Large Scale Assurance

Briefing to Army AvMC

July 12, 2023

Dionisio de Niz and Gabriel Moreno
Assuring Cyber-Physical Systems

Carnegie Mellon University Software Engineering Institute

Problem

Assurance of new capability cannot be provided with the required speed.

- Inability to integrate the many intertwined types of analyses required for assurance.
  - e.g., control, timing, security, logical correctness
- Lack of awareness of assurance interdependence between subsystems.
  - Extensive testing needed to discover interactions between multiple subsystems.
- Lack of effective reuse of assurance results.

Solution Overview

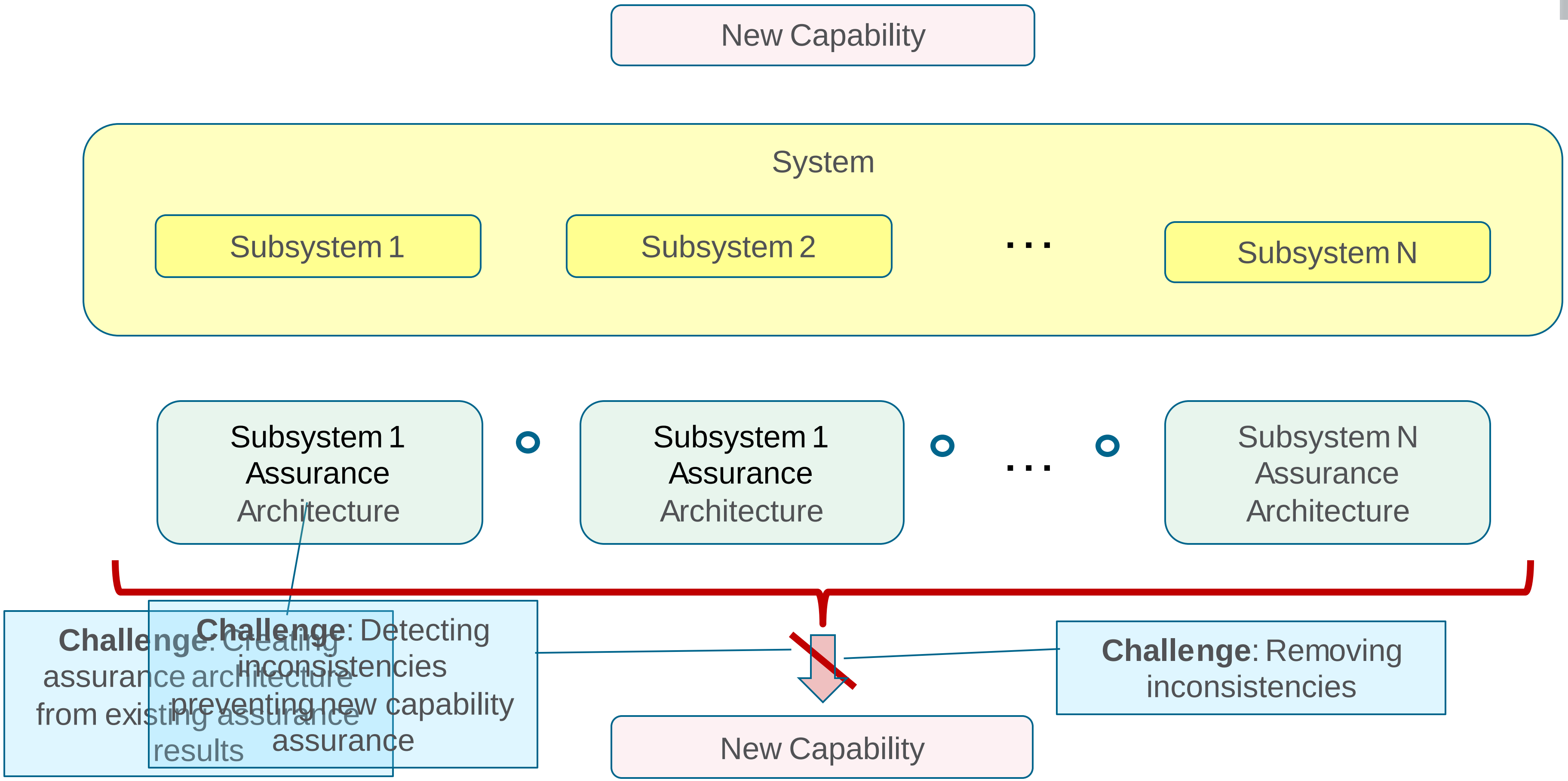
Focus: rapid assurance with confidence through

- increased formality
- maximal reuse of prior assurance results
- automatically combine assurance results

Approach:

- Incrementally coalesce and structure prior assurance results of subsystems into assurance architectures and
- compose those assurance architectures to assure new capabilities provided by the system.

LSA Approach



State of the Practice

DoD is moving towards Continuous Authorization to Operate (cATO) *"to deliver cyber resilient capabilities to warfighters at the speed of relevance"* [DoD 2022]

- cATO is focused on cybersecurity and does not deal with other mission-critical system properties

CI/CD and DevSecOps use some automatable assurance approaches (e.g., regression and integration testing) that evolve with the system [Carleton 2021]

- Assurance arguments cannot evolve at the fast pace that DevSecOps affords
- The evaluation of assurance arguments cannot be automated

State of the Art

Assurance cases are structured arguments in tree form to show how evidence supports claims that the system has some property (e.g., being safe or secure to operate).

- Argument steps often only increase our belief in the correctness of a claim (inductive) but do not ensure that it is correct [Bloomfield 2020]
- Researchers have proposed to make steps deductive to make them formal
- But it is recognized that full formalization is not practical [Viger 2020]

Assurance cases do not support mathematical composition

- Modularization is supported as syntactic vehicle to connect parts of an assurance case [Denney 2018]

ETMAC project's analysis contracts formalize the dependencies between claims and analyses with assumptions and guarantees [de Niz 2023]

- FALSA ETP building on analysis contracts
- LSA will build on FALSA's results and ETMAC's *contract argumentation tree*

ETMAC Argumentation Motivation

Model-Analyze-Build

Late Discovery of Design Errors in DoD Systems is very costly.

Architecture modeling and analysis can **detect** design **error early**

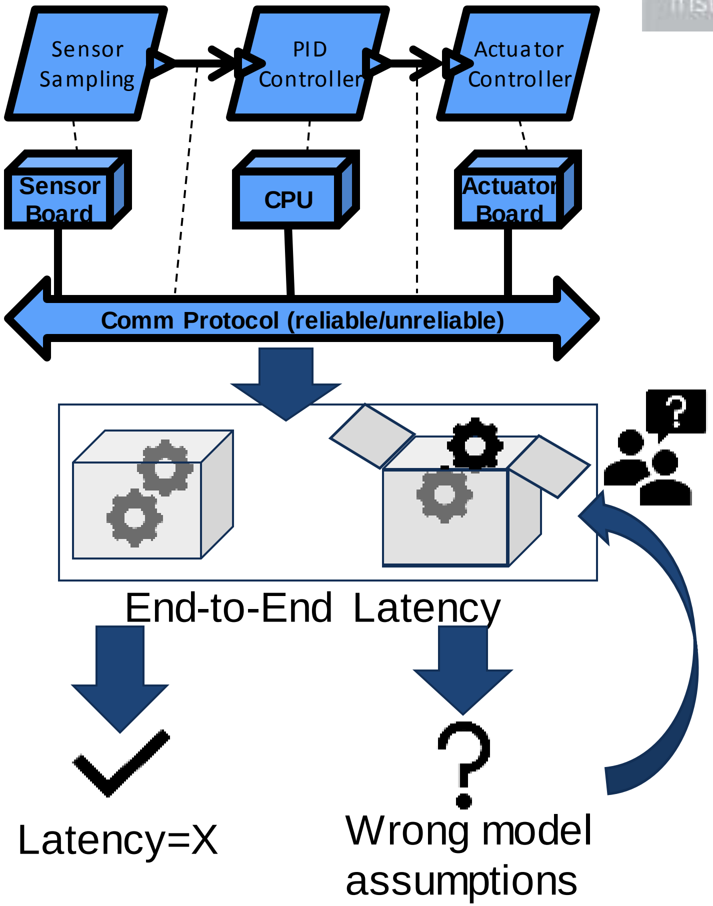
BUT:

Analysis assumptions are often implicit

if analysis **assumptions not met**: analyses break down for reasons not clear to users of analysis tools.

E.g., e2e Latency Assumption: periods multiple of each other (harmonic)

DoD barrier for adoption

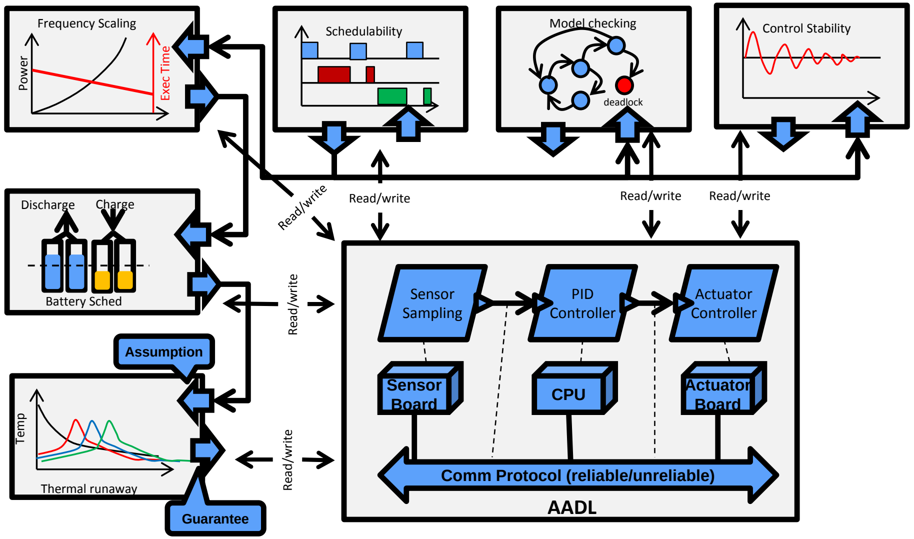


Digital Engineering: Multiple Claims - Multiple Analyses

Different Assurance Claims

- Combine multiple analysis
- Validate assumptions
- Resolve assumption conflicts

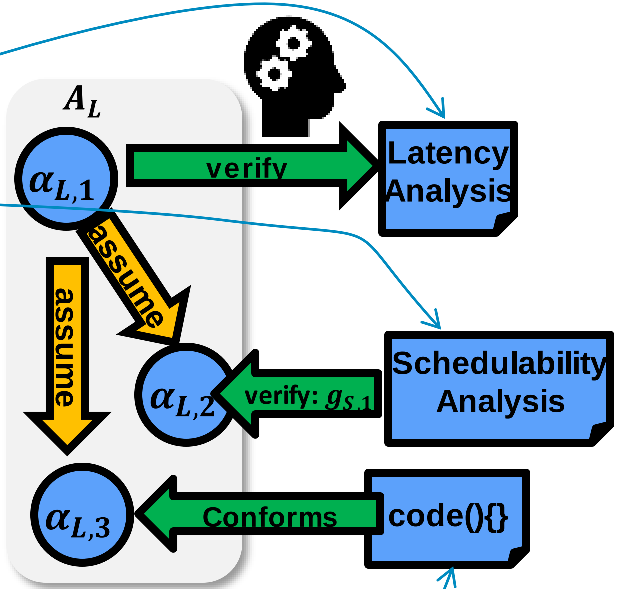
Integrate into arguments to satisfy claims



Analysis Contract: Tracking Assumptions and Guarantees

```
contract{
  inputs:
        E2ELatancies
  assumptions:
        areConnectionsDelayed()
        areDeadlinesConstrained()
        areTasksSchedulable()
        areAllThreadsPeriodic()
  analysis:
        meetEndToEndLatencies()
  guarantee:
        [E2EResponses[i] <= E2ELatencies[i]
          for i in range(len(Responses))]
}
```

$C_L = (A_L, G_L)$ with $A_L = \{\alpha_{L,1}, \alpha_{L,2}, \alpha_{L,3}\}$



Shift Left And Down to the Metal

Early Analysis

- Evaluate design decisions with partial information
- E.g., latency analysis before worst-case execution time (WCET)
  - periods of tasks must be multiples of each other

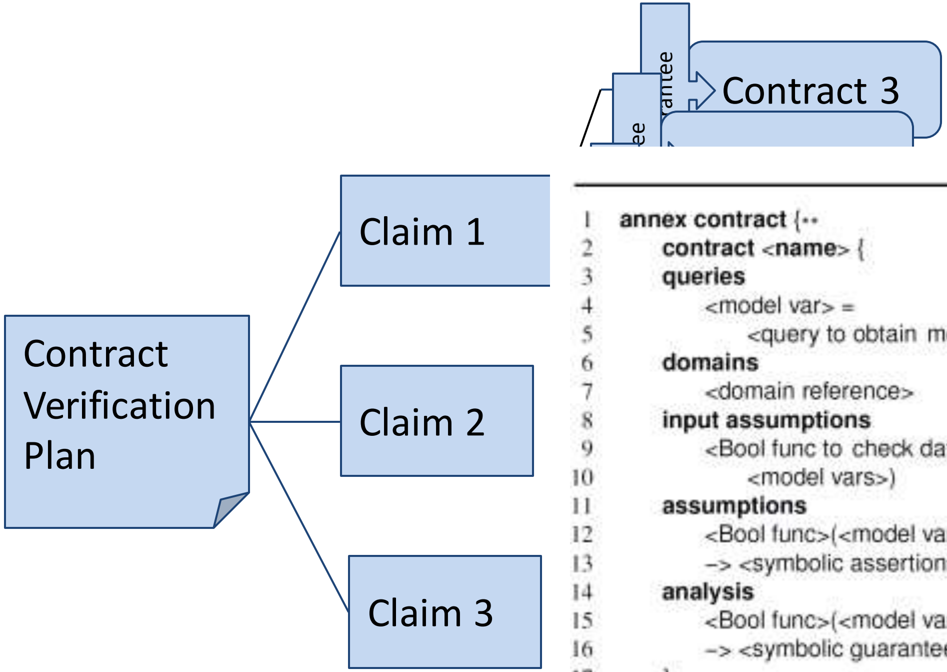
Refinement

- Track pending information
  - WCET
- Track and execute pending verification
  - Schedulability

Conformance

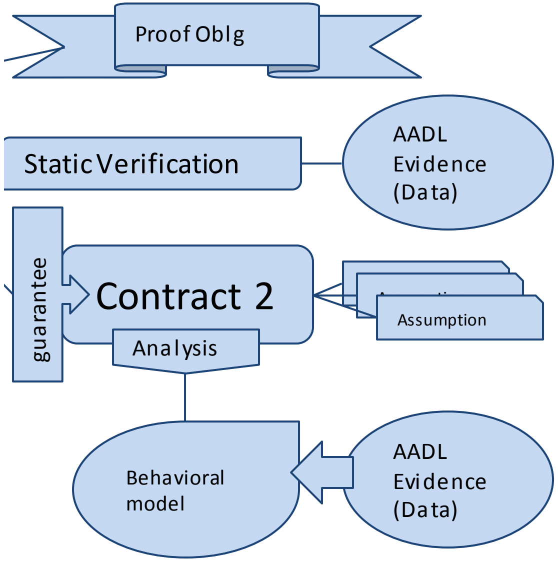
- Track implementation assumption
- Verify implementation conformance
  - Task executed strictly periodic $\alpha_{L,3}$

Assurance Contract Argumentation

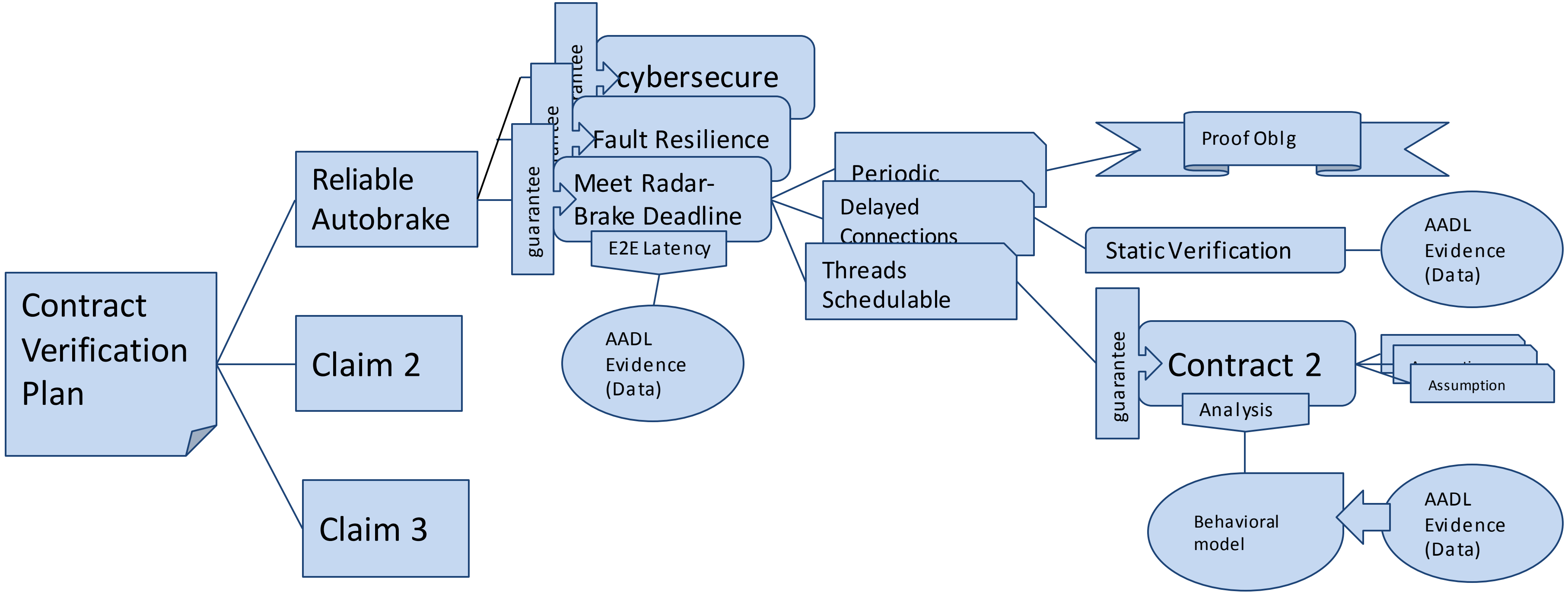



```
annex contract {**
    contract <name> {
    queries
        <model var> =
            <query to obtain model data>
    domains
        <domain reference>
    input assumptions
        <Bool func to check data consistency>(
            <model vars>)
    assumptions
        <Bool func>(<model vars>)
        -> <symbolic assertion>
    analysis
        <Bool func>(<model vars>)
        -> <symbolic guarantee>
    }
**};
```

Listing 1: Contract Template



Assurance Contract Argumentation



Symbolic Contract Argumentation

Assumptions

- Constraints that must be satisfied for a valid analysis

Analysis

- Evaluate whether the guarantee can be discharged

Guarantee

- Assertion presented as a true fact on model

Implementation

- Satisfiability Modulo Theories (Z3)
- Implements contract argumentation
  - Evaluate whether constrains can be satisfied with facts from analysis guarantees
- Validate assumptions
  - Proof obligations: lack of constraints allow any value that satisfy assumption (e.g., RM priorities)

Encoding In SMT

Verification Plan

- $P = (K, D)$
  - $K \coloneqq \{(k_i, C_i)\}$
  - k_i predicates over symbolic variables

Contract

- $C_i = (V_d, Q, I, A, G, N)$
  - V_d symbolic variables from a domain
  - Q model variables (e.g., from AADL models)
  - I input assumptions (if enough data for analysis)
  - $A = (p, a)$ set of assumptions
  - G guarantee
  - N analysis predicate (imperative Boolean function)

Algorithm 1 getSMTEncoding($Plan$)

$F \leftarrow \{k_i | (k_i, C_i) \in Plan.K\}$
$T \leftarrow \{C_i | (k_i, C_i) \in Plan.K\}$
while $T \neq \emptyset$ **do**
    select t from T and remove it from T
    if t is argument **then**
        $T \leftarrow T \cup t.C_\alpha$
        $F \leftarrow F \cup (\text{replG4C}(t.F_\alpha) \implies t.G_\alpha)$
    else if $\bigwedge_{i \in t.I} i$ **then**
        for $p \in \{p | (p, a) \in t.A\}$ **do**
            if p is contract **then**
                $T \leftarrow T \cup p$
                $F \leftarrow F \cup (\bigwedge_{\forall \{a | (p,a) \in t.A\}} a \wedge N \implies t.G)$
            else
                $F \leftarrow F \cup (p \implies a)$
            end if
        end for
    end if
end while
return F

Contract Argumentation Scalability

Exploit Knowledge from Scientific Domain

- Efficient algorithms from specialized domains
  - E.g., greedy worst-case response time in real-time theory
  - Implemented in imperative languages

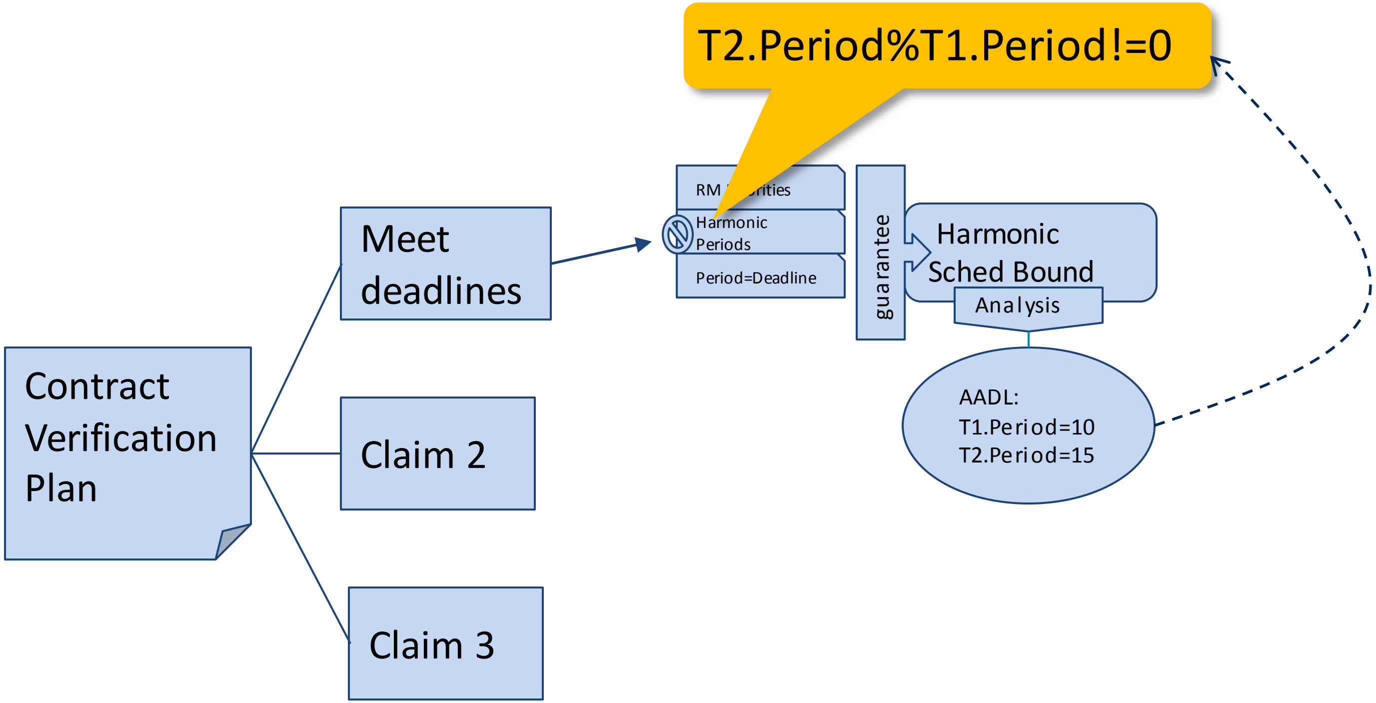
Assume correctness of analysis

- When validating the contract argumentation
- To be connected with other lower-level verification results
  - E.g., PROSA: coq (theorem prover) verification of real-time theory

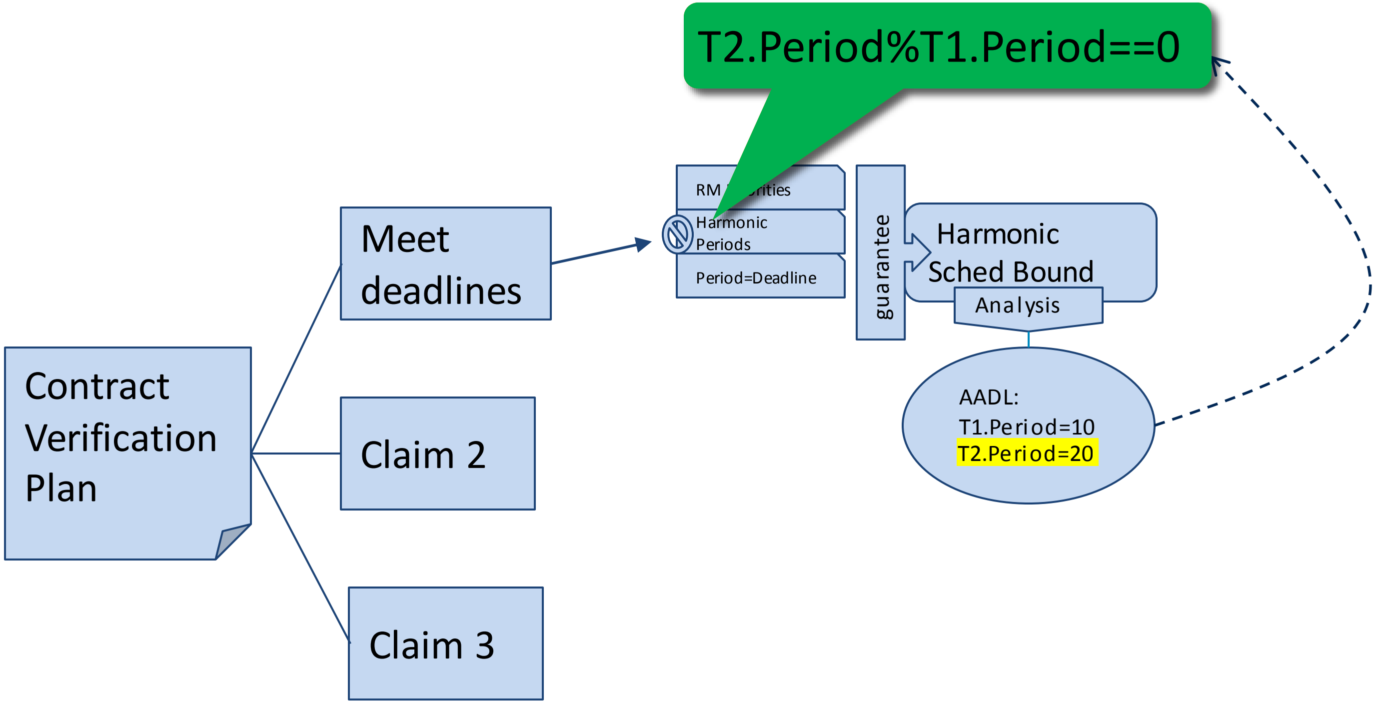
Correctness of implementation

- Exploit proven properties of runtime mechanisms: e.g., schedulers, hypervisors
- Exploit code generation
- Deferred code verification to conform to assumptions

Repairing Assumptions



Repairing Assumptions



Repairing Assumptions

T2.Period%T1.Period!=0

Contract Verification Plan

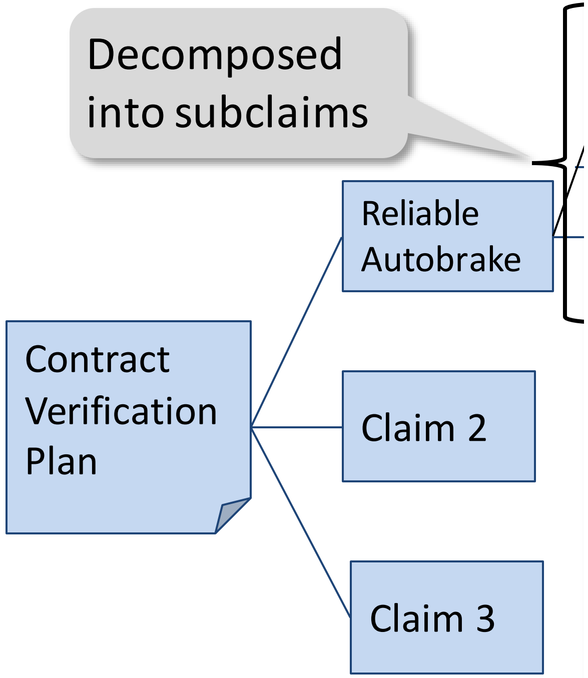
```
annex contract {**
argument schedulability
    argument
        Or(RMBound, RTA)
    ...
}

contract RMBound {
    assumptions
        RMPriorities(periods,  priorities )
    analysis
        RMBoundTest(...)
}

contract RTA {
    assumptions
        ...
    analysis
        RTATest (...)
}
**}
```

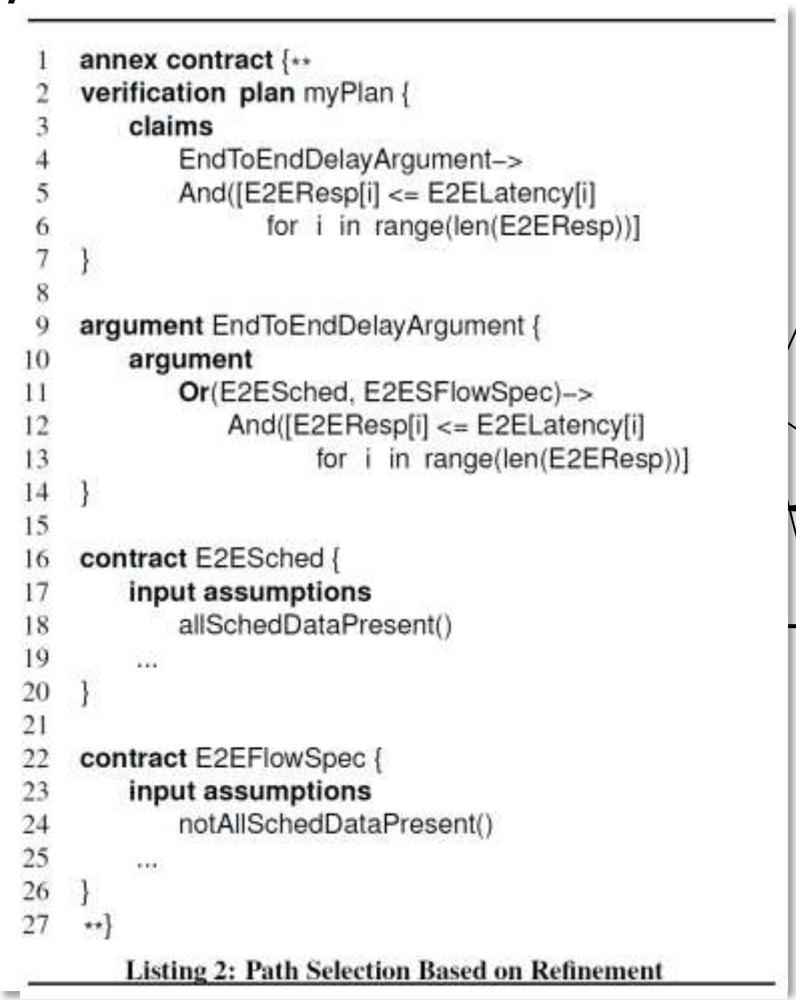
Listing 3: Path Selection Based on Assumptions

Sugg
Non
doe
periods

c

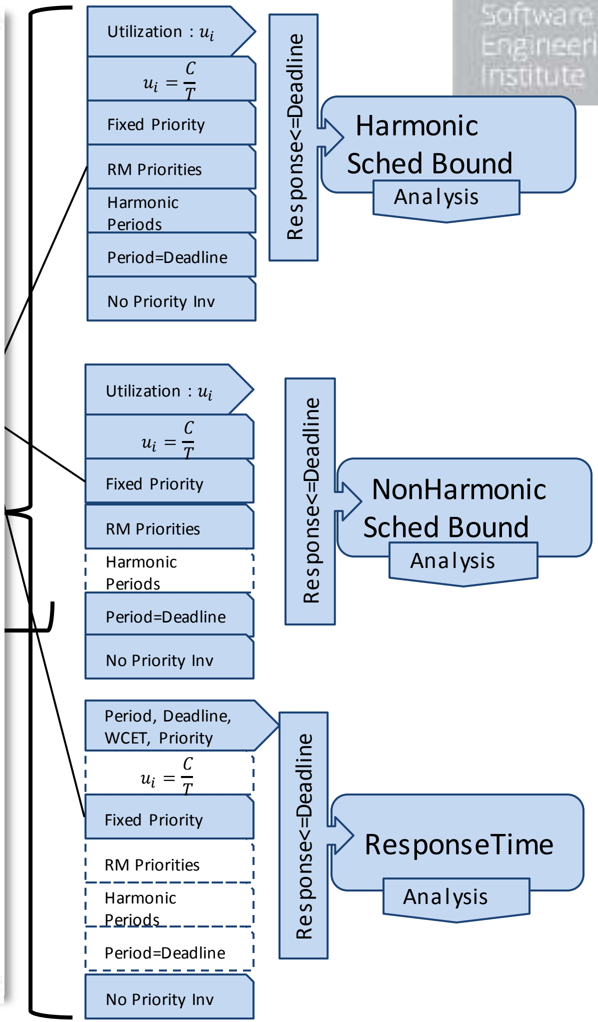
Argument Modularity



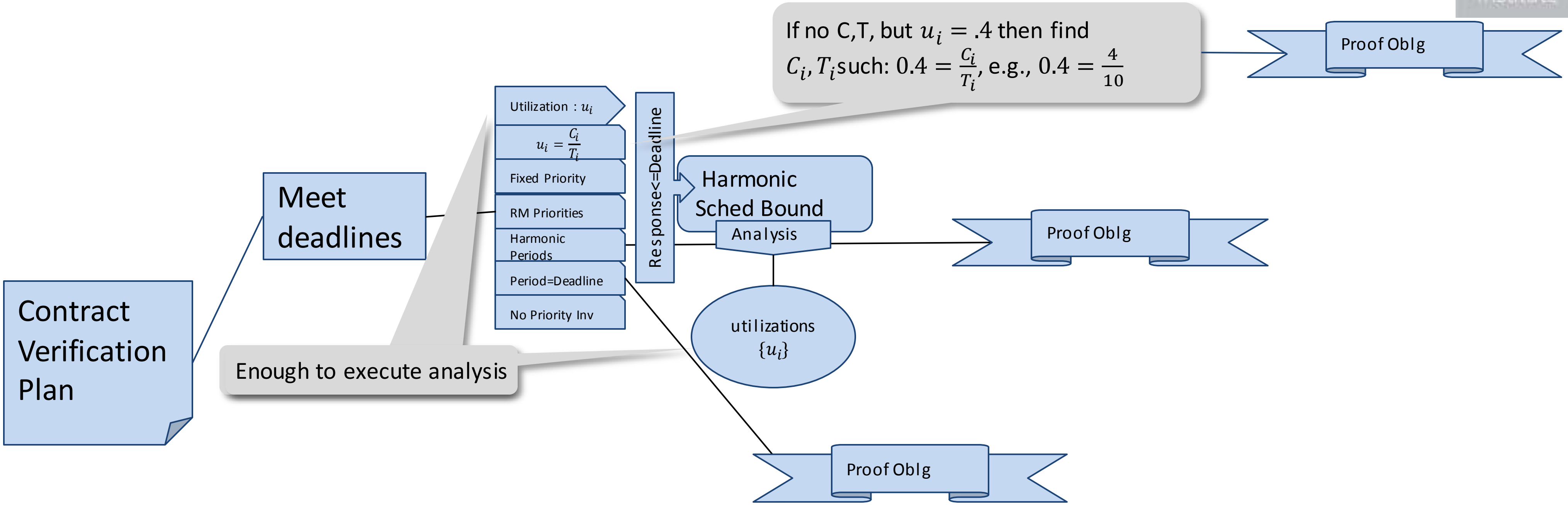


```
annex contract {**
verification plan myPlan {
    claims
        EndToEndDelayArgument->
        And([E2EResp[i] <= E2ELatency[i]
                for i in range(len(E2EResp))]
}

argument EndToEndDelayArgument {
    argument
        Or(E2ESched, E2ESFlowSpec)->
            And([E2EResp[i] <= E2ELatency[i]
                    for i in range(len(E2EResp))]
}

contract E2ESched {
    input assumptions
        allSchedDataPresent()
        ...
}

contract E2EFlowSpec {
    input assumptions
        notAllSchedDataPresent()
        ...
}
**}
```

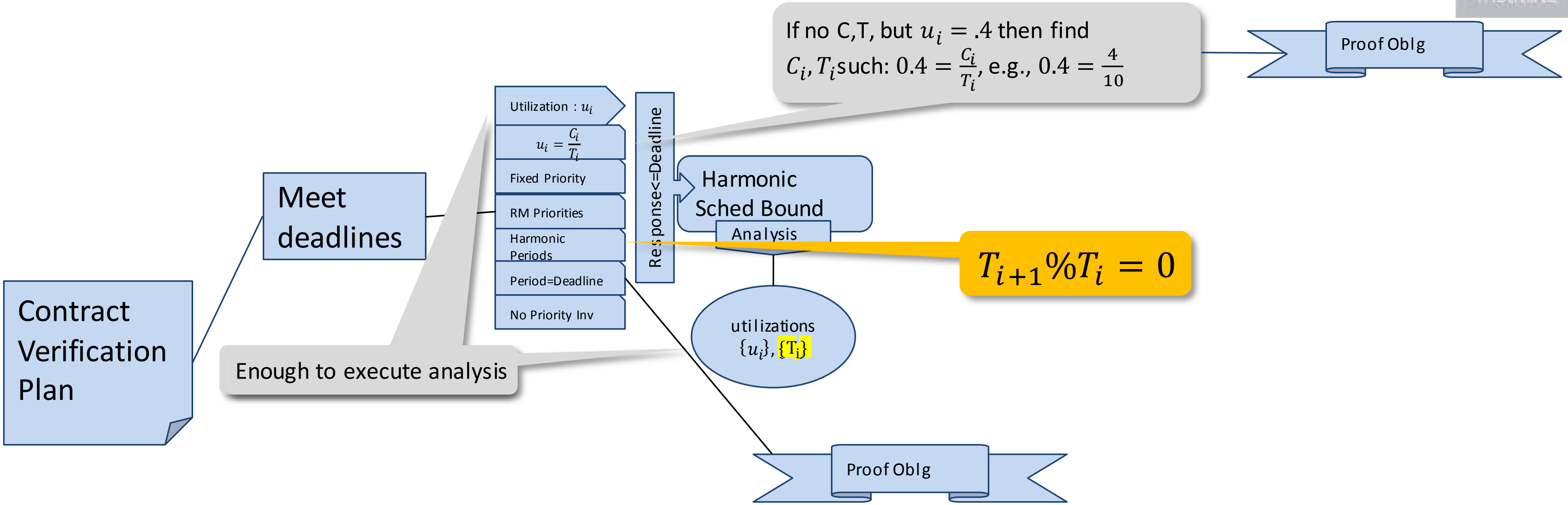
Listing 2: Path Selection Based on Refinement



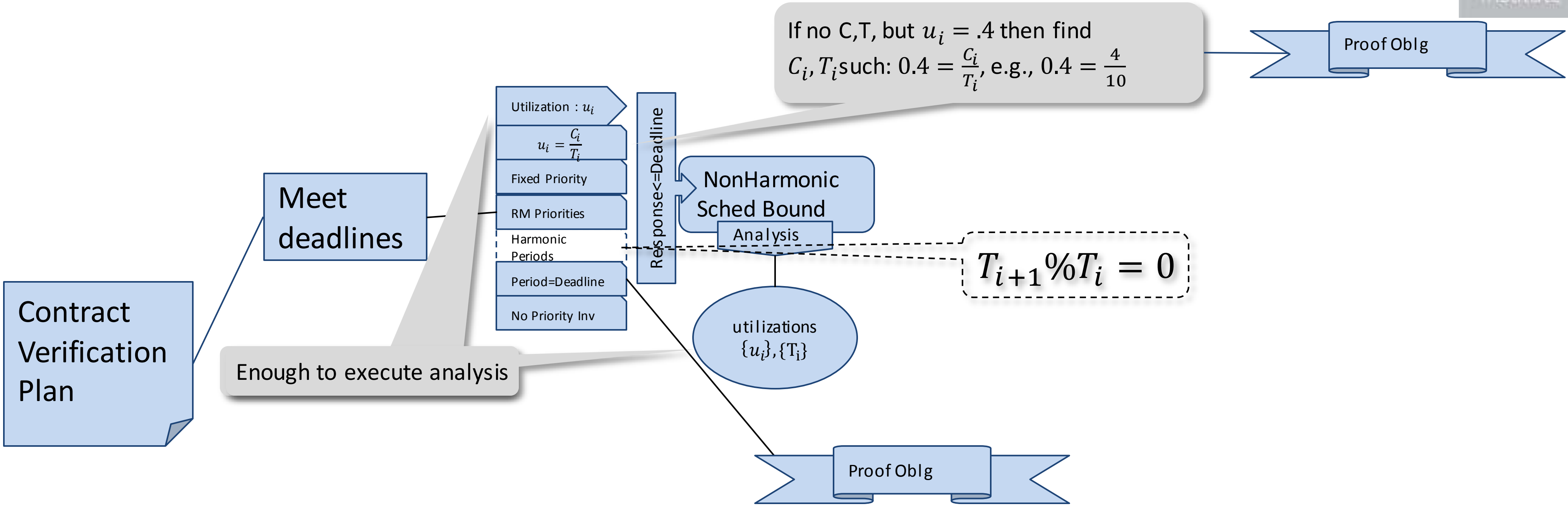
Refinement throughout development (1)



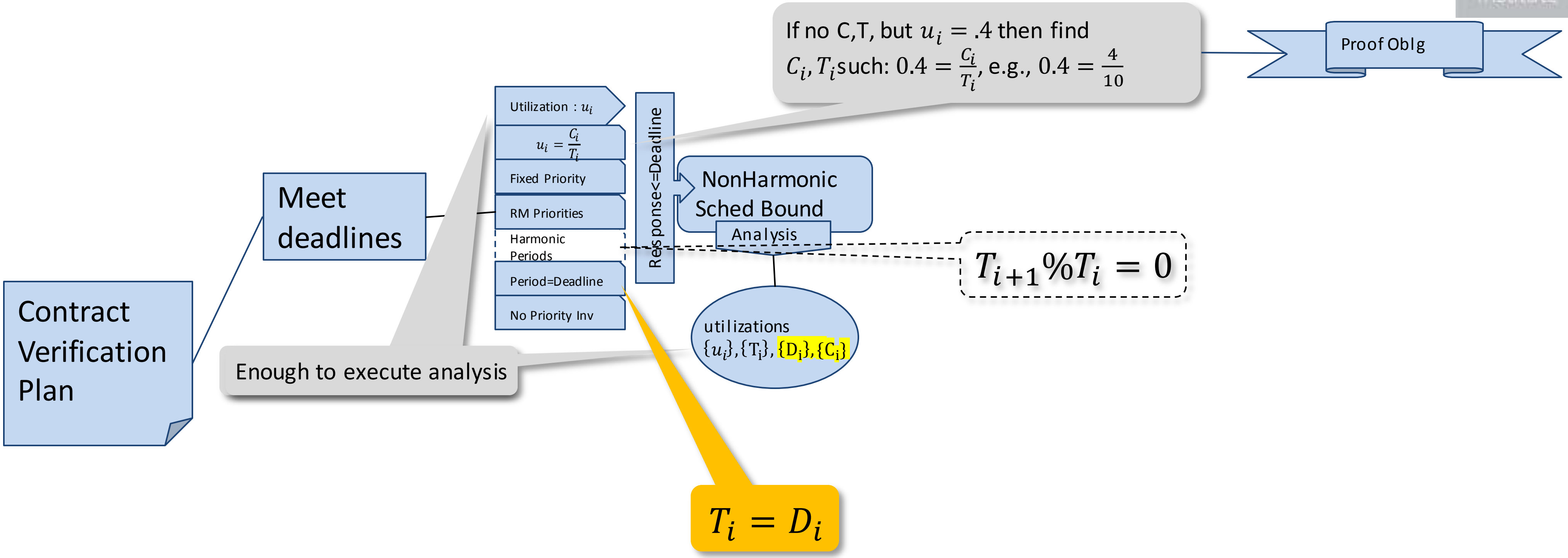
Refinement throughout development (2)



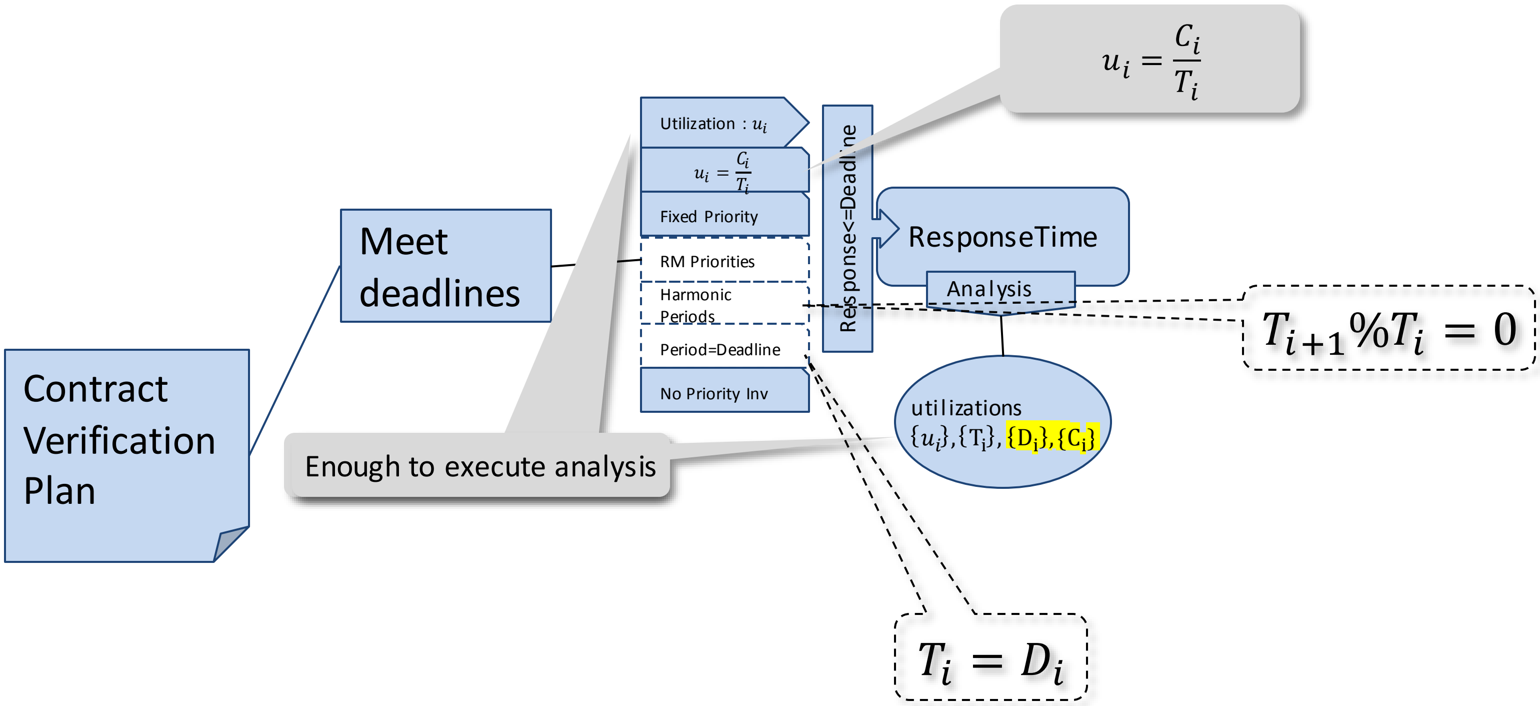
Refinement throughout development (3)



Refinement throughout development (4)



Refinement throughout development (5)



Engagements

Army System Readiness Directorate

- Implement Airworthiness Qualification Plan as contract argumentation
- Track evidence that prove claims in the plan connected to analyzed models
  - What verification procedures are used with what data
- Find where the verification failed
  - What verification procedure, what data, why it failed (e.g., assumption not met).

Assurance Evidence for Continuously Evolving Real-Time (ASERT) workgroup

- Modeling triple computer failure of Airbus A330 incident in Taiwan in 2020
- Analysis that discover the flaw and violate assumptions of the design
- Working on creating argumentation that prevents failure from happening

Other

Concluding Remarks

Large Scale Assurance

- Key to keep the pace of software evolution
- Addresses assurance of continuously increasing complexity

Assurance Argumentation Automation

- Formalizes and enables combination of verification procedures and assumption verification
- Encoded as Constraint-Satisfaction Problem (SMT) for automatic evaluation
- Enables Incremental Refinement
  - unassigned variables take satisfying values (proof obligations)
- Encodes alternative contracts based on refinement/design

Takes Advantage of Previous Analyses Proofs / Efficient Implementations