RESEARCH REVIEW 2022

Carnegie Mellon University Software Engineering Institute

Maturing Assurance Contracts in Model-Based Engineering

NOVEMBER 14–16, 2022

Dionisio de Niz
Technical Director, Assuring Cyber-Physical Systems

Carnegie Mellon University Software Engineering Institute

DoD Digital Engineering Challenges

Model-Analyze-Build

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

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

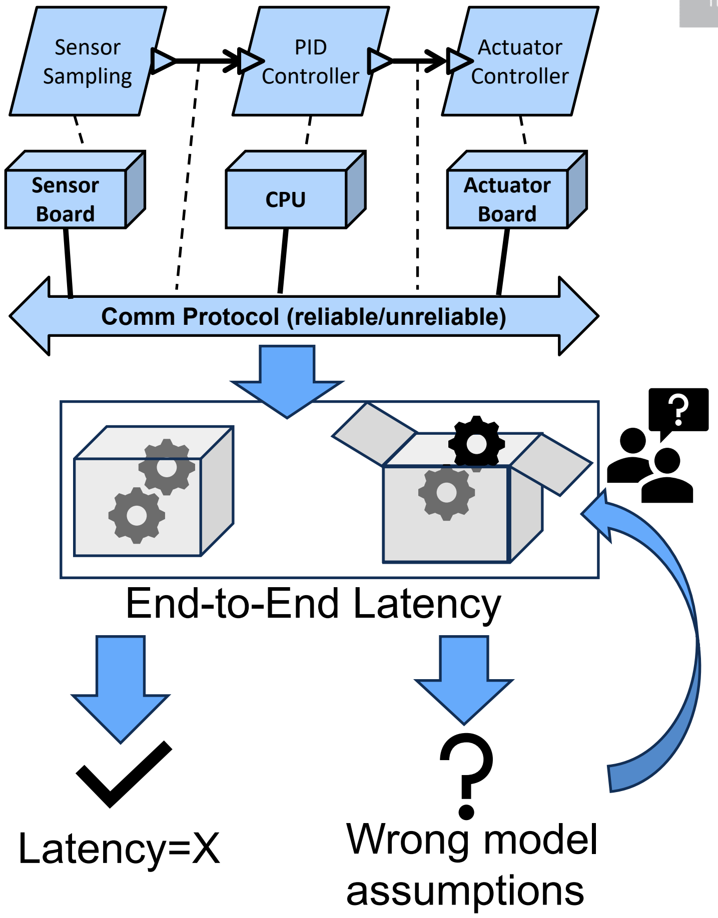
BUT:

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

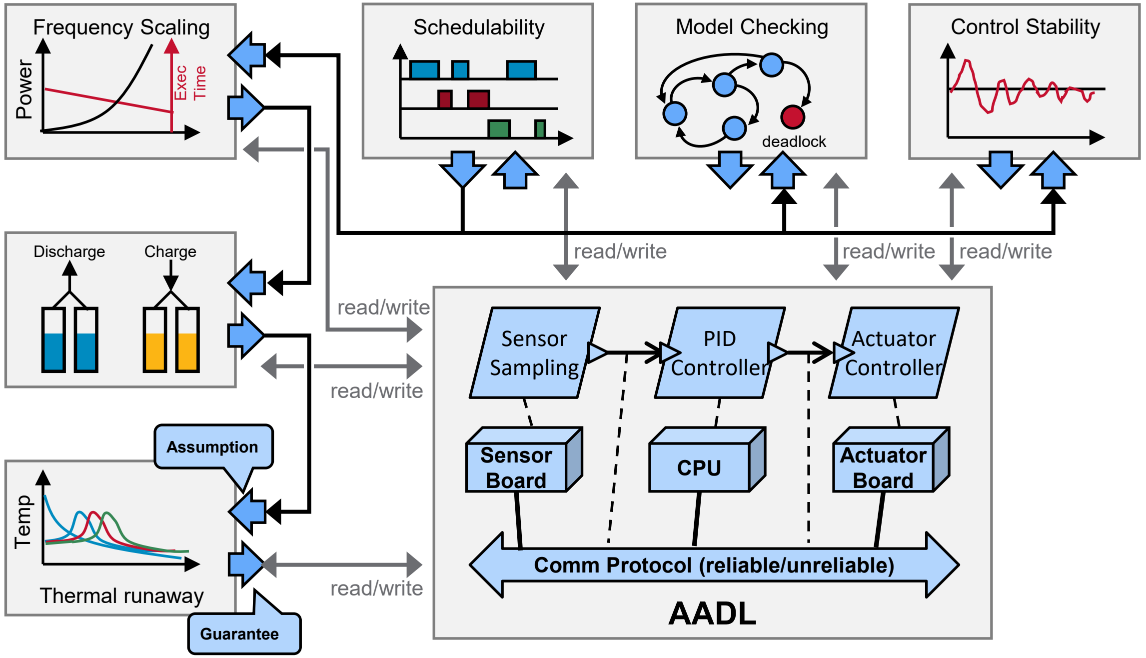


Digital Engineering: Multiple Claims – Multiple Analyses

Different Assurance Claims

- Combine multiple analysis
- Validate assumptions
- Resolve assumption conflicts

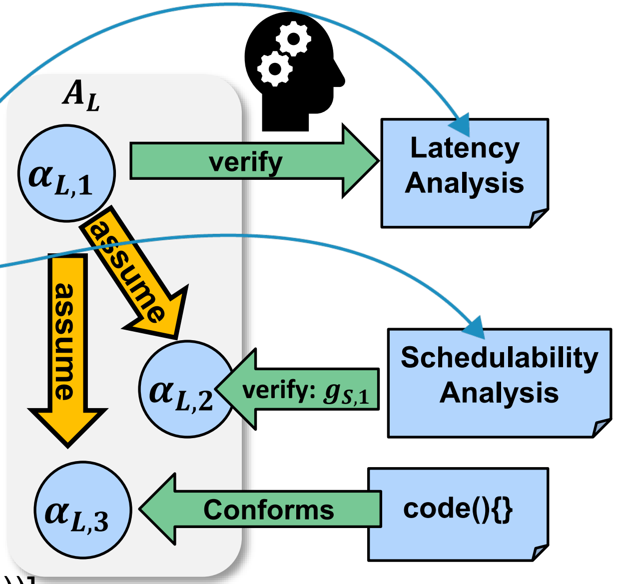
Integrate into arguments to satisfy claims



Analysis Contract: Tracking Assumptions and Guarantees

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

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



Shift Left And Down to the Implementation

Early Analysis

- Evaluate design decisions with partial information
- E.g., latency analysis before periods
  - periods of tasks must be multiples of each other

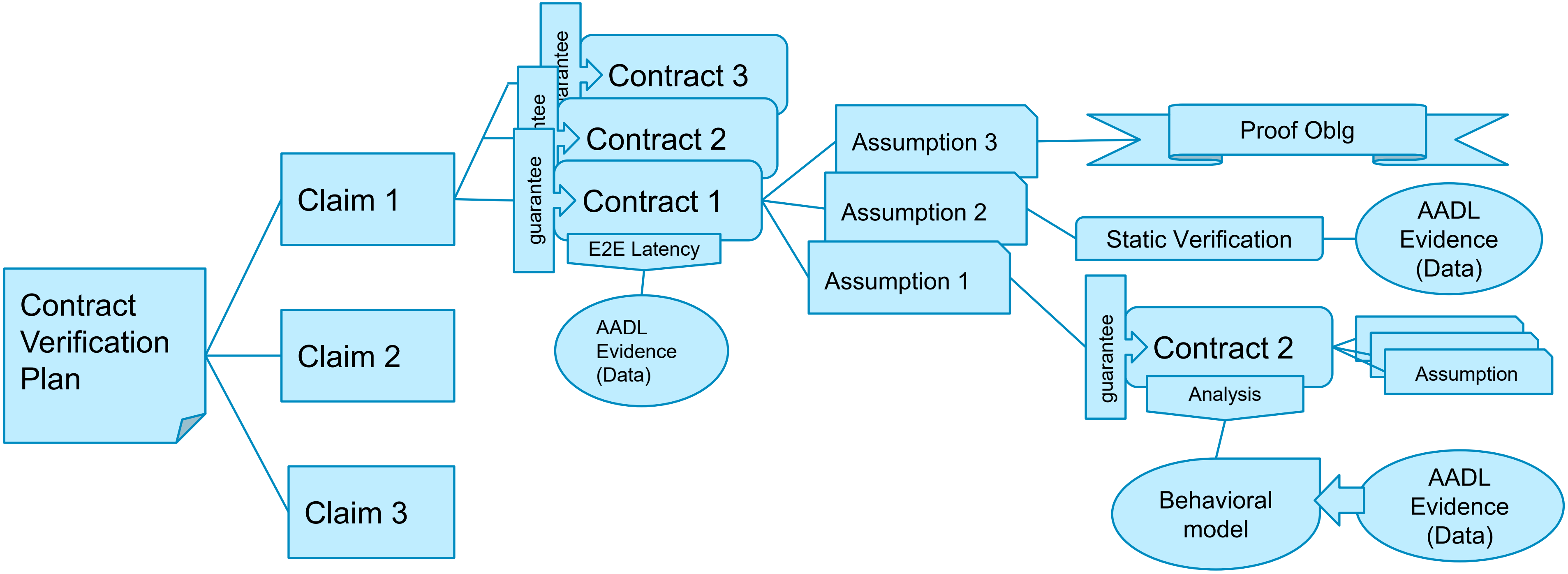
Refinement

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

Conformance

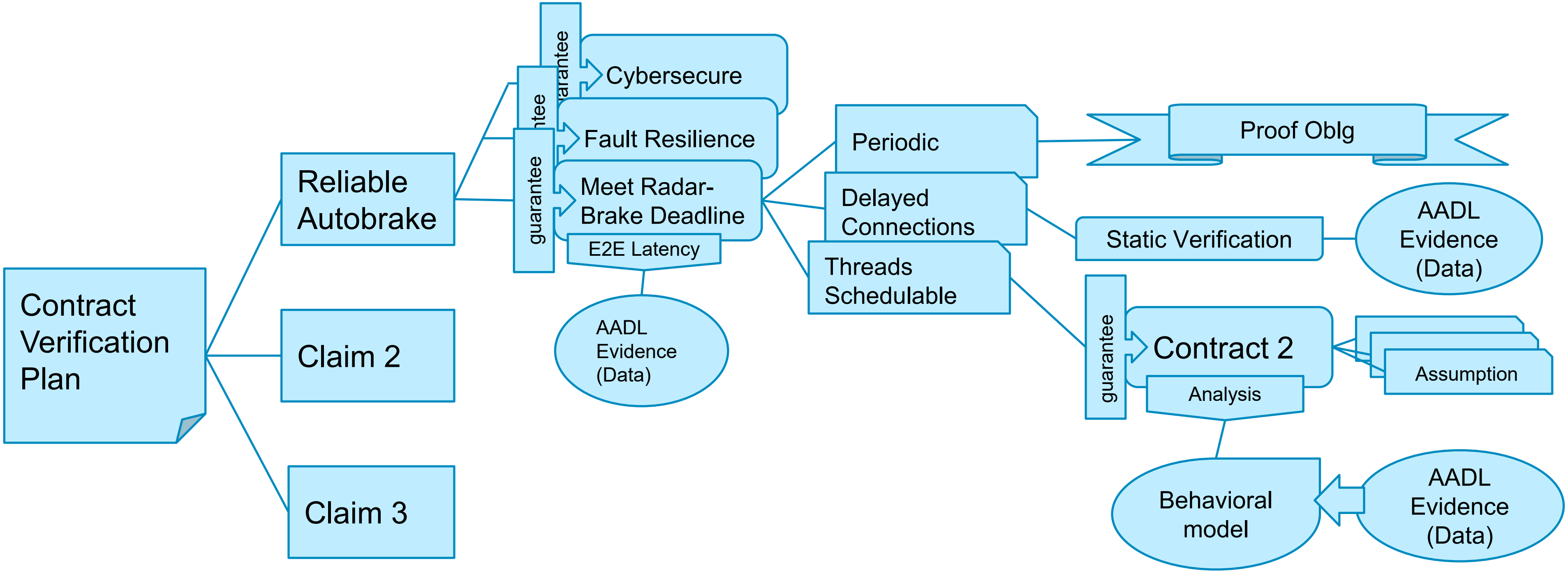
- Track implementation assumption
- Verify implementation conformance
  - Task executed strictly periodic $\alpha_{L,3}$

Assurance Contract Argumentation (1)



Carnegie Mellon University
Software Engineering Institute

Assurance Contract Argumentation (2)



Contract Argumentation in the Development Lifecycle

Integrity of Analysis

- Verify assumptions
  - Detect violation
  - Suggest repairs
- Offer alternative analysis that satisfy assumptions

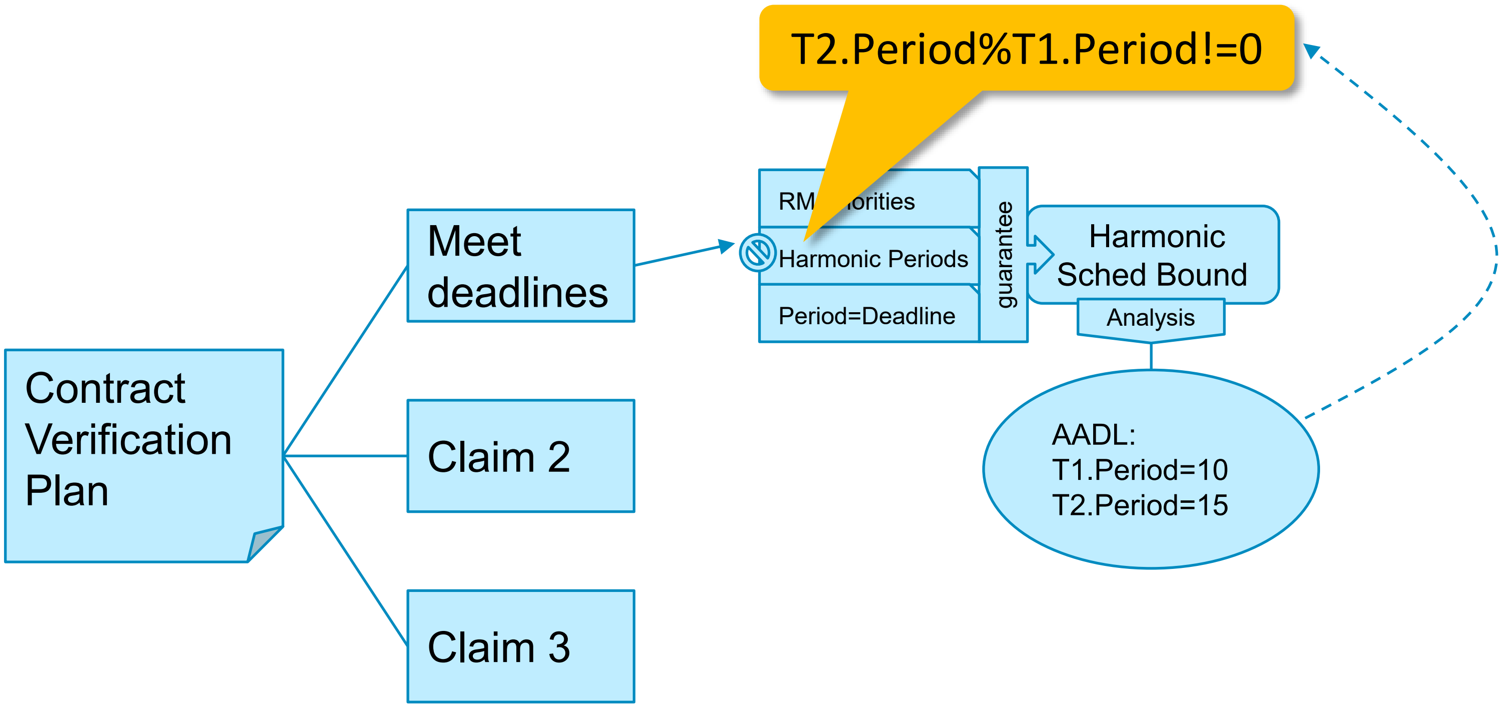
Refine Design

- When enough new data for new analysis
- When new data affects proof obligations

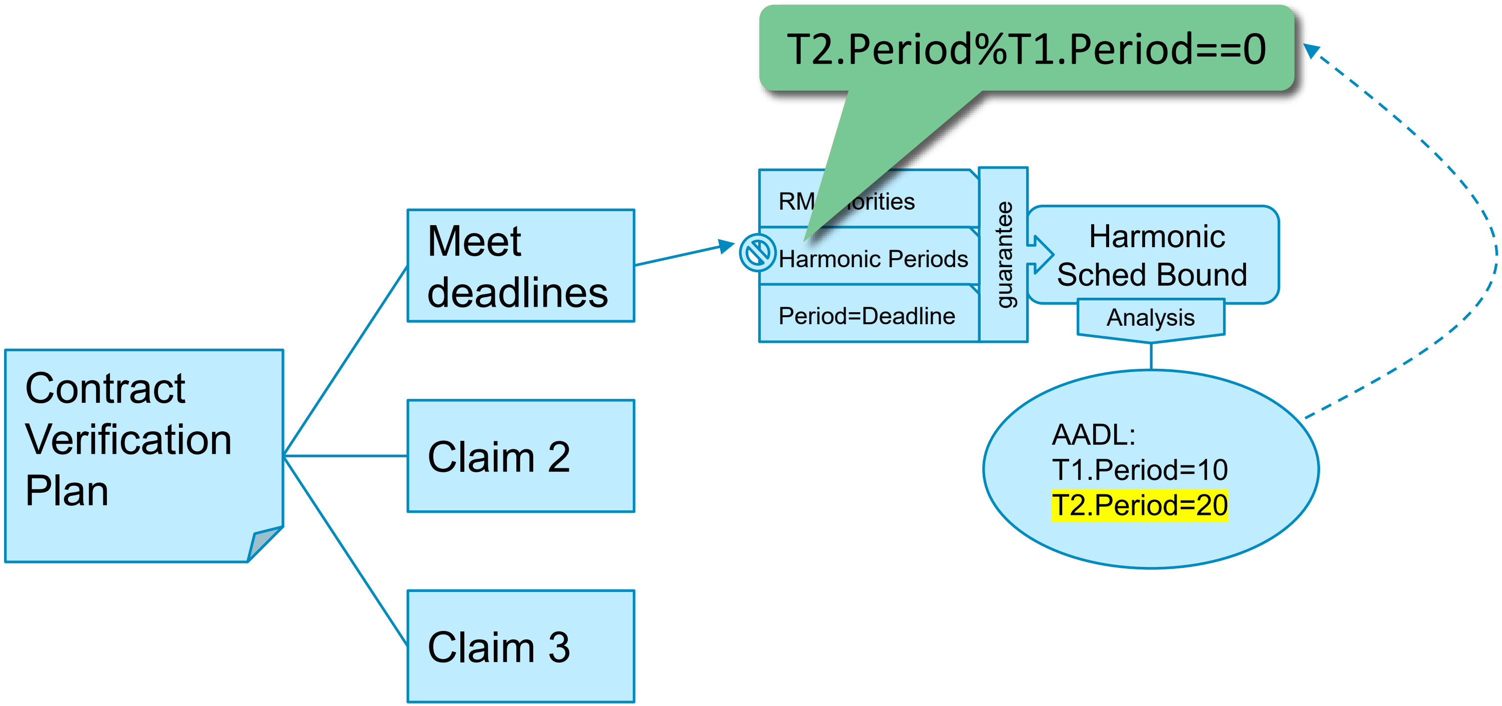
Argument reusability

- Self-contained modular analysis contracts

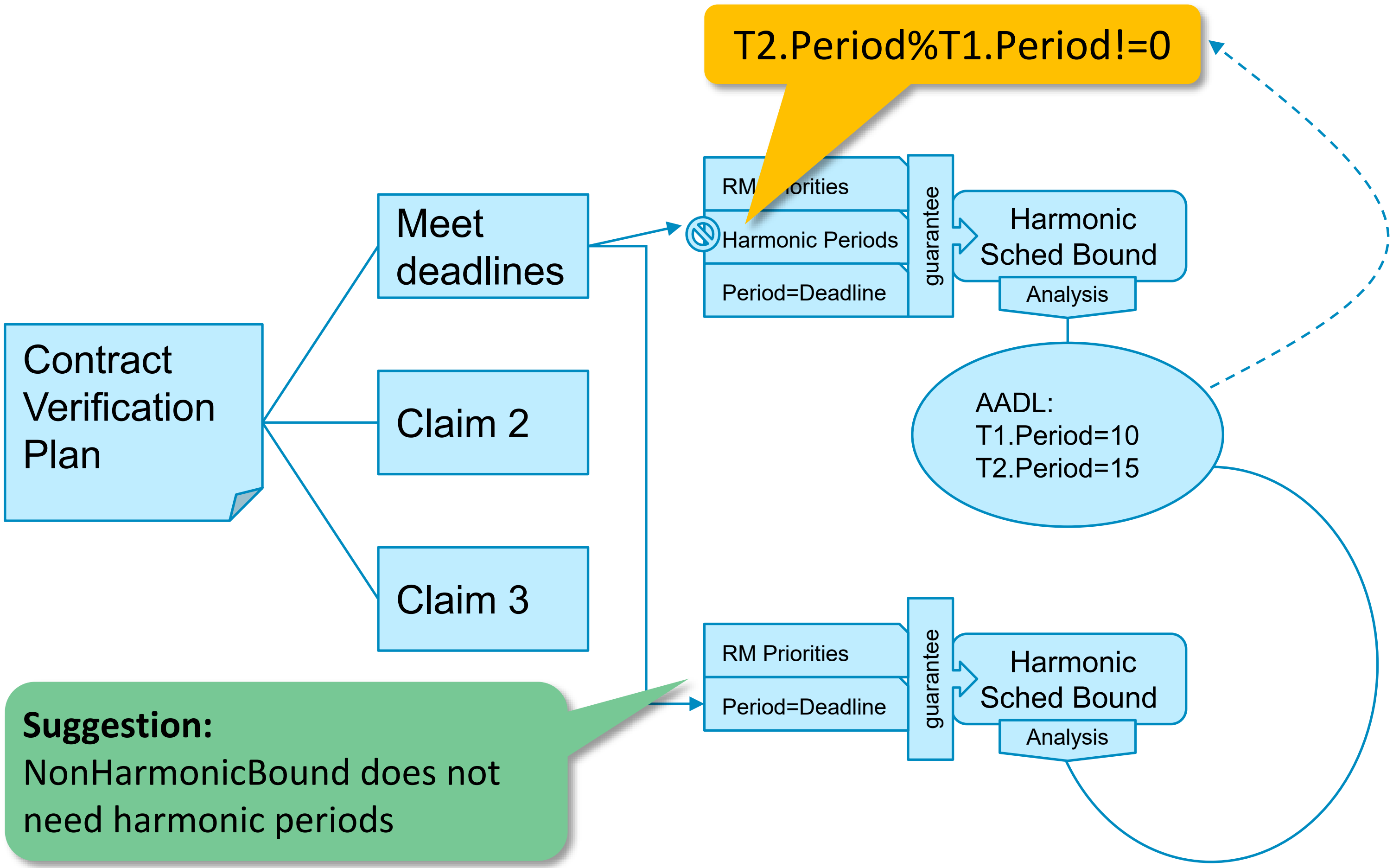
Integrity of Analysis: Repairing Assumptions (1)



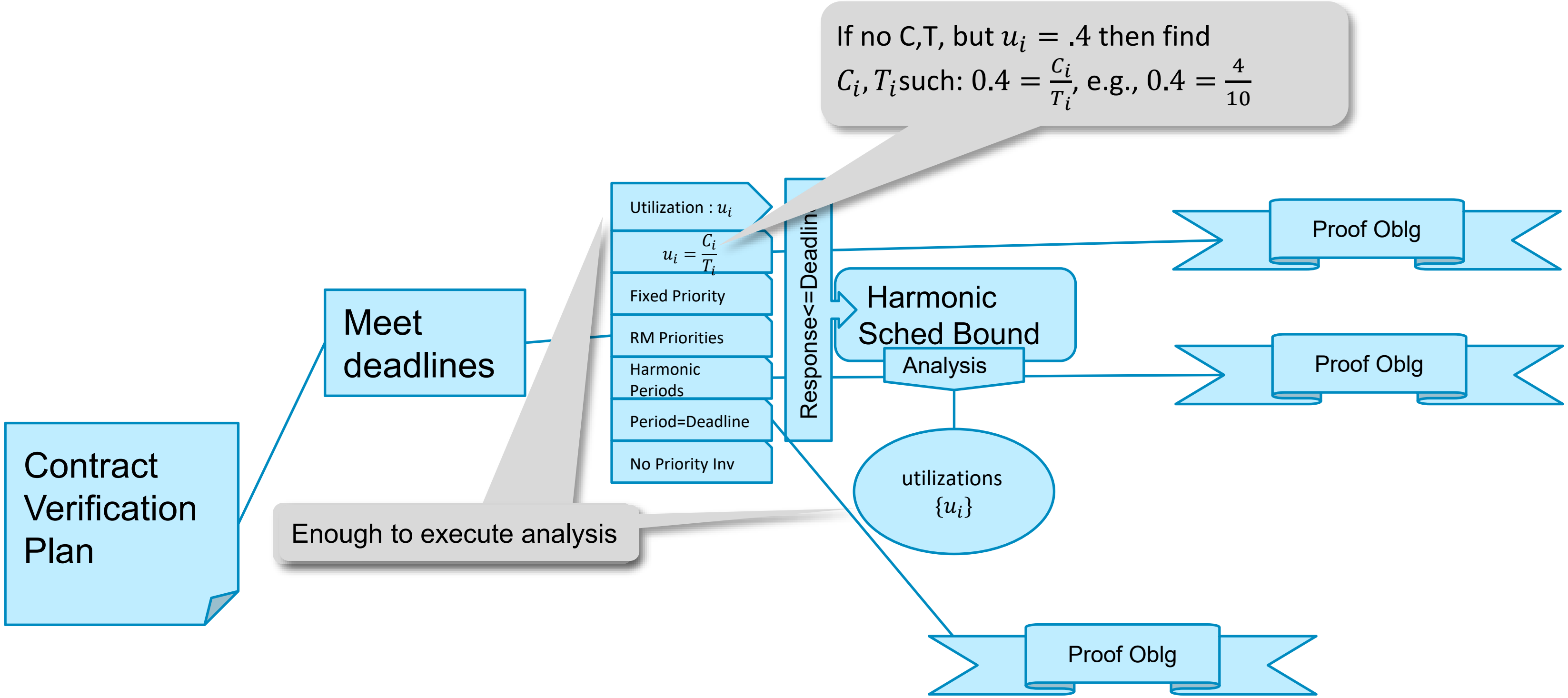
Integrity of Analysis: Repairing Assumptions (2)



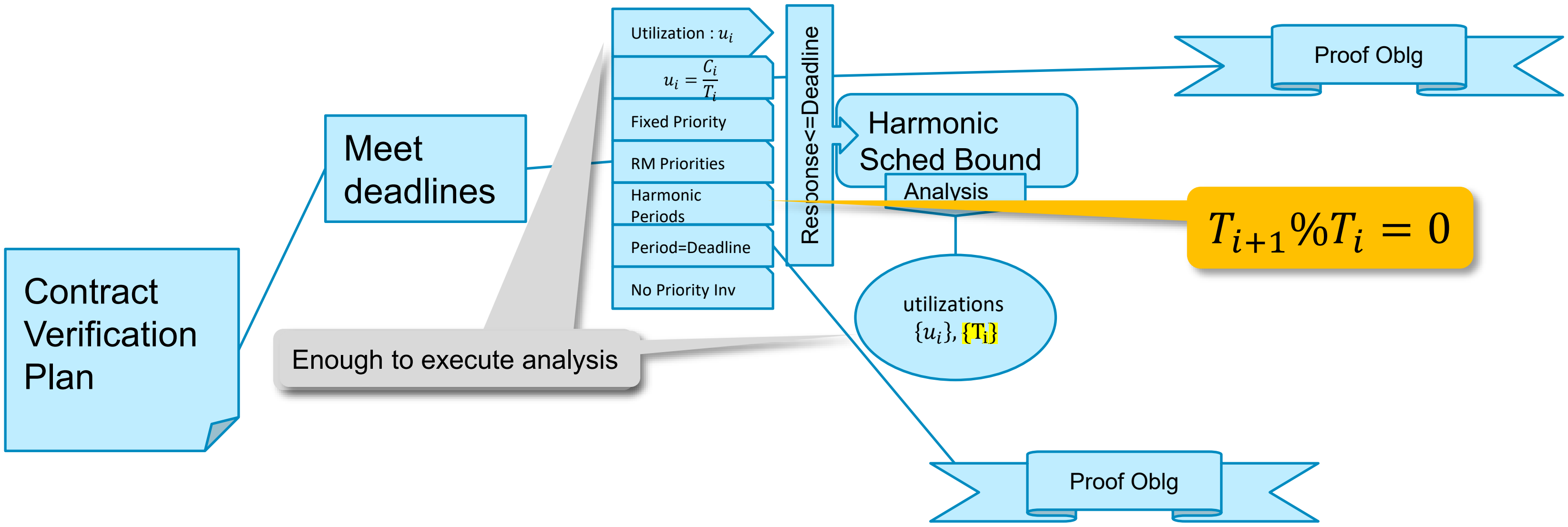
Integrity of Analysis: Repairing Assumptions (3)



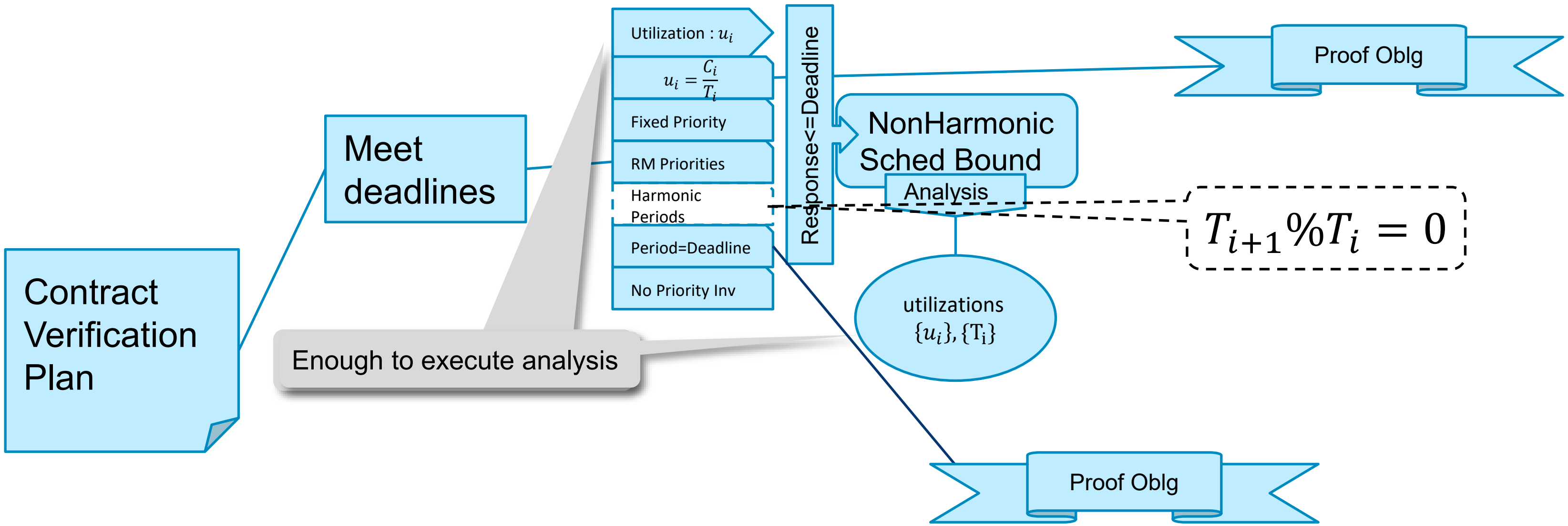
Refinement Throughout Development (1)



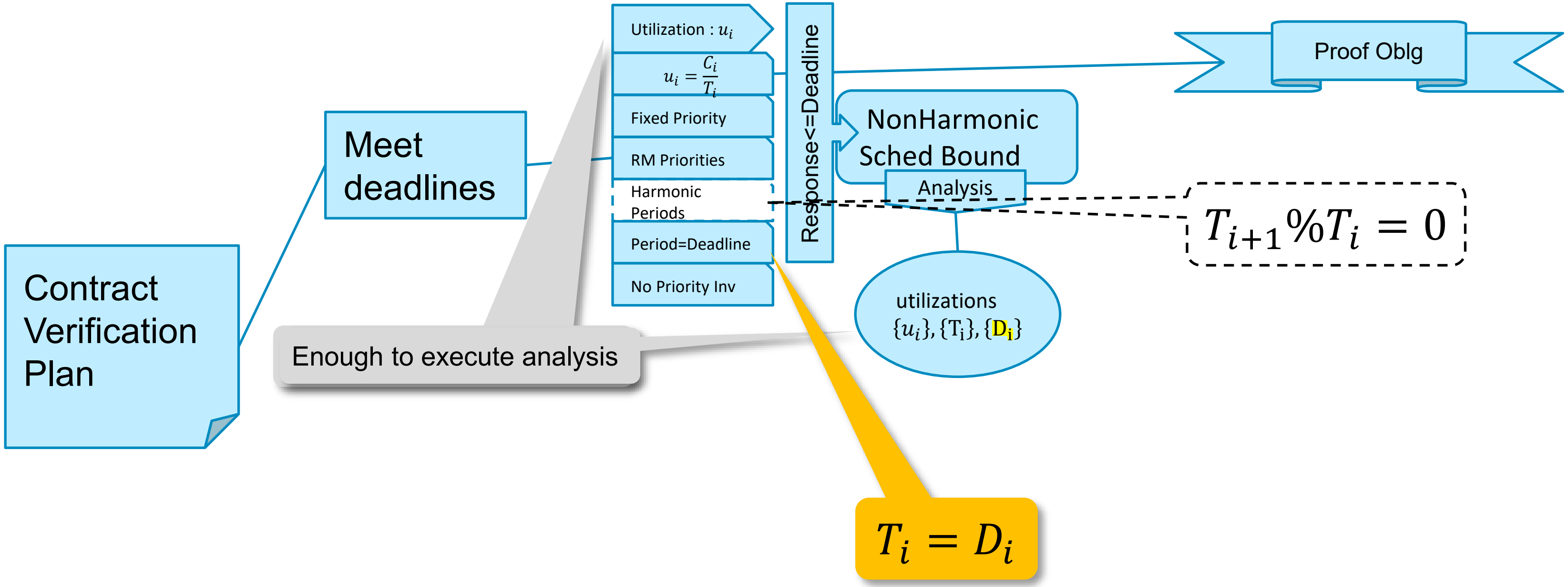
Refinement Throughout Development (2)



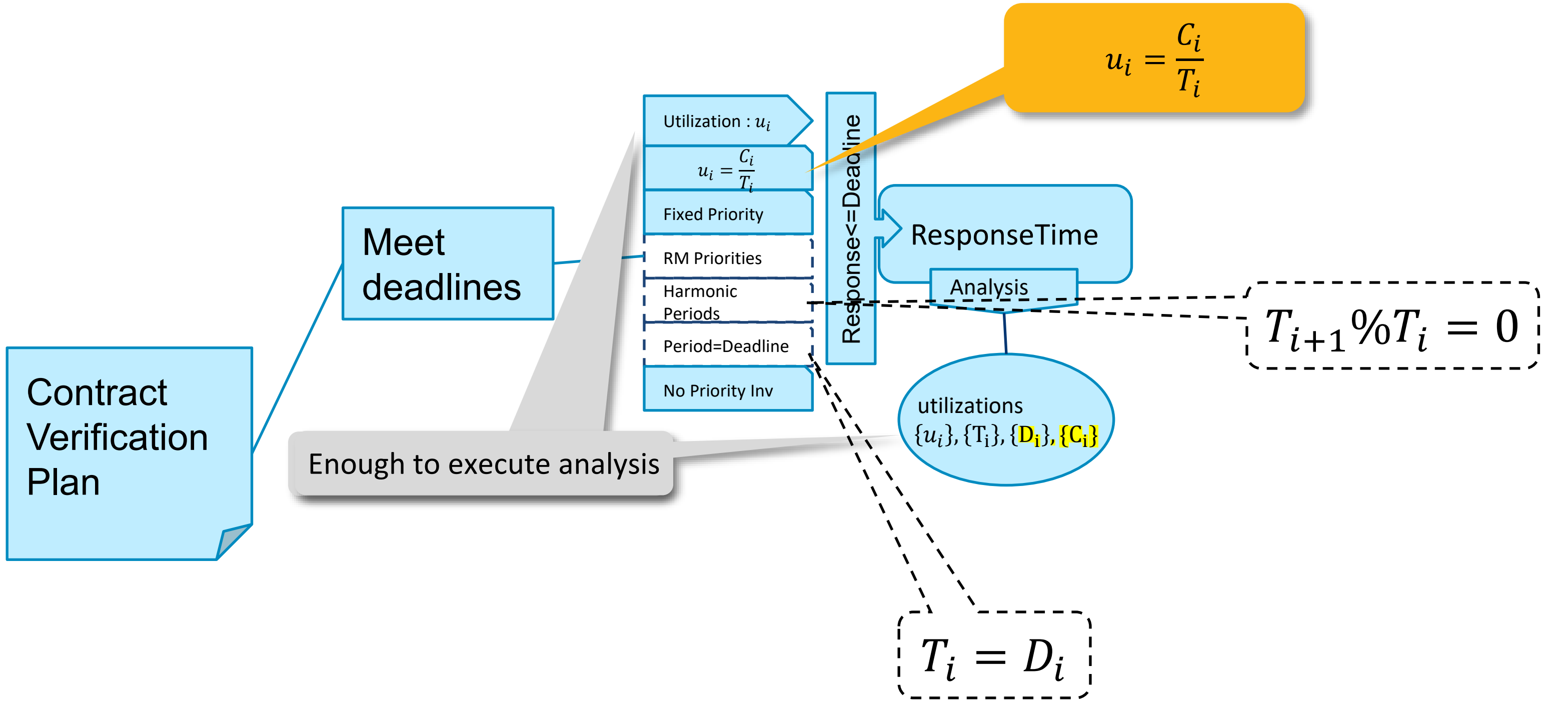
Refinement Throughout Development (3)



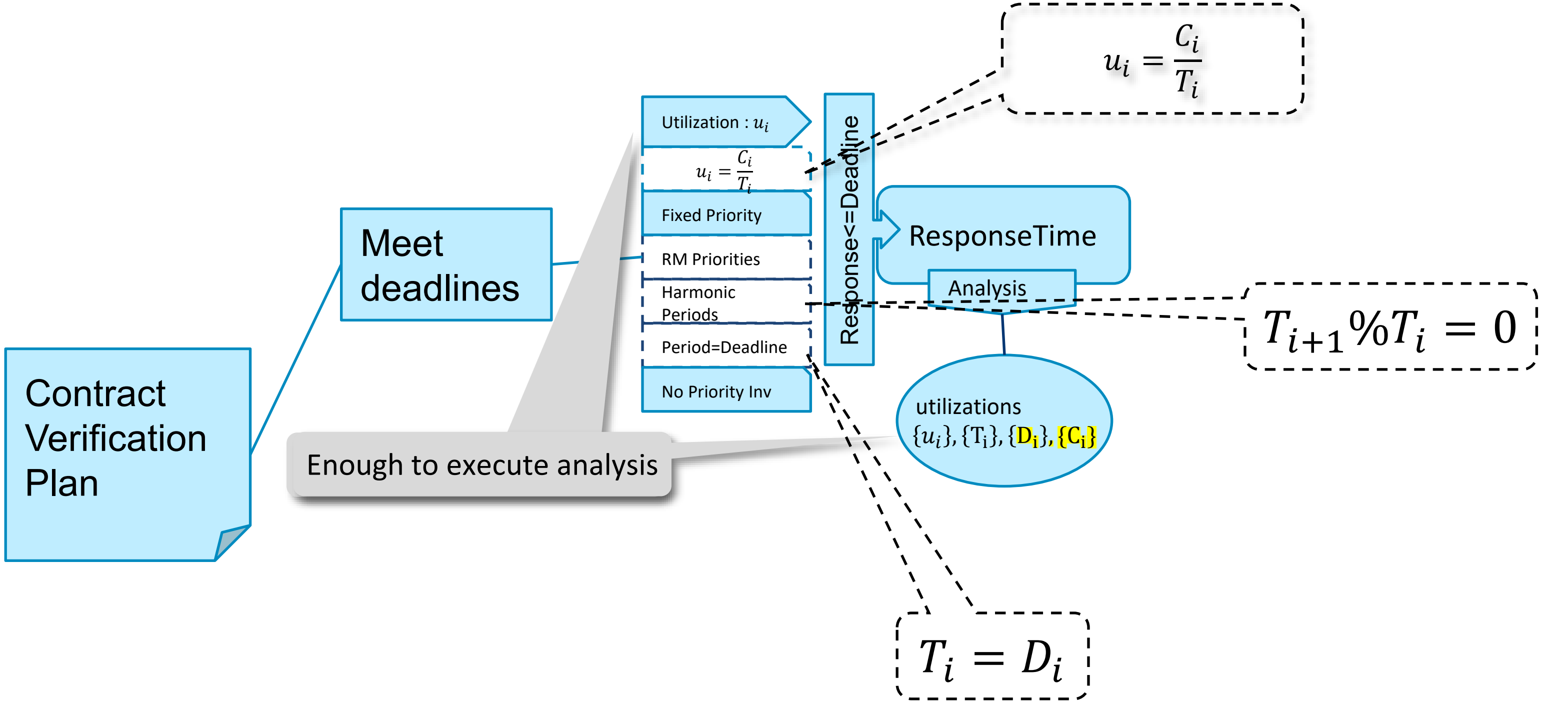
Refinement Throughout Development (4)



Refinement Throughout Development (5)

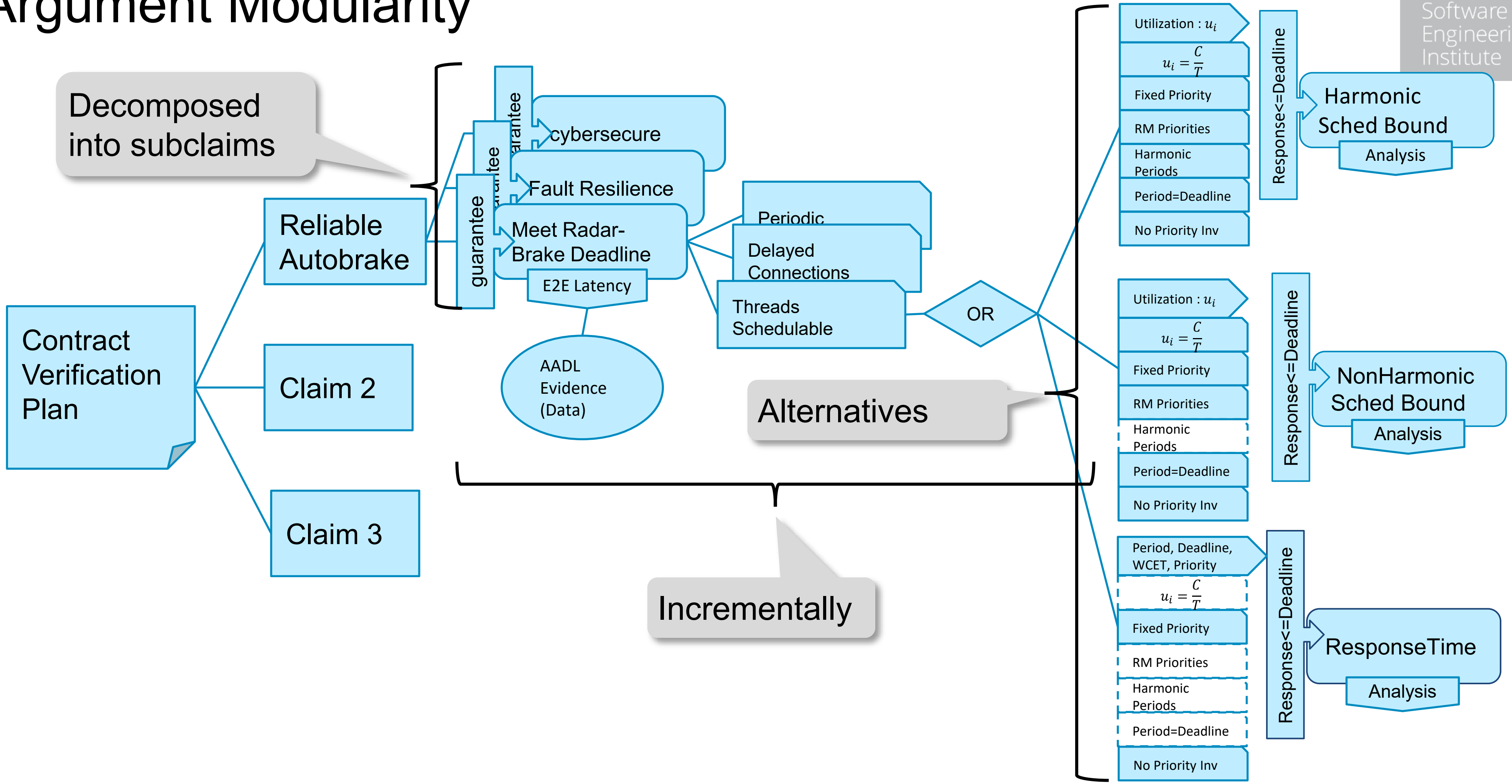


Refinement Throughout Development (6)



Carnegie Mellon University Software Engineering Institute

Argument Modularity



Implementation: 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
- Constraint Satisfaction Solver (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)

Artifacts
- Annex language hosted in AADL/OSATE
  - Model Query Language adaptable for multiple modeling language (e.g., SysML V2)
- Automatic Execution of Contract Verification Plan
  - Assumption repair & analyses alternatives

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
- Enables connection 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

Impact

Certification

- Automated and sound verification of assurance claims through models

Fielding Speed

- Automated assurance watchdog
  - Validate incremental refinement through design
- Concurrent formal assurance argument construction and system development

Digital Engineering and AADL Ecosystem

- Incremental sound analysis infrastructure to DoD modeling efforts
- Architecture-Centric Virtual Integration Practice (ACVIP) within FVL
- DARPA programs using AADL

Concluding Remarks

Certification in Digital Engineering Era

- Follow model-analyze-build
  - Automated argumentation supported by model-based analysis contracts

Shift Left

- Start verifying early design decisions
- As design is refined
  - Ensure properties of pervious design decisions are preserved
  - New refinements can provide additional evidence / properties to support assurance

Down To The Implementation

- Drive properties & assumptions down to the implementation

Scalable

- Exploit efficient analysis from different domains

Sound

- Exploit advances in formal verification
  - Combination of analysis, verification of assumptions, implementation compliance, analysis correctness

Carnegie Mellon University Software Engineering Institute

Team

Bjorn Andersson
Principal Researcher

Jerome Hugues
Senior Researcher

Sam Procter
Senior Researcher.

John Hudak
Principal Engineer

Sholom Cohen
Principal Engineer

Lutz Wrage
Senior Researcher

Aaron Greenhouse
Senior Researcher

Ruben Martins
Research Professor CMU/CS

Gabriel Moreno
Principal Researcher

Joseph Seibel
Software Engineer

Dionisio de Niz
Technical Director ACPS.

Carnegie Mellon University Software Engineering Institute

Email us at: info@sei.cmu.edu

To learn how the results of our project can help improve the effectiveness of your systems' assurance.

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM22-0869