



ARBITER & UPSAT (Assured Open Source: Experience Report)

Howard Reubenstein -- howard.reubenstein@str.us

Greg Eakman – greg.eakman@str.us

7 May 2023

V2.3

DOCUMENT RESTRICTIONS

Distribution Statement 'A' (Approved for Public Release, Distribution Unlimited). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA).


Overview



- **Evidence-based Assurance**
- **Top-down Mission Critical Claims**
- **UPSat Critical Claims**
- **Watchdog Errors Revealed**
- **Working with an Open-Source Project**

What Would Give You Confidence That Your System Could Standup to Attack?



- **Next Generation Assurance (NGA) goes beyond required check-box assurance, e.g. Risk Management Framework (RMF)**
- **Authority to Operate (ATO) via RMF is mandated, but** 
- **Hackers know our systems better than we do (Rob Joyce – NSA, Cybersecurity Director)**
- **This presentation is about the use of evidence-based assurance that will provide confidence in critical system properties**



Assurance Cases and Formal Methods

STR SECURE
DECISIONS
A DIVISION OF APPLIED VISIONS, INC.

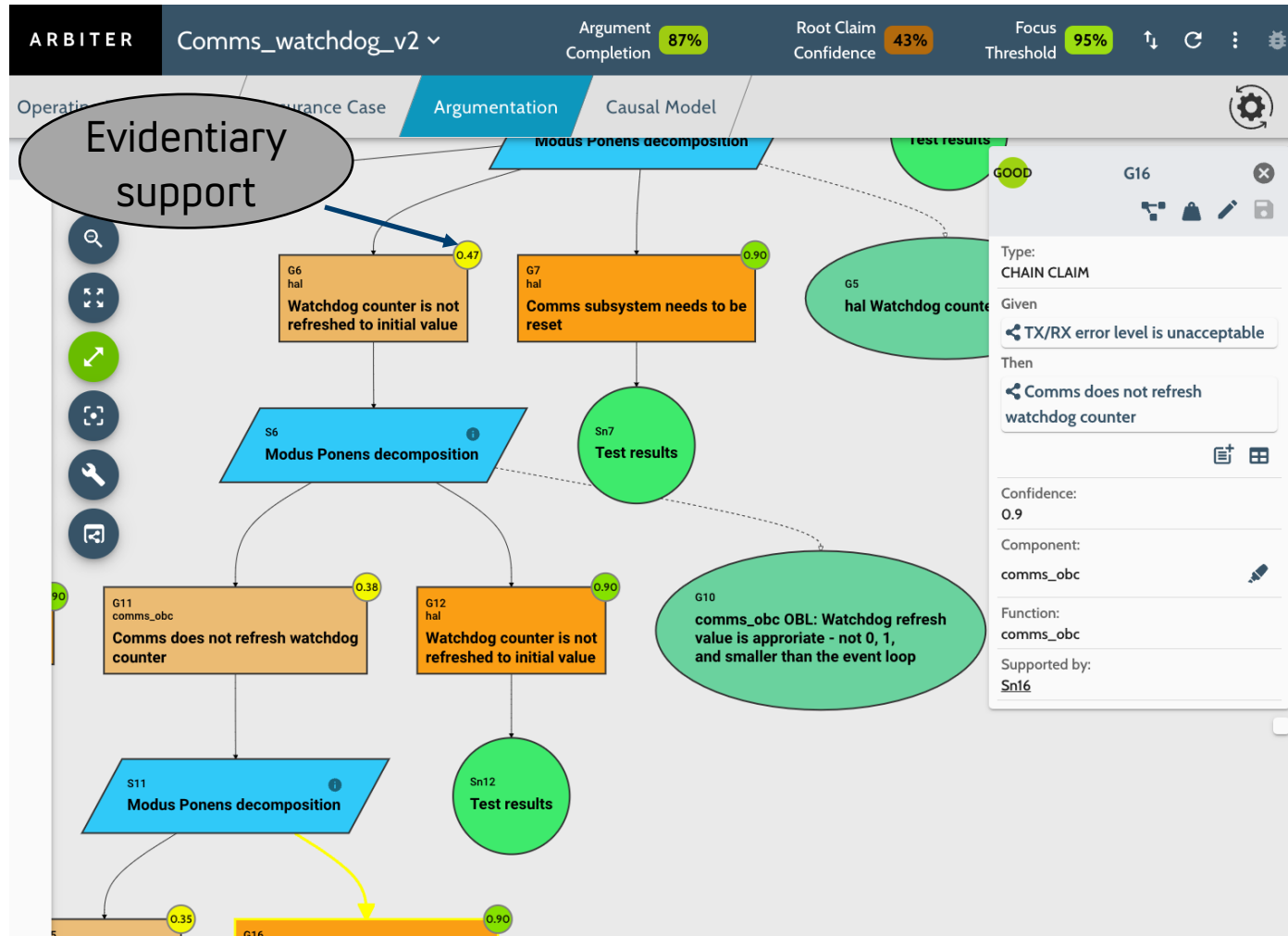
str

- For a cyber-physical system *there is no single proof* (or even set of proofs) that by itself will establish critical **SYSTEM** level properties
- There are many properties that might be proven about the software in a system, *which ones are valuable to prove?*
- An assurance case provides an informal argument to justify a claim using *multiple evidence sources*: testing, analysis, trade studies, ... and proof

ARBITER: A System Used to Build Evidence-Based Assurance Arguments

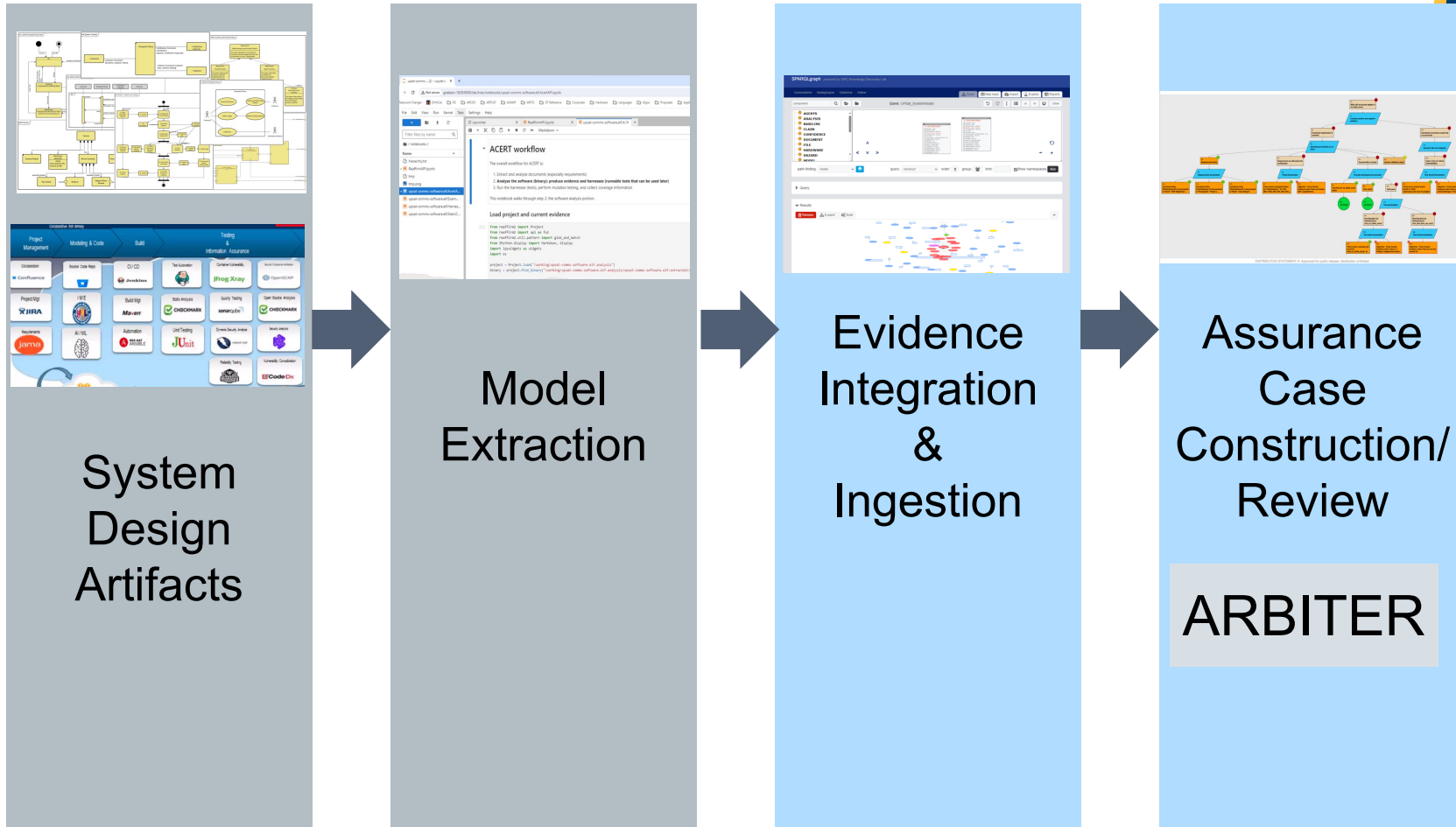


- Uses: Claim, Argument, Evidence, Defeater assurance case notation
- Built as part of DARPA's ARCOS program
- Designed with a philosophy to avoid users MSU ("making stuff up")
- Provides assessment of evidentiary support (confidence) to guide development of assurance case





Integrated Assurance Information Flow



Automated Rapid Certification of Software (ARCOS) provides evidence-based assurance to support Authority to Operate (ATO) decisions

ARBITER Pilot Workflow



- **Phase 1: Identify mission specific critical claims. Establish top level schematic assurance case structure (i.e., what methodology will be used to establish assurance)**
- **Phase 2: Identify types of evidence that can support assurance case**
 - For discrete evidence consider connector/import strategy for evidence acquisition
 - For informal evidence use document evidence with populated structured meta-data
- **Phase 3: Structure assurance case and include evidence**
 - Populate evidence repository
 - Define top level assurance case structure (methodology)
 - Develop assurance case in ARBITER with strategy templates

What kind of evidence is needed to substantiate high impact claims and how to source that evidence?

Overarching Critical Claims Explored



Overarching Property Methodology:

- Intent – specification is correct
- Correctness – implementation is correct
- Innocuity – no unacceptable “collateral” impact

- 1. The watchdog shall reset the comms subsystem XX clock ticks after the subsystem enters a degraded mode (*correctness*)**
- 2. The watchdog shall never reset the comms subsystem in any other case (*innocuity*)**

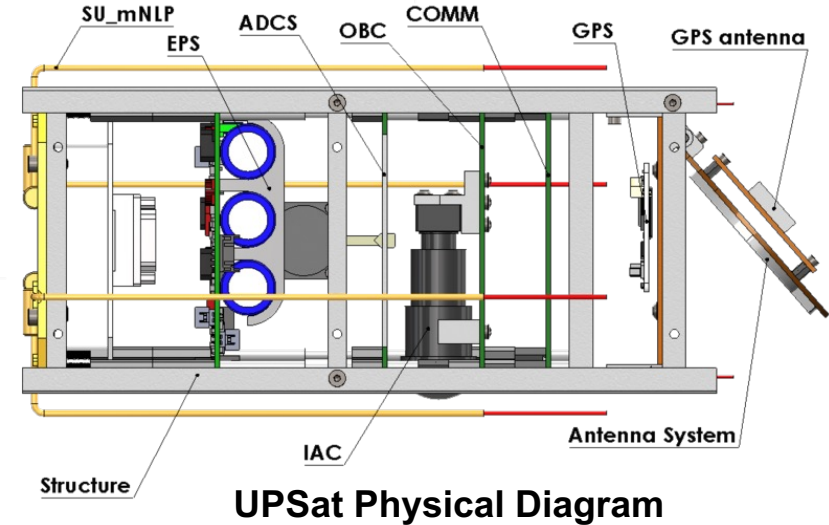
Overarching Properties: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9594298>



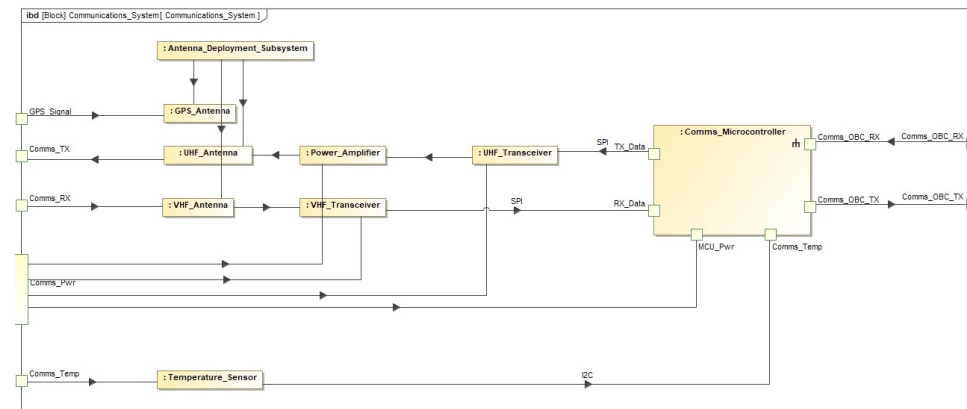
UPSat

UPSat Assurance Pilot

- UPSat open-source hardware and software CubeSat launched April 2017
- Part of the QB50 network of 50 CubeSats
- ARCOS focus on Communications
 - Periodic telemetry transmission
 - Receive, process, route messages
 - AX.25, ECSS protocols
 - Encryption
 - Data encodings
 - Hardware interfaces
 - Transmitter
 - Receiver
 - UART
 - Watchdog

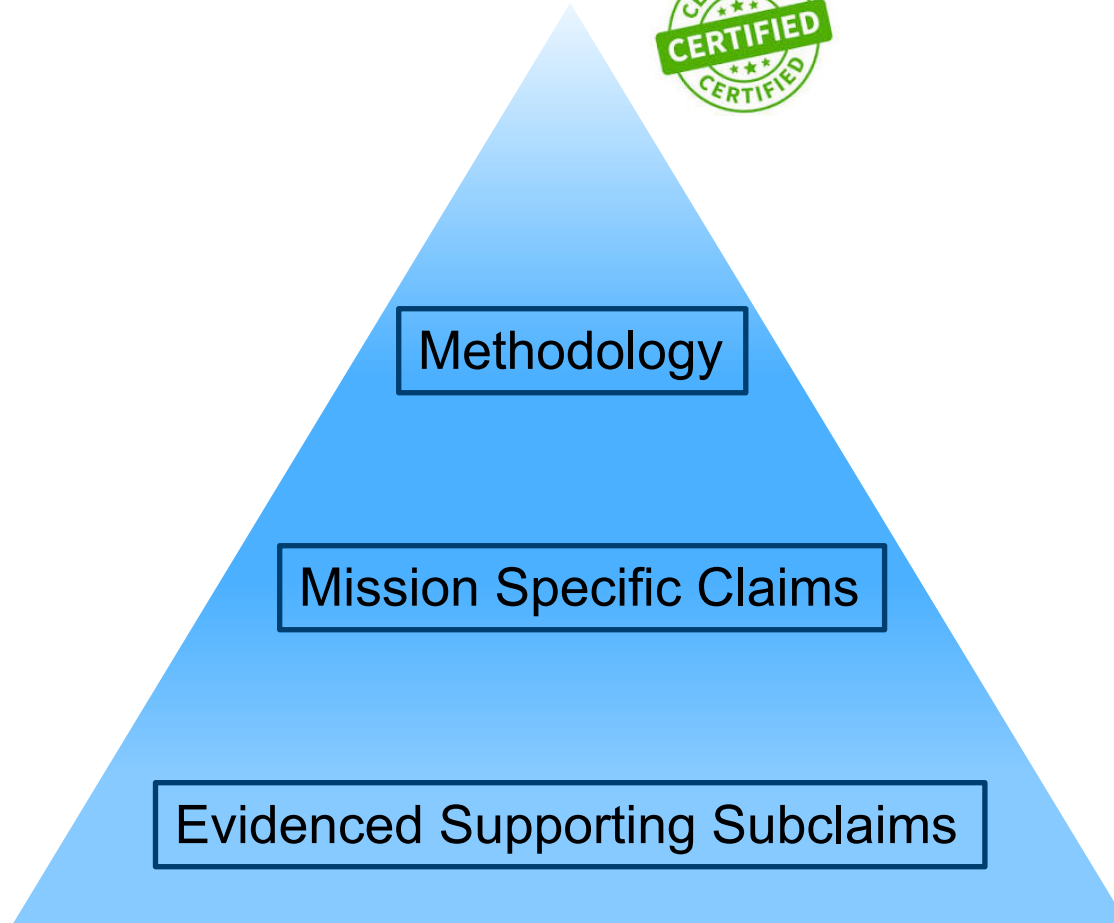


UPSat Physical Diagram



UPSat SysML Internal Block Diagram

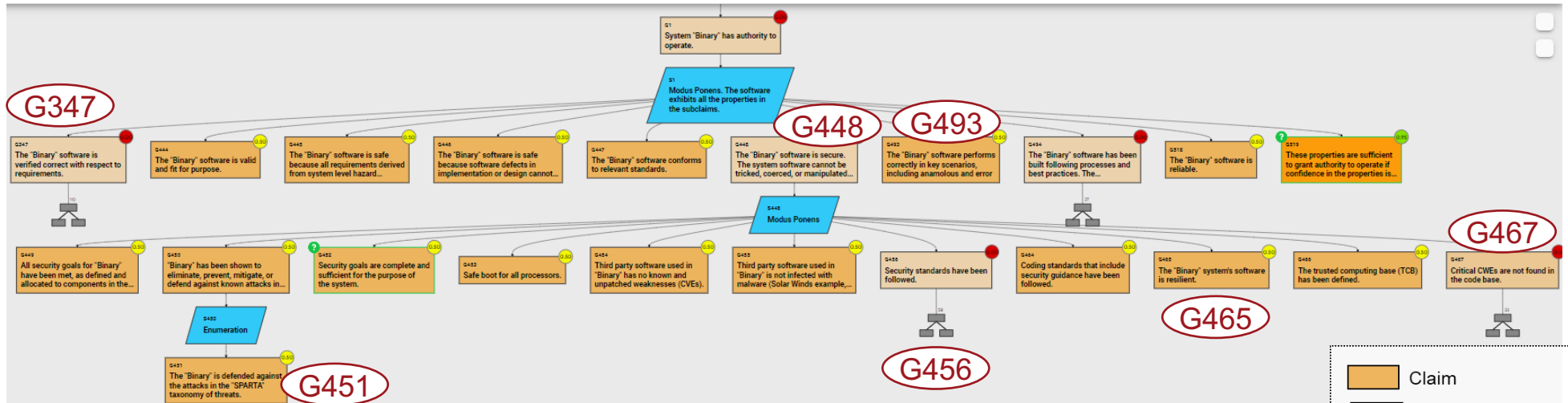
Assurance Case Organization



UPSat Assurance Case: Methodology Level

SECURE
DECISIONS
A DIVISION OF APPLIED VISIONS, LLC.

str



Key Claim Nodes

G347 Correct wrt requirements

G448 Software is secure

G451 Mitigates the SPARTA attack taxonomy

G465 Software is resilient

G493 Satisfies mission critical claims

G467 Software is free of critical CWEs

UPSat Evidence and Confidence Sources

ALL SOURCE
OPERATIONS
A DIVISION OF APPLIED VISIONS, LLC.

str

- **COMM subsystem tests run in QEMU simulation environment**
 - AX.25 communications protocol encoding/decoding
- **Functional Requirements**
 - Some requirements supported by traditional pass/fail tests
 - Causal model requirements (e.g., Watchdog)
 - Instrumentation to collect evidence for causal model analysis
- **Security**
 - Static analysis of CWEs
 - Analysis of SPARTA attacks and mitigations
- **Evidentiary support**
 - Dynamic evidence directly applied to causal model from QEMU traces
 - Analytic/document evidence applied to claims
 - Objections used as one source of counter evidence

Space Attack Research & Tactic Analysis (SPARTA)

show sub-techniques hide sub-techniques

Resource Development	Initial Access	Execution	Persistence	Defense Evasion	Lateral Movement	Exfiltration
Techniques	12 techniques	18 techniques	5 techniques	11 techniques	7 techniques	10 techniques
Structure (4)	Compromise Supply Chain (2)	Replay (2)	Memory Compromise (3)	Disable Fault Management (2)	Hosted Payload (2)	Replay (2)
Compromise Structure (3)	Compromise Software Defined Radio (2)	Position, Navigation, and Timing (PNT) Spoofing (2)	Backdoor (2)	Prevent Downlink (2)	Exploit Lack of Bus Segregation (2)	Side-Channel Attack (2)
Cyber Threats (2)	Crosslink via Compromised Neighbor (2)	Modify Authentication Process (2)	Ground System Presence (2)	Modify On-Board Values (12)	Constellation Hopping via Crosslink (2)	Eavesdrop (2)
Non-Cyber Threats (4)	Secondary/Backup Communications (2)	Compromise Boot Memory (2)	Replace Cryptographic Keys (2)	Masquerading (2)	Visiting Vehicle Interface(s) (2)	Out-of-Band Communication Link (2)

<https://sparta.aerospace.org/>

UPSat Critical Claims Explored

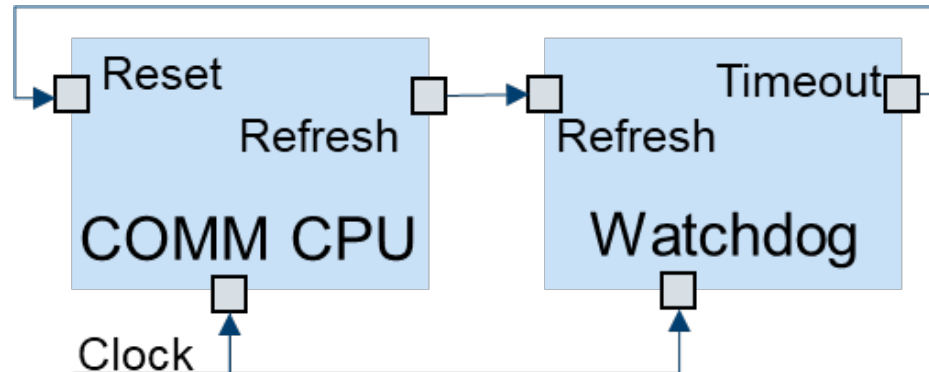


- 1. The watchdog shall reset the comms subsystem XX clock ticks after the subsystem enters a degraded mode (correctness)**
- 2. The watchdog shall never reset the comms subsystem in any other case (innocuity)**

Correctness of Watchdog Timer (WDT) Automated Reset



- Supports safety, security, availability requirements
- Every iteration of the COMM event loop:
 - If subsystem state is OK, refresh the watchdog timer
 - Else, ignore the watchdog, and the subsystem eventually resets
- Key requirements claims
 1. The watchdog shall reset the comms subsystem XX clock ticks after the subsystem enters a degraded mode.
 2. The watchdog shall never reset the comms subsystem in any other case.



Watchdog Simplified Causal Model



QEMU COMM SW/Watchdog Traces

```

{
  "sim_config": {
    "wd_counter_init": 10, "rx_error_init": 0, "tx_error_init": 0, "rx_error_rate": 0.000000, "tx_error_rate": 0.000000, "sat_clock_error_rate": 0.000000,
    "loop_time": 100, "sim_time": 551000
  },
  "data": [
    {"sim_clock": 1100, "stats_tick": 0, "sat_clock": 1100, "wd_counter": 10, "rx_errors": 0, "tx_errors": 0, "system_status": "COMMS_DISPATCH"},
    {"sim_clock": 1100, "stats_tick": 1100, "sat_clock": 1100, "wd_counter": 10, "rx_errors": 0, "tx_errors": 0, "system_status": "REFRESH"},
    {"sim_clock": 2200, "stats_tick": 1100, "sat_clock": 2200, "wd_counter": 10, "rx_errors": 0, "tx_errors": 0, "system_status": "COMMS_DISPATCH"},
    {"sim_clock": 2200, "stats_tick": 2200, "sat_clock": 2200, "wd_counter": 10, "rx_errors": 0, "tx_errors": 0, "system_status": "REFRESH"}]
  }
  
```

Claim 1

COMM Not OK



Counter Not Refreshed



Watchdog Works



COMM Reset



The watchdog shall reset the comms subsystem XX clock ticks after the subsystem enters a degraded mode.

100%

Contract Extraction

Claim 2

COMM OK



Refresh Counter



Watchdog Works



COMM Not Reset



The watchdog shall never reset the comms subsystem in any other case.

100%

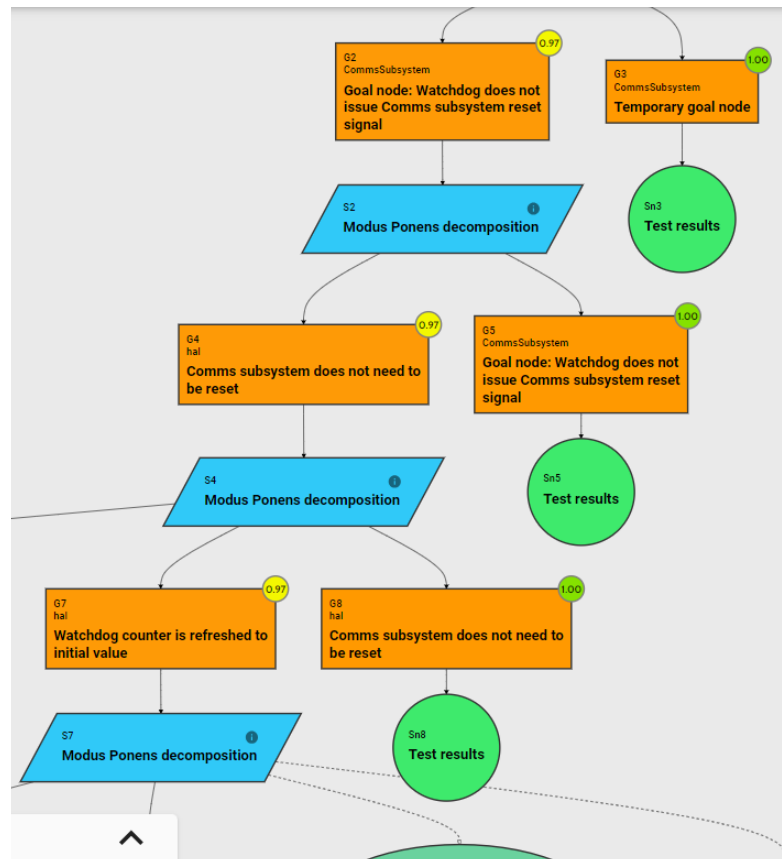
Additional Test Data – No Tx Mode



```
int32_t comms_routine_dispatcher(comms_tx_job_list_t *tx_jobs)
{
  if(tx_jobs == NULL){
    return COMMS_STATUS_NO_DATA;
  }
  ...
  if (comms_stats.rx_failed_cnt < 10 && comms_stats.tx_failed_cnt < 5) {
    HAL_IWDG_Refresh (&hiwdg);
  }
}
```

14 contract failures

```
"wd_refresh": [
  [
    "T",
    "T",
    500,
    514
  ],
```

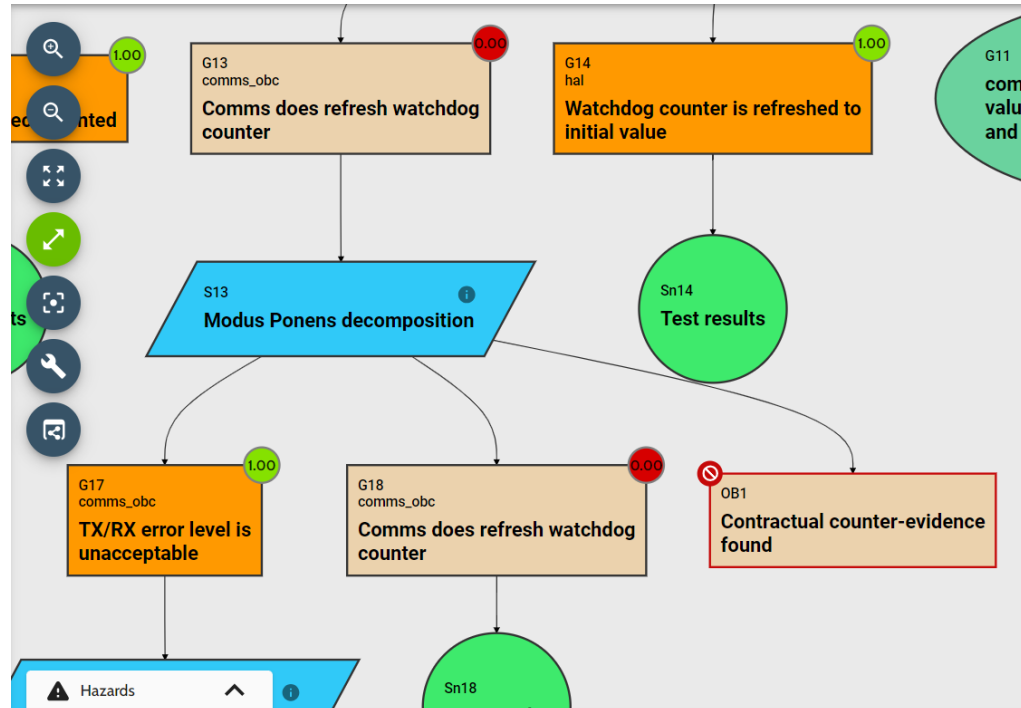


With Tx Disabled Tests

97%

Amplifying Counter-Evidence

- Failures in CPT where the contract postcondition should be TRUE generate objections



No-Tx Tests – Objection generated based on counter-evidence in software contracts

UPSat Pilot Post-Mortem



- 1. We developed claims and derived requirements**
- 2. We built a causal model (CM) and ran telemetry from the emulation through it as evidence**
- 3. Code inspection made us suspicious about the Watchdog. We added the innocuity claim and updated the emulator to provide more telemetry**
- 4. Additional emulation allowed us to “discover” the problem**
- 5. Nevertheless, contracts and CMs are powerful and would have discovered the issue if intent model present**

Working with Open-Source Systems



- **UPSat: a complete open-source system, versus third party libraries or components**
- **Missing systems engineering artifacts**
 - Reengineered missing systems engineering artifacts upon which assurance is typically built
 - System/software architecture and design
 - Requirements (some informally described in thesis, others from the QB50 program)
- **Correctness**
 - Identify critical claims for mission specific application
 - Overlay code with contracts to derive a causal model
 - Instrument cyber-physical system via QEMU emulation to extract contractual evidence
- **Security**
 - Static analysis to look for CWEs
 - SPARTA attack technique taxonomy for the satellite domain
 - No need here to architect/isolate untrusted code

Summary



- **System engineering artifacts do double duty: they provide design guidance and define the intent behind system operation**
- **Causal models can confirm that the system is operating as intended**
- **In open-source systems we need the design artifacts and evidence that might accompany a more formal development process concerned with assuring the overarching properties**
- **Assurance cases provide an informal unifying argument behind mission critical system claims**

Thank You For Your Attention



Questions?

greg.eakman@str.us

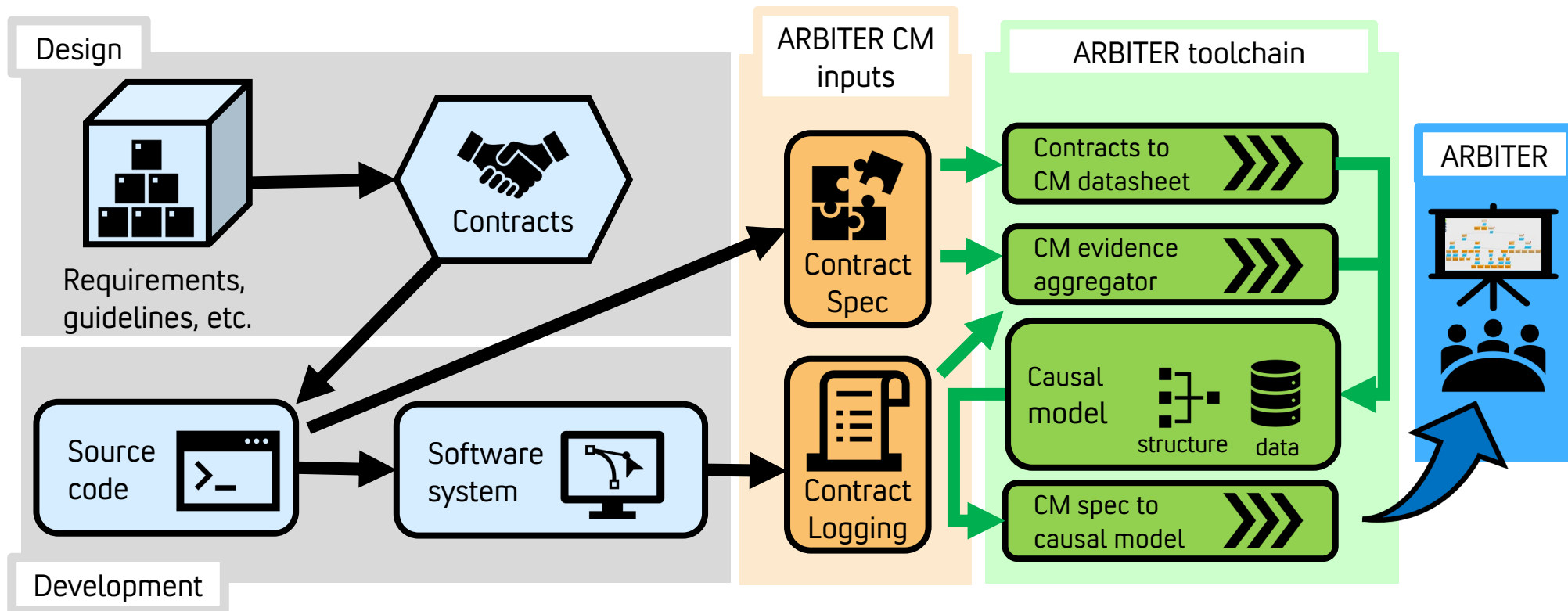
howard.reubenstein@str.us



ARBITER



Building a Causal Model from Contracts



The ARBITER toolchain extracts causal model structure and probability tables from a [language-independent contract specification](#) and saves them as [intermediate data products](#). The extraction from source code uses language-specific tooling and commenting schemes (currently supported languages are Rust and Python).