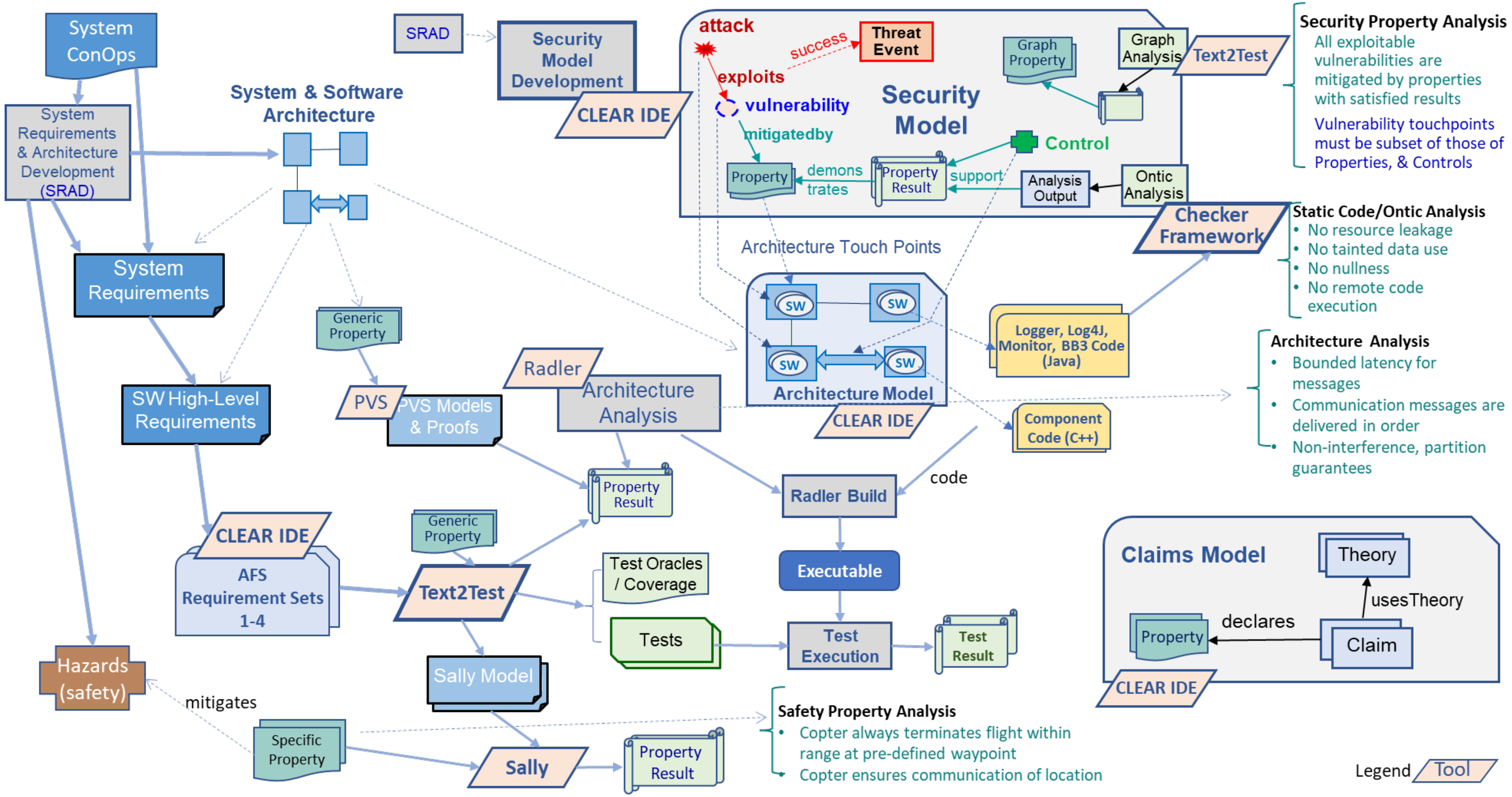
Ontological categories for *modeling* of:

1. *Threats*: Bypassing access control/ input validation, race conditions, timing attacks, phishing, privilege escalation, malicious code, remote code execution
2. *Vulnerabilities*: Null dereference, SQL injection, Buffer overflow
3. *Controls*: Physical security, Access control, Monitoring, Reporting, Authentication
4. *Risk/loss events*: Loss of Confidentiality, Integrity, Availability, Safety,..
5. *Architecture/Touch (entry) Points*: Sensors, Actuators, Communication channels, Files, Hardware, Software Components etc.

Threat	Entry Point	Risk	Mitigation
Malicious Code	Build Process	Failure, Unauthorized Access	Radler Certified Build/Attestation
Malicious Inside Actor	Untrusted Code	DoS, Failure, exfiltration/infiltration	Radler Security Enclaves
Loss of Information Integrity	Tampering	Failure	Radler Security Enclaves
Loss of Comm. integrity	Communication layer	Infiltration, Exfiltration, Jamming	Radler/SROS2 protections
Access Control Violation	Architecture	Failure, Unauthorized Access	Radler config., Ontic analysis
Bad/Unexpected Input	Unchecked input ports	Failure/Remote Code Execution	Ontic Type Analysis



END-TO-END, TOP-DOWN EVIDENCE GENERATION



Security Property Analysis

All exploitable vulnerabilities are mitigated by properties with satisfied results

Vulnerability touchpoints must be subset of those of Properties, & Controls

Static Code/Ontic Analysis

- No resource leakage
- No tainted data use
- No nullness
- No remote code execution

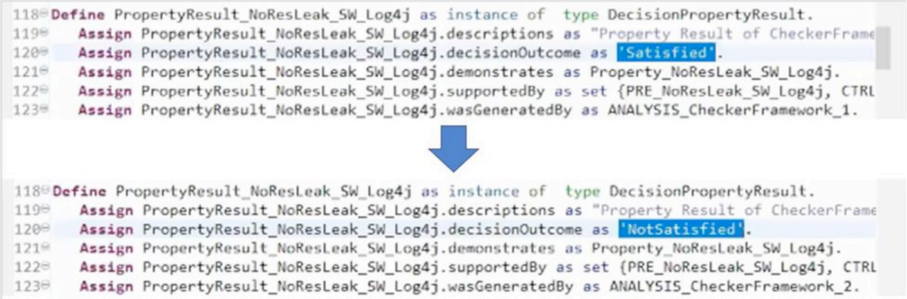
Architecture Analysis

- Bounded latency for messages
- Communication messages are delivered in order
- Non-interference, partition guarantees

Safety Property Analysis

- Copter always terminates flight within range at pre-defined waypoint
- Copter ensures communication of location

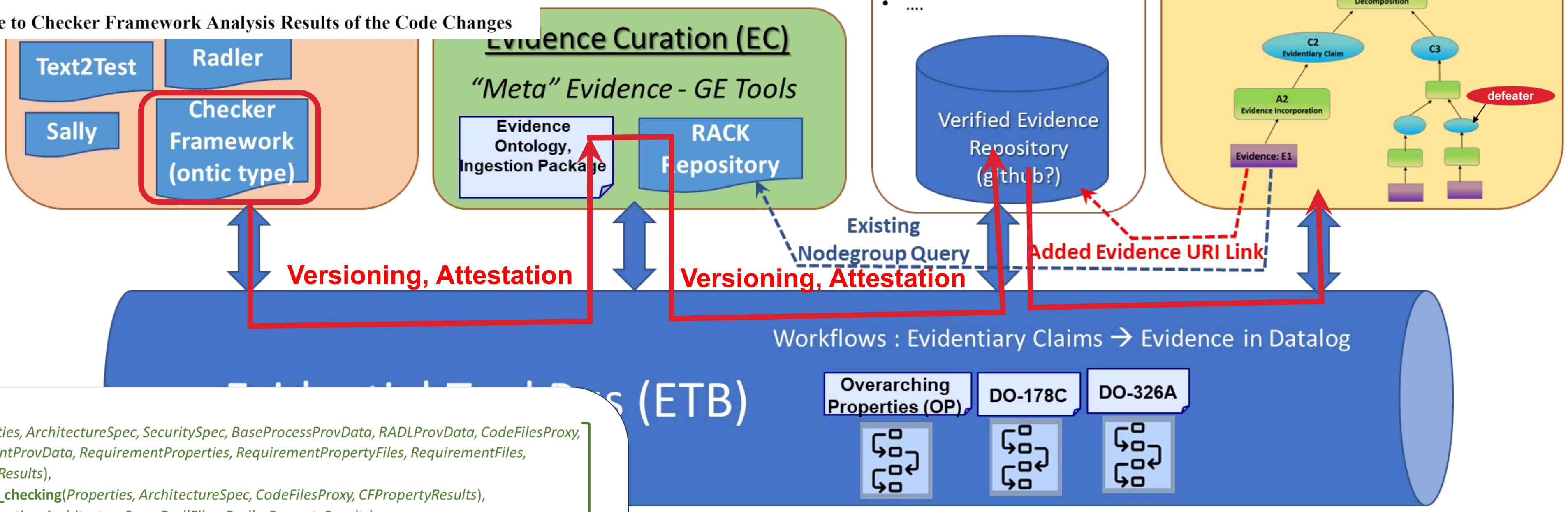
EVIDENCE INTEGRATED FOR CONTINUOUS ASSURANCE



```
Define PropertyResult_NoResLeak_SW_Log4j as instance of  type DecisionPropertyResult.
   Assign PropertyResult_NoResLeak_SW_Log4j.descriptions as "Property Result of CheckerFrame
   Assign PropertyResult_NoResLeak_SW_Log4j.decisionOutcome as 'Satisfied'.
   Assign PropertyResult_NoResLeak_SW_Log4j.demonstrates as Property_NoResLeak_SW_Log4j.
   Assign PropertyResult_NoResLeak_SW_Log4j.supportedBy as set {PRE_NoResLeak_SW_Log4j, CTRL
   Assign PropertyResult_NoResLeak_SW_Log4j.wasGeneratedBy as ANALYSIS_CheckerFramework_1.
```

```
Define PropertyResult_NoResLeak_SW_Log4j as instance of  type DecisionPropertyResult.
   Assign PropertyResult_NoResLeak_SW_Log4j.descriptions as "Property Result of CheckerFrame
   Assign PropertyResult_NoResLeak_SW_Log4j.decisionOutcome as 'NotSatisfied'.
   Assign PropertyResult_NoResLeak_SW_Log4j.demonstrates as Property_NoResLeak_SW_Log4j.
   Assign PropertyResult_NoResLeak_SW_Log4j.supportedBy as set {PRE_NoResLeak_SW_Log4j, CTRL
   Assign PropertyResult_NoResLeak_SW_Log4j.wasGeneratedBy as ANALYSIS_CheckerFramework_2.
```

Figure 44: Scenario 2 update to Checker Framework Analysis Results of the Code Changes



Text2Test
Radler
Sally
Checker Framework (ontic type)
Evidence Curation (EC)
"Meta" Evidence - GE Tools
Evidence Ontology, Ingestion Package
RACK Repository
Evidence Artifacts
Verified Evidence Repository (github?)
Assurance Generation (AG)
Claim C1
Argument: A1 Decomposition
C2 Evidentiary Claim
C3
A2 Evidence Incorporation
Evidence: E1
defeater
Versioning, Attestation
Existing Nodegroup Query
Versioning, Attestation
Added Evidence URI Link
Workflows : Evidentiary Claims → Evidence in Datalog
Evidential Tool Bus (ETB)
Overarching Properties (OP)
DO-178C
DO-326A

Evidence Artifacts

- Requirements
- Code w/wo Annotations
- Tests, Analysis, V&V artifacts
- Architecture Specs
- Design
- Models
- Results, Logs
- Documents, Reports
- ....

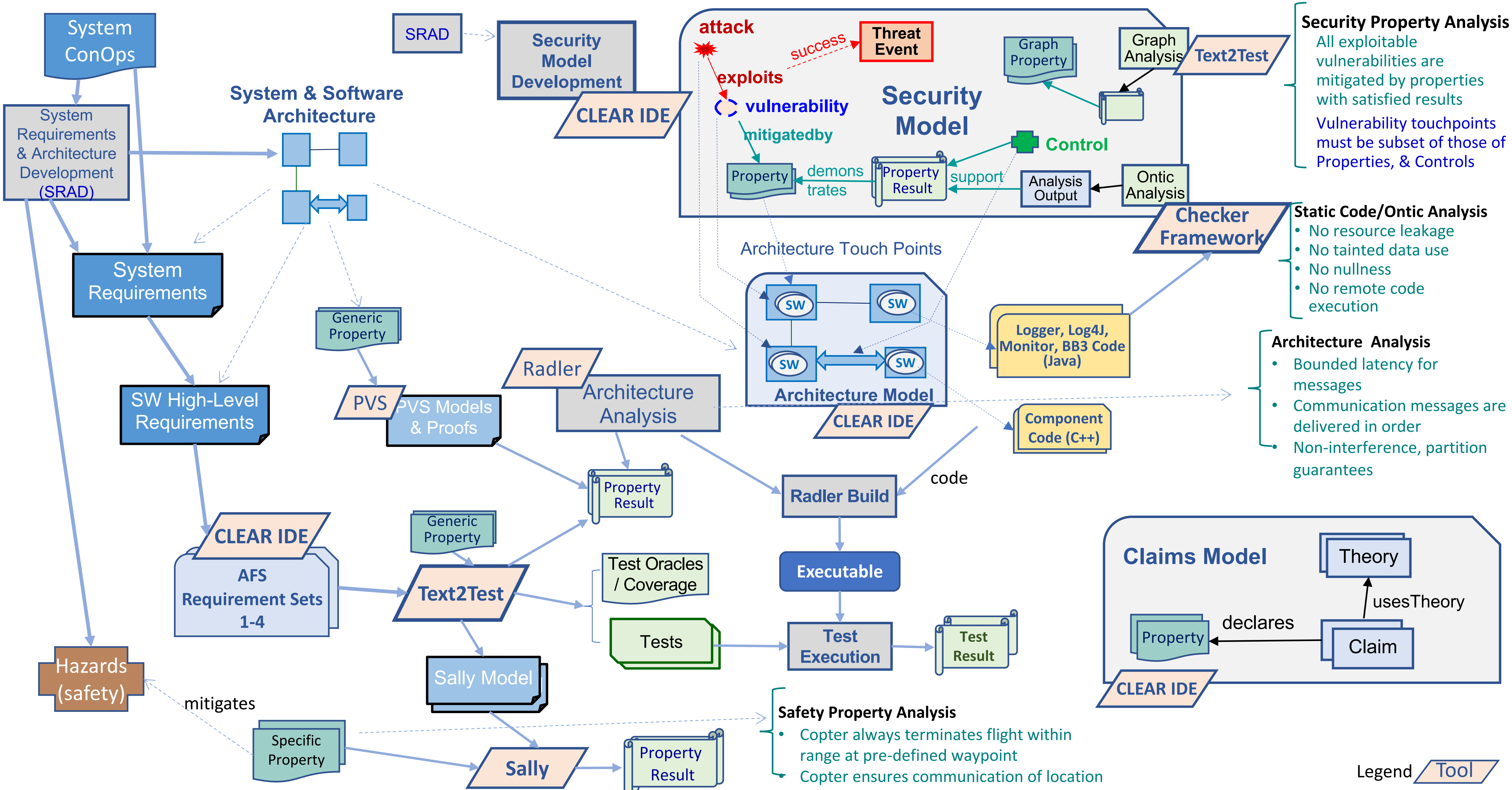
Assurance Generation (AG)

CLARISSA – HON/Adelard/SRI/UTD Tools Safety + Security cases

```
demo_workflow(Input) :-
    system_modeling(Properties, ArchitectureSpec, SecuritySpec, BaseProcessProvData, RADLProvData, CodeFilesProxy,
        RadlFiles, RequirementProvData, RequirementProperties, RequirementPropertyFiles, RequirementFiles,
        ArchControlPropertyResults),
    checkerFramework_type_checking(Properties, ArchitectureSpec, CodeFilesProxy, CFPropertyResults),
    radler_radl_analysis(Properties, ArchitectureSpec, RadlFiles, RadlerPropertyResults),
    text2Test_requirement_analysis(ArchitectureSpec, BaseProcessProvData, RequirementProperties,              EG
        RequirementProvData, RequirementPropertyFiles, RequirementFiles, ReqAnalysisPropertyResults),
    securityAnalysis_and_ingestion_creation(Properties, ArchitectureSpec, SecuritySpec, BaseProcessProvData,
        RADLProvData, RequirementProvData, RequirementProperties, ArchControlPropertyResults, CFPropertyResults,
        RadlerPropertyResults, ReqAnalysisPropertyResults, IngestionPackageFiles, IngestionPackageManifest),
    rack_ingestion(IngestionPackageFiles, IngestionPackageManifest, SuccessOrError, ErrorMsg, Handle, Diff), } EC
    update_asce_evidence_dnr(Diff, AsceFile),
    detect_property_violation(Diff, PropertyViolated),                AG
    update_asce_with_defeaters(PropertyViolated, AsceFile).
```

THANK YOU

Honeywell



Security Property Analysis

All exploitable vulnerabilities are mitigated by properties with satisfied results

Vulnerability touchpoints must be subset of those of Properties, & Controls

Static Code/Ontic Analysis

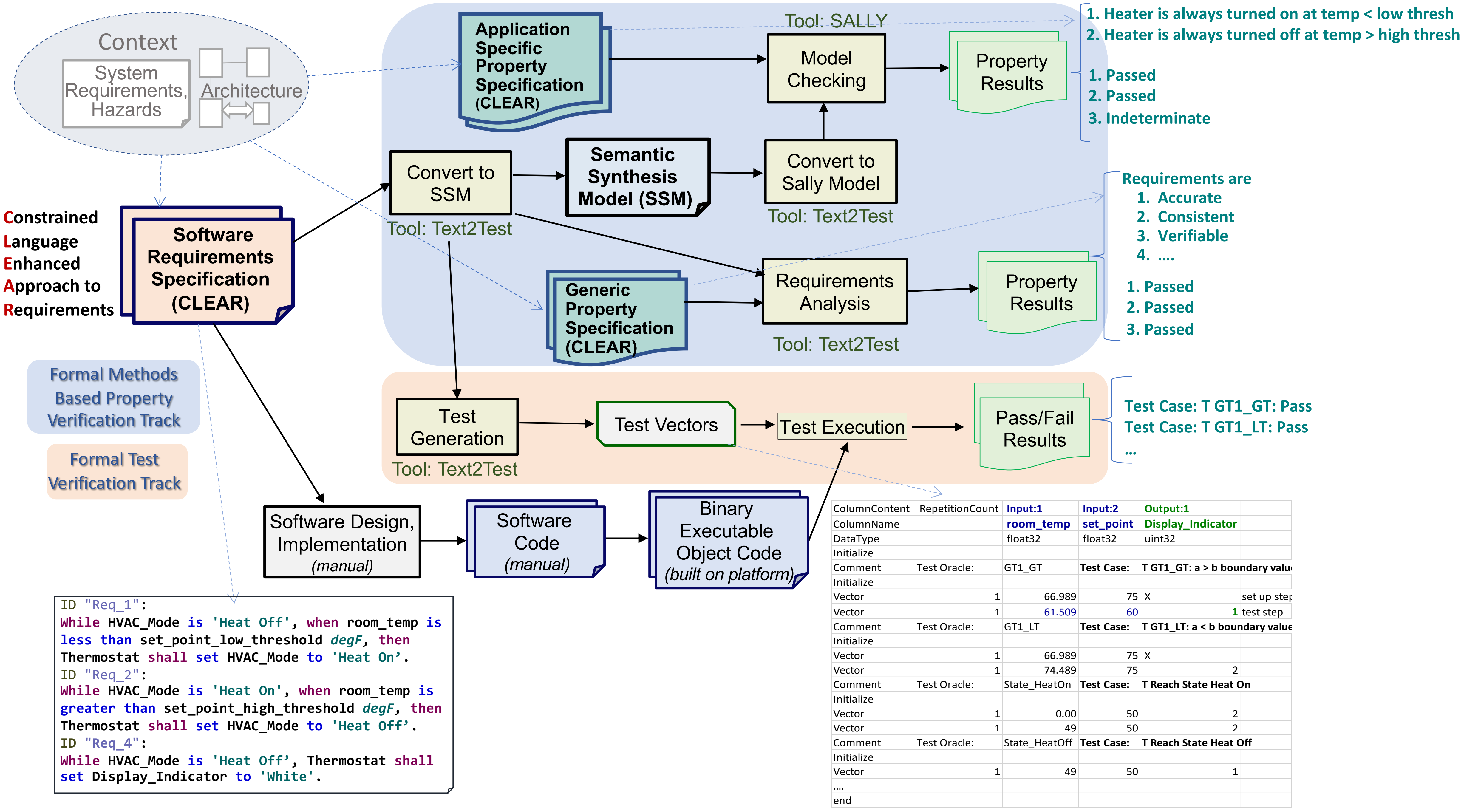
- No resource leakage
- No tainted data use
- No nullness
- No remote code execution

Architecture Analysis

- Bounded latency for messages
- Communication messages are delivered in order
- Non-interference, partition guarantees

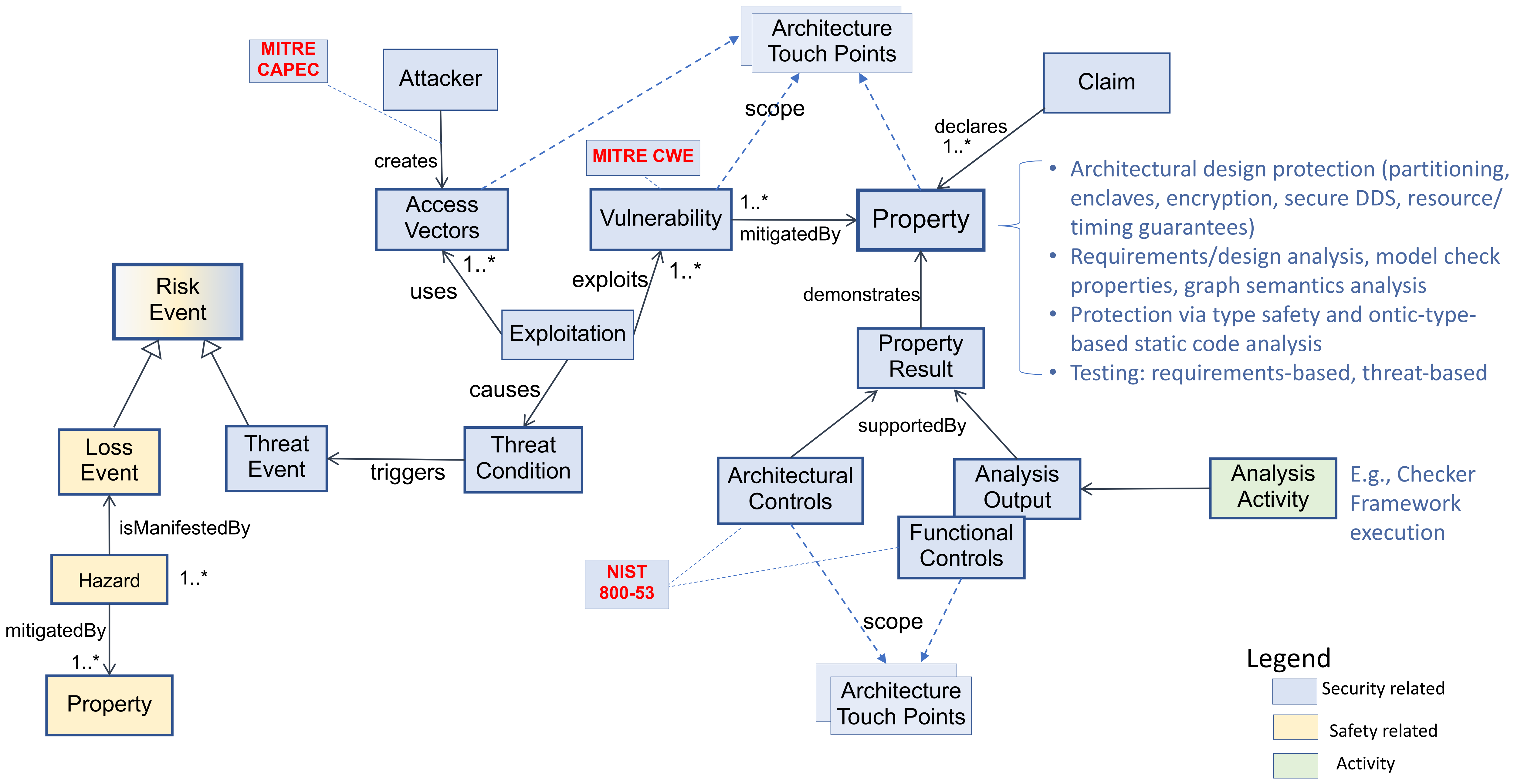
Safety Property Analysis

- Copter always terminates flight within range at pre-defined waypoint
- Copter ensures communication of location


Context
System Requirements, Hazards
Architecture
Constrained Language Enhanced Approach to Requirements
Software Requirements Specification (CLEAR)
Formal Methods Based Property Verification Track
Formal Test Verification Track
Application Specific Property Specification (CLEAR)
Tool: SALLY
Model Checking
Property Results
1. Heater is always turned on at temp < low thresh
2. Heater is always turned off at temp > high thresh
1. Passed
2. Passed
3. Indeterminate
Convert to SSM
Tool: Text2Test
Semantic Synthesis Model (SSM)
Convert to Sally Model
Tool: Text2Test
Requirements are
1. Accurate
2. Consistent
3. Verifiable
4. ....
Generic Property Specification (CLEAR)
Requirements Analysis
Tool: Text2Test
Property Results
1. Passed
2. Passed
3. Passed
Test Generation
Tool: Text2Test
Test Vectors
Test Execution
Pass/Fail Results
Test Case: T GT1_GT: Pass
Test Case: T GT1_LT: Pass
…
Software Design, Implementation (manual)
Software Code (manual)
Binary Executable Object Code (built on platform)

```
ID "Req_1":
While HVAC_Mode is 'Heat Off', when room_temp is
less than set_point_low_threshold degF, then
Thermostat shall set HVAC_Mode to 'Heat On’.
ID "Req_2":
While HVAC_Mode is 'Heat On', when room_temp is
greater than set_point_high_threshold degF, then
Thermostat shall set HVAC_Mode to 'Heat Off’.
ID "Req_4":
While HVAC_Mode is 'Heat Off’, Thermostat shall
set Display_Indicator to 'White'.
```

ColumnContent	RepetitionCount	Input:1	Input:2	Output:1	
ColumnName		room_temp	set_point	Display_Indicator	
DataType		float32	float32	uint32	
Initialize					
Comment	Test Oracle:	GT1_GT	**Test Case:**	**T GT1_GT: a > b boundary valu**	
Initialize					
Vector	1	66.989	75	X	set up step
Vector	1	61.509	60	1	test step
Comment	Test Oracle:	GT1_LT	**Test Case:**	**T GT1_LT: a < b boundary value**	
Initialize					
Vector	1	66.989	75	X	
Vector	1	74.489	75	2	
Comment	Test Oracle:	State_HeatOn	**Test Case:**	**T Reach State Heat On**	
Initialize					
Vector	1	0.00	50	2	
Vector	1	49	50	2	
Comment	Test Oracle:	State_HeatOff	**Test Case:**	**T Reach State Heat Off**	
Initialize					
Vector	1	49	50	1	
....					
end					

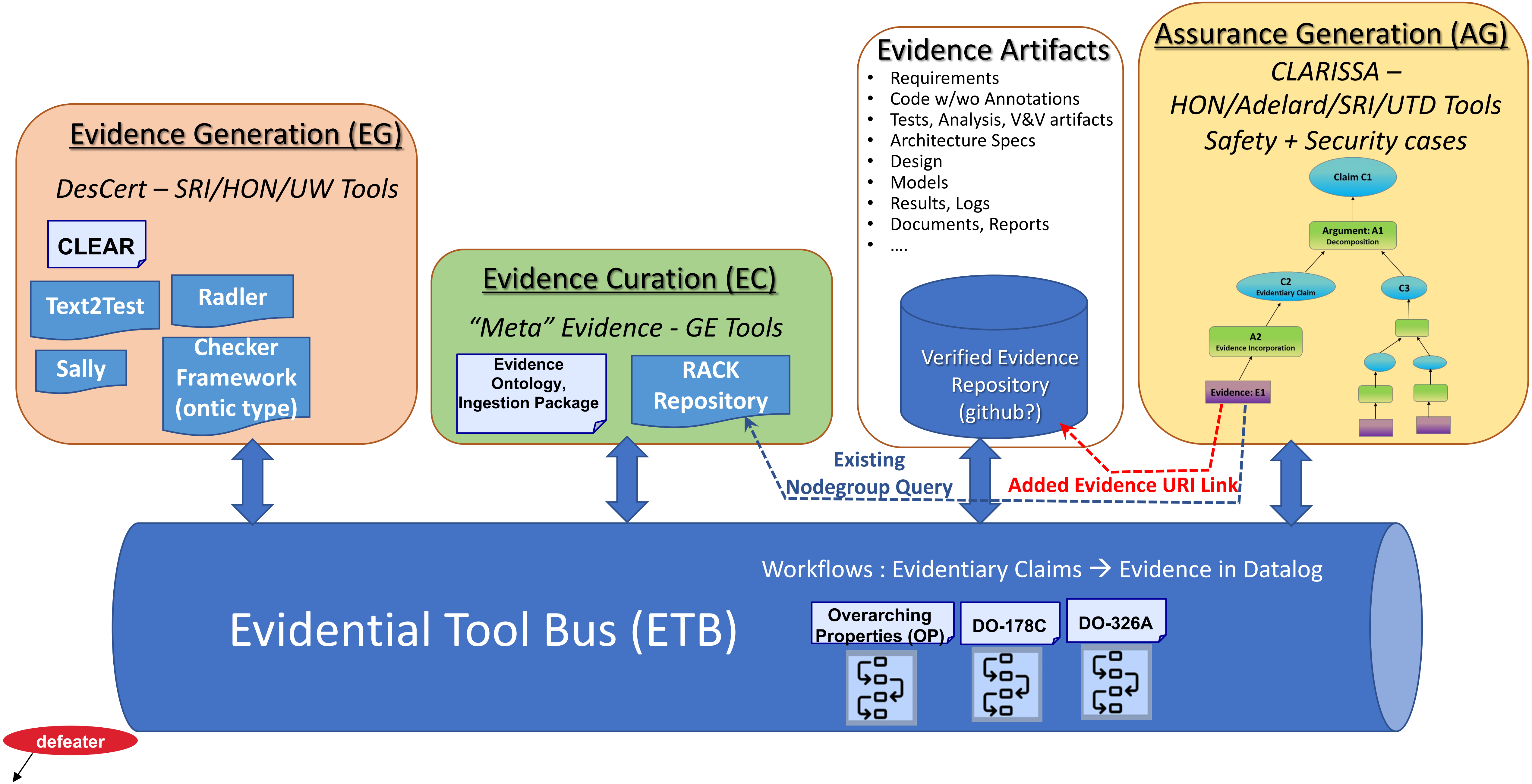


- Architectural design protection (partitioning, enclaves, encryption, secure DDS, resource/ timing guarantees)
- Requirements/design analysis, model check properties, graph semantics analysis
- Protection via type safety and ontic-type-based static code analysis
- Testing: requirements-based, threat-based

E.g., Checker Framework execution

Legend

🟦	Security related
🟨	Safety related
🟩	Activity



Evidence Generation (EG)

DesCert – SRI/HON/UW Tools

CLEAR

Text2Test

Radler

Sally

Checker Framework (ontic type)

Evidence Curation (EC)

“Meta” Evidence - GE Tools

Evidence Ontology, Ingestion Package

RACK Repository

Evidence Artifacts

- Requirements
- Code w/wo Annotations
- Tests, Analysis, V&V artifacts
- Architecture Specs
- Design
- Models
- Results, Logs
- Documents, Reports
- ….

Verified Evidence Repository (github?)

Assurance Generation (AG)

CLARISSA – HON/Adelard/SRI/UTD Tools Safety + Security cases

Existing Nodegroup Query

Added Evidence URI Link

Evidential Tool Bus (ETB)

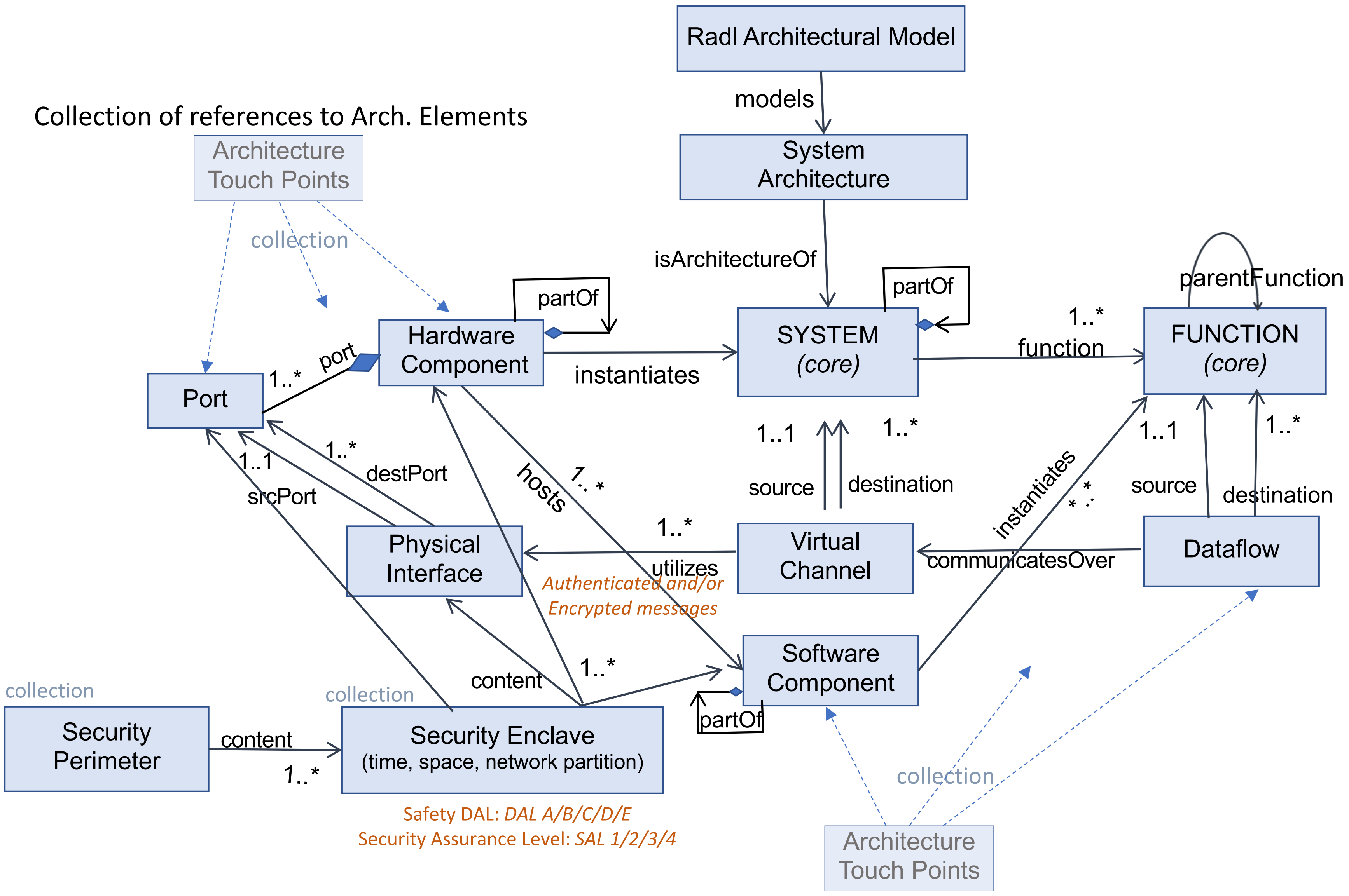
Workflows : Evidentiary Claims → Evidence in Datalog

Overarching Properties (OP)

DO-178C

DO-326A

```
demo_workflow(Input) :-
    system_modeling(Properties, ArchitectureSpec, SecuritySpec, BaseProcessProvData, RADLProvData, CodeFilesProxy,
        RadlFiles, RequirementProvData, RequirementProperties, RequirementPropertyFiles, RequirementFiles,
        ArchControlPropertyResults),
    checkerFramework_type_checking(Properties, ArchitectureSpec, CodeFilesProxy, CFPropertyResults),
    radler_radl_analysis(Properties, ArchitectureSpec, RadlFiles, RadlerPropertyResults),                        EG
    text2Test_requirement_analysis(ArchitectureSpec, BaseProcessProvData, RequirementProperties,
        RequirementProvData, RequirementPropertyFiles, RequirementFiles, ReqAnalysisPropertyResults),
    securityAnalysis_and_ingestion_creation(Properties, ArchitectureSpec, SecuritySpec, BaseProcessProvData,
        RADLProvData, RequirementProvData, RequirementProperties, ArchControlPropertyResults, CFPropertyResults,
        RadlerPropertyResults, ReqAnalysisPropertyResults, IngestionPackageFiles, IngestionPackageManifest),
    rack_ingestion(IngestionPackageFiles, IngestionPackageManifest, SuccessOrError, ErrorMsg, Handle, Diff),  EC
    update_asce_evidence_dnr(Diff, AsceFile),
    detect_property_violation(Diff, PropertyViolated),                          AG
    update_asce_with_defeaters(PropertyViolated, AsceFile).
```



Collection of references to Arch. Elements

Architecture Touch Points

collection

Radl Architectural Model

models

System Architecture

isArchitectureOf

SYSTEM *(core)*

partOf

1..* function

FUNCTION *(core)*

parentFunction

Hardware Component

partOf

instantiates

port 1..*

Port

1..* destPort

1..1 srcPort

hosts 1.. *

1..1 source

1..* destination

Virtual Channel

1..* utilizes

Authenticated and/or Encrypted messages

Physical Interface

communicatesOver

Dataflow

1..1 source

1..* destination

instantiates * ..*

Software Component

partOf

content

1..*

collection

Security Perimeter

content 1..*

collection

Security Enclave (time, space, network partition)

Safety DAL: *DAL A/B/C/D/E*

Security Assurance Level: *SAL 1/2/3/4*

collection

Architecture Touch Points