Analysis Contracts for Cyber-Physical Systems

Presenter: Dionisio de Niz[1]
dionisio@sei.cmu.edu

S5 Symposium. June 10-12, 2014

Team: Ivan Ruchkin[2], Sagar Chaki[1], David Garlan[2]

[1]Software Engineering Institute &
[2]School of Computer Science

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0001369

Motivation

The development of Cyber-Physical Systems (aircrafts, cars, trains, robots, etc.) increasingly relies on many types of analyses from different disciplines for assurance purposes

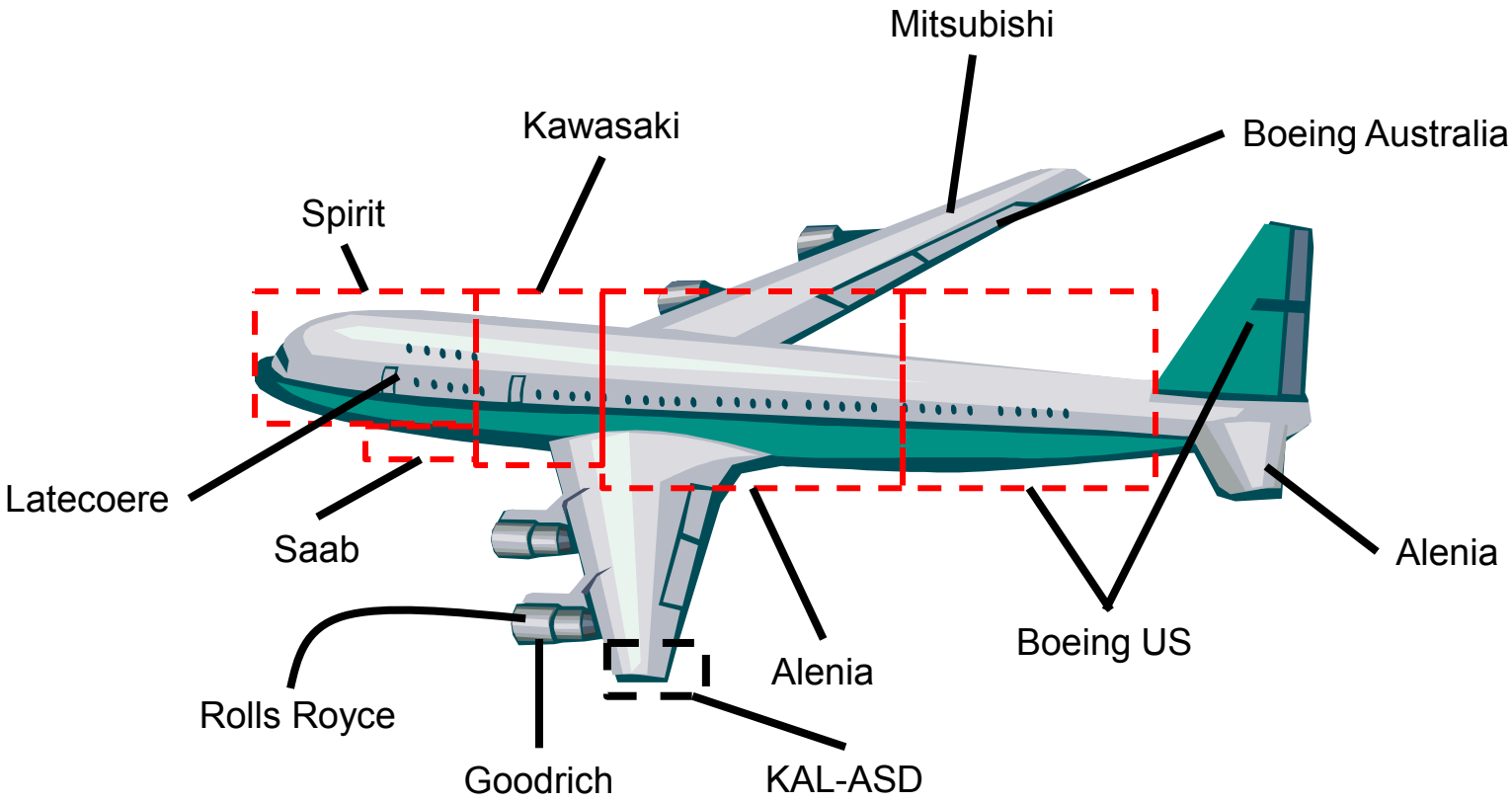
- Control stability, scheduling, logic, thermal, power, aerodynamics, etc.

Large CPS are integrated out of components developed by suppliers that use their own analysis methods and make their own assumptions

Analysis assumption mismatches are discovered late in the system integration phase

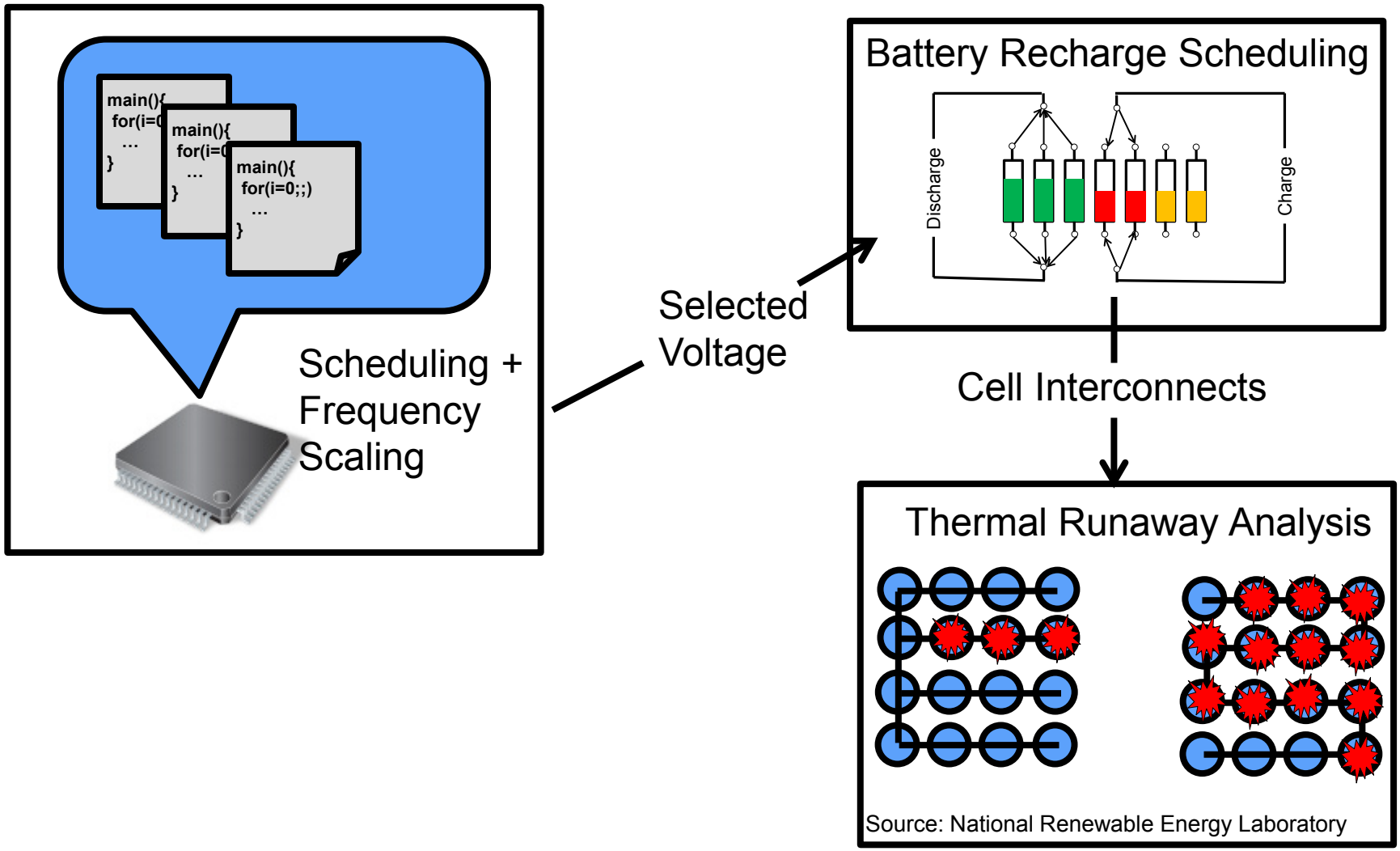
- Difficult and costly to solve

Boeing 787 Suppliers

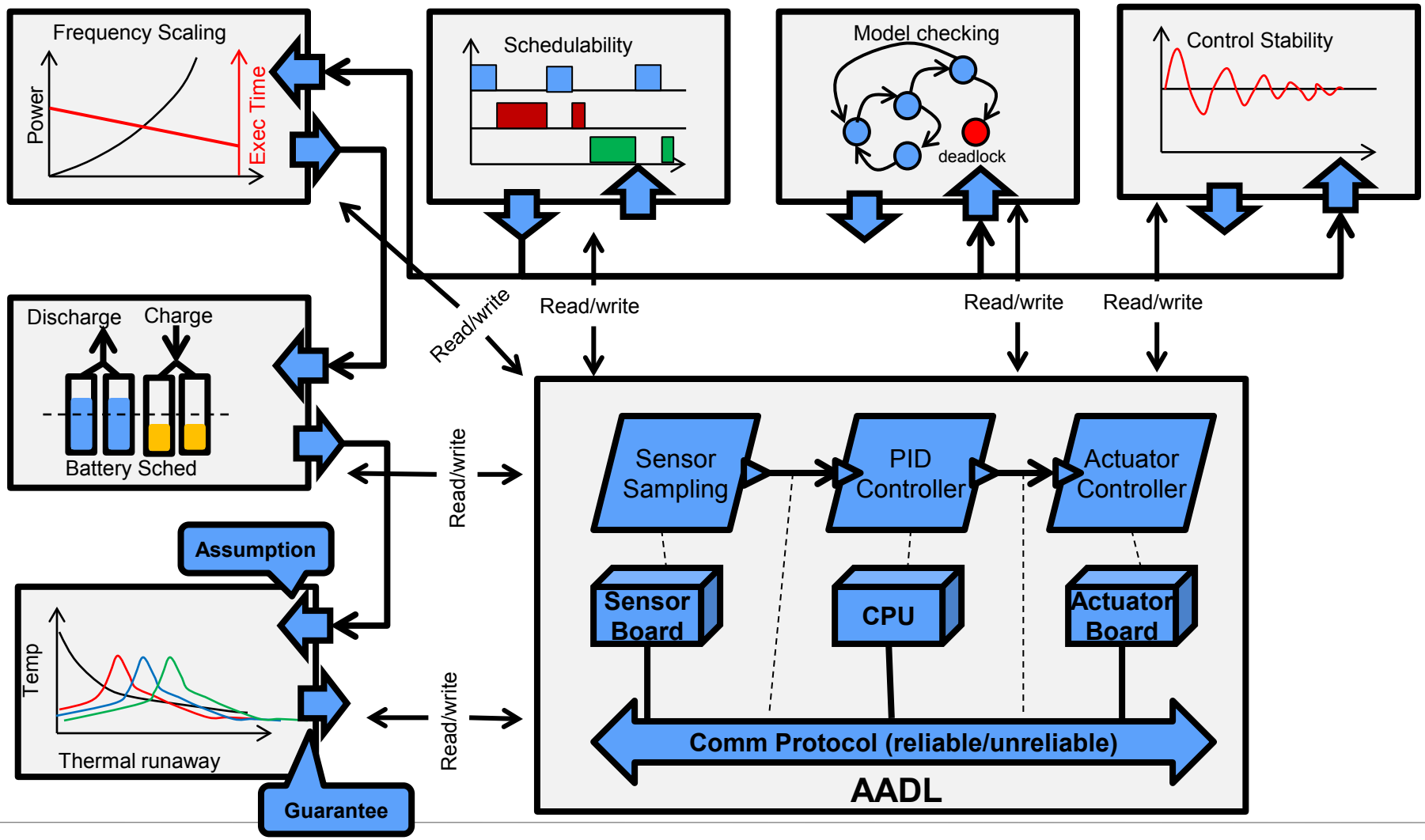


Source: Boeing / Reuters

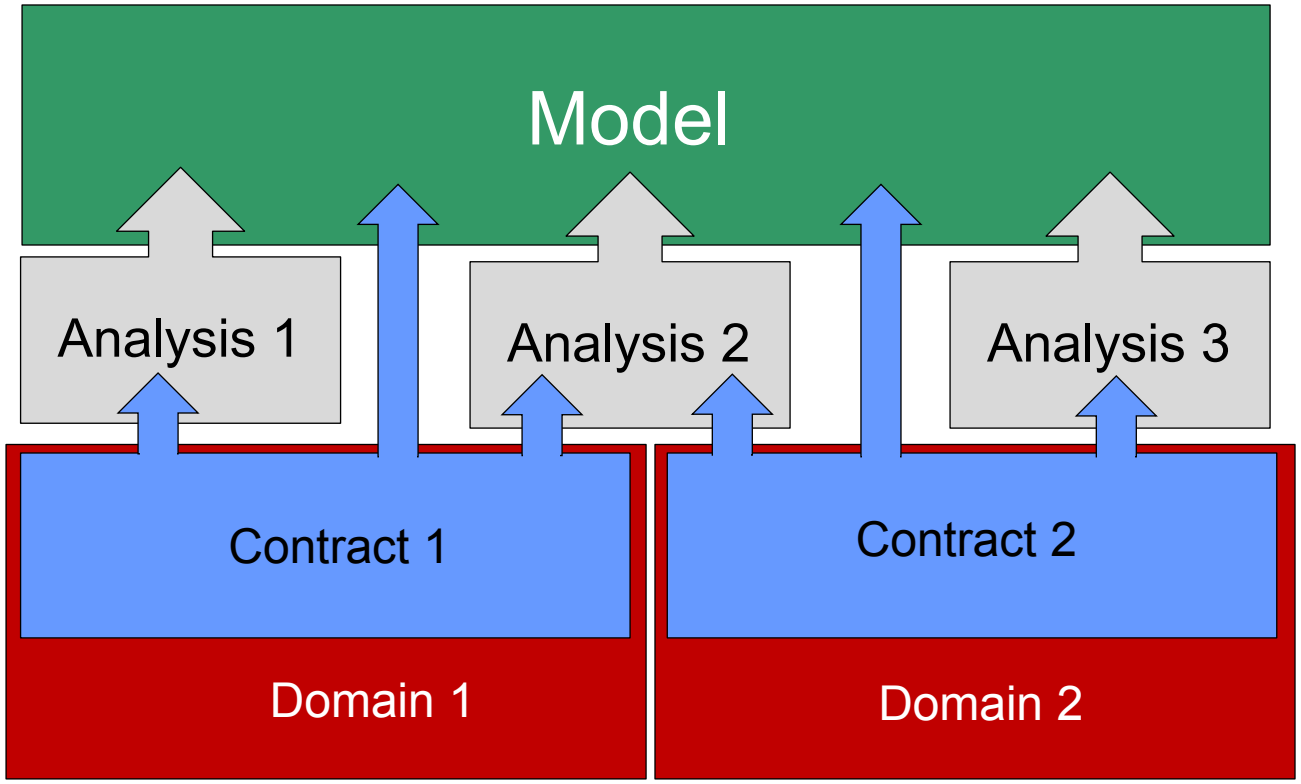
Analyses Interactions



Analysis Contracts



Analysis Contract Scheme



Contract Semantic Basis

Domain $(\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, [\![\cdot]\!]_\sigma)$

- Sorts $\mathcal{A} = \{A_1, A_2, \dots\}$, e.g. Booleans, Integers, Threads, Priorities
- Static Properties : $\mathcal{S} = \{S_i\},\ S_i: A_1 \times \cdots \times A_j \mapsto A_k$
  - Design-time invariants, Regular operators: $(\wedge, \mathcal{B} \times \mathcal{B} \mapsto \mathcal{B})$
- Runtime Properties : $\mathcal{R} = \{R_i\},\ R_i: A_1 \times \cdots \times A_j \mapsto A_k$
  - Evolving valuation at different states q: $\mathrm{q}(R_i)$
- Domain execution semantics: $\mathcal{T}$
  - Infinite sequence of assignments to runtime properties (executions)
- Domain interpretation of sorts/static properties: $[\![\cdot]\!]_\sigma$
  - E.g. allowed schedulers, some left **uninterpreted**

Architectural Model Interpretation: $[\![\cdot]\!]_M$ on $\mathcal{A}, \mathcal{S}, \mathcal{T}$

- E.g. threads and periods: $[\![T]\!]_M = \{t_1, t_2\}; [\![Per]\!]_M = \{t_1 \mapsto 10, t_2 \mapsto 20\}$

Executions of system defined by M: $[\![\mathcal{T}]\!]_M$

- Combining $[\![\cdot]\!]_\sigma, [\![\cdot]\!]_M$ into $[\![\cdot]\!]$
- Each state q in possible states Q maps R_i to function $q(R_i)$: $[\![A_1]\!] \times \cdots \times [\![A_j]\!] \mapsto [\![A_k]\!]$
- With all infinite sequence of states Q^ω
- $[\![\mathcal{T}]\!]_M \subseteq Q^\omega$

Contract Language

Contract formulas

- Given domain $\sigma = (\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{T}, [\![\cdot]\!]_\sigma)$,
- $\mathcal{F}_\sigma ::= \forall\, v_1, \ldots, v_j \cdot \phi \mid \exists\, v_1, \ldots, v_j \cdot \phi \mid \forall\, v_1, \ldots, v_j \cdot \phi : \psi \mid \exists v_1, \ldots, v_j \cdot \phi : \psi$
  - $v_i : A_i$, ϕ: static (first order) formula
  - ψ : LTL formula

Contract $C = (I, O, A, G)$

- $I \subseteq (\mathcal{A} \cup \mathcal{S})$: Sorts and properties read by the analysis
- $O \subseteq (\mathcal{A} \cup \mathcal{S})$: Sorts and properties written by the analysis
- $A \subseteq \mathcal{F}_\sigma$: assumptions: must be true in input
- $G \subseteq \mathcal{F}_\sigma$: guarantees: must be true in output

Contract Verification

Given model M and set of analyses $\mathcal{AN} = \{An_i\}$:

- For $An_i.C = (I, O, A, G)$: M application to An_i iff
  - $\forall\, a \in A \cdot M \vDash a,\ \ \forall\, g \in G \cdot \mathrm{An_i(M)} \vDash g$

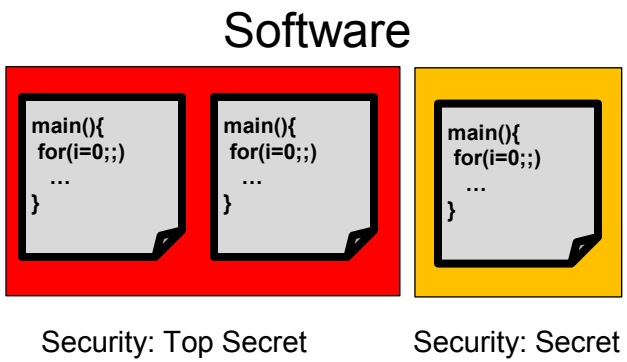
Valid analysis ordering: no dependencies from later analysis

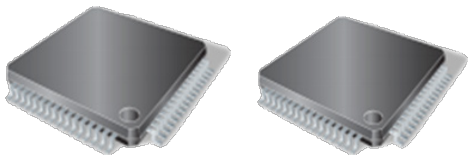
- Contract (& analysis) dependency: $\mathrm{d(C_i, C_j)}: \mathrm{C_i.I} \cap C_j.O \neq \emptyset$

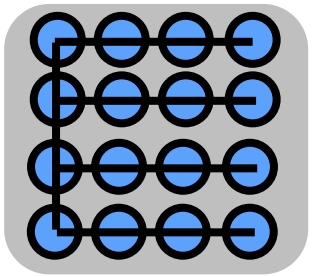
Contract Formulas

- First order: in SMT (Z3)
- LTL : Model checker
- FOL + LTL: Generate all solutions for FOL, check LTL in each

Example: Surveillance Aircraft







Analysis

Security: tasks of different level to different processor

Scheduling: meet all deadlines

Freq. Scaling: minimize power

Logic: no deadlocks or race conditions

Battery scheduling: meet battery lifetime

Battery thermal: no runaways

Surveillance Aircraft Contracts

Security Analysis

- $An_{sec}.C: I = \{T, ThSecCl\}, O = \{NotColoc\}, A = \emptyset, G = \{g\}$
  - $g: \forall\, t_1, t_2 \cdot ThSecCl(t_1) \neq ThSecCl(t_2) \Rightarrow t_1 \in NotColoc(t_2)$

Multiprocessor scheduling: (Binpacking + scheduling)

- $An_{sched}.C: I = \{T, C, NotColoc, Per, WCET, Dline\}, O = \{CPUBind\}, A = \emptyset, G = \{g\}$
  - $g: \forall\, t_1, t_2 \cdot t_1 \in NotColoc(t_2) \Rightarrow CPUBind(t_1) \neq CPUBind(t_2)$

Frequency Scaling

- $An_{freqsc}.C: I = \{T, C, CPUBind, Dline\}, O = \{CPUFreq\}, G = \emptyset, A = \{a\}$
  - $a: \forall t_1, t_2 \cdot CPUBind(t_1) = CPUBind(t_2): G(CanPrmpt(t_1, t_2) \Rightarrow Dline(t_1) < Dline(t_2)$

Model checking periodic program (REK):

- $An_{rek}.C: I = \{T, C, Per, Dline, WCET, CPUBind\}, O = \{ThSafe\}, G = \emptyset, A = \{a_1, a_2\}$
- $a_1: \forall t \cdot Per(t) = Dline(t),\ \ a_2: \forall t_1, t_2 \cdot G(Canprmpt(t_1, t_2) \Rightarrow G\ \neg CanPrmpt(t_2, t_1))$

Thermal runaway:

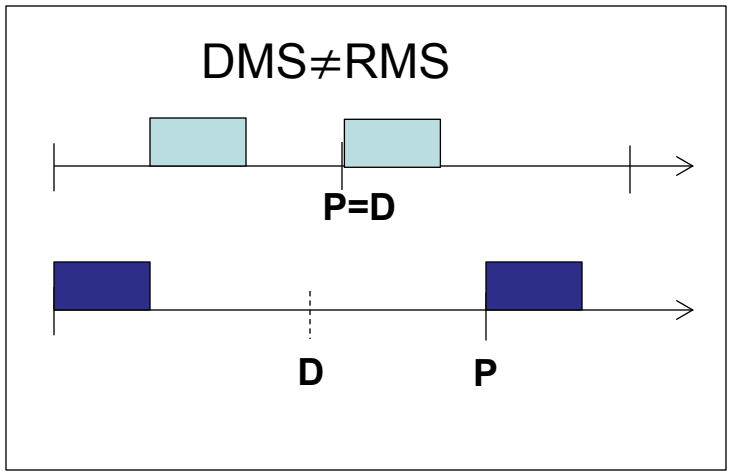
- $An_{therm}.C: I = \{B, BatRows, BatCols, Voltage\}, O = \{K\}, A = \emptyset, G = \emptyset$

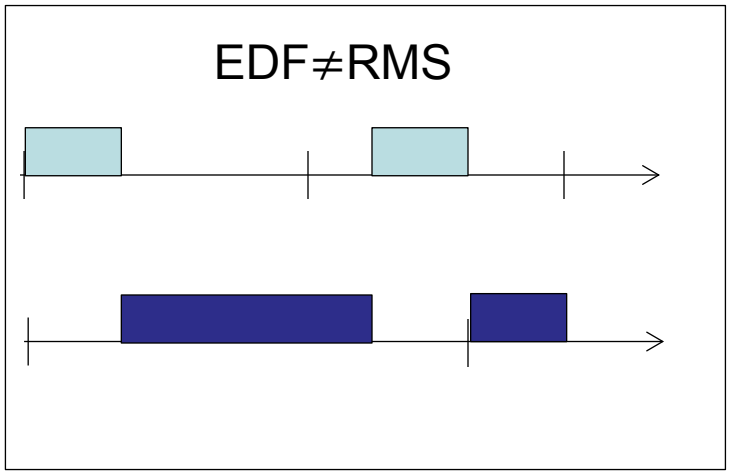
Battery Scheduling

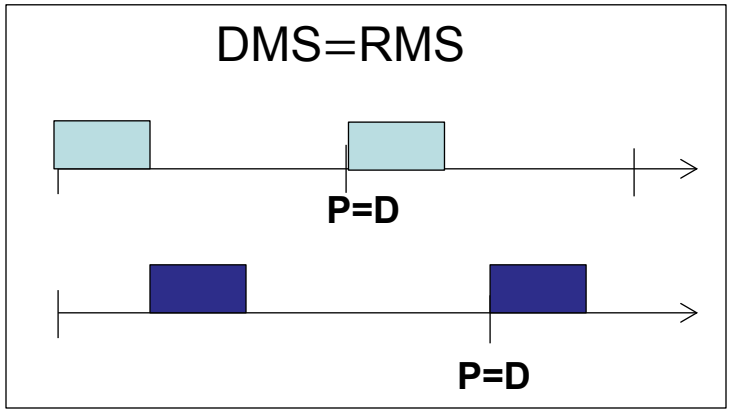
- $An_{bsched}.C: I = \{B, BatRows, BatCols\}, O = \{BatConnSchedPol, HasReqLifetime, SeriqlReq, ParalRea\}, A = \emptyset, G = \{g\}$
- $g: G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0)$

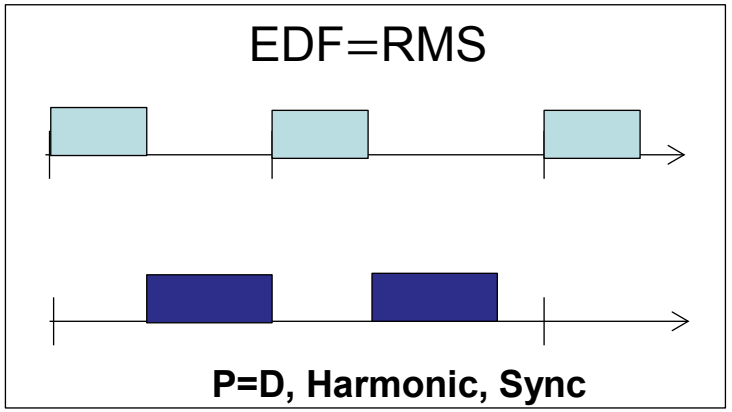
Frequency Scaling Assumption

$a: \forall t_1, t_2 \cdot CPUBind(t_1) = CPUBind(t_2): G(CanPrmpt(t_1, t_2) \Rightarrow Dline(t_1) < Dline(t_2)$









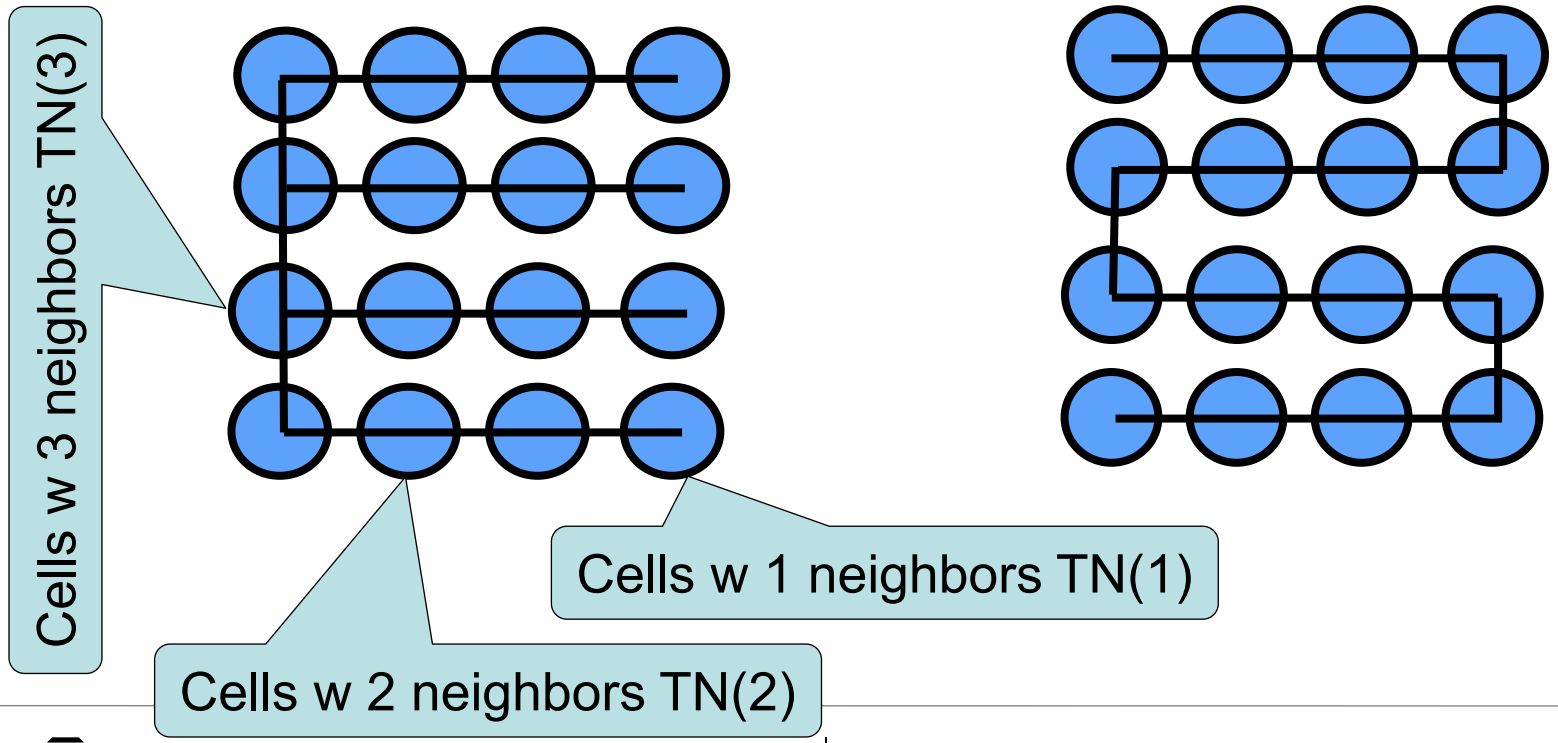
Battery Scheduling Assumption

$$g: G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0)$$

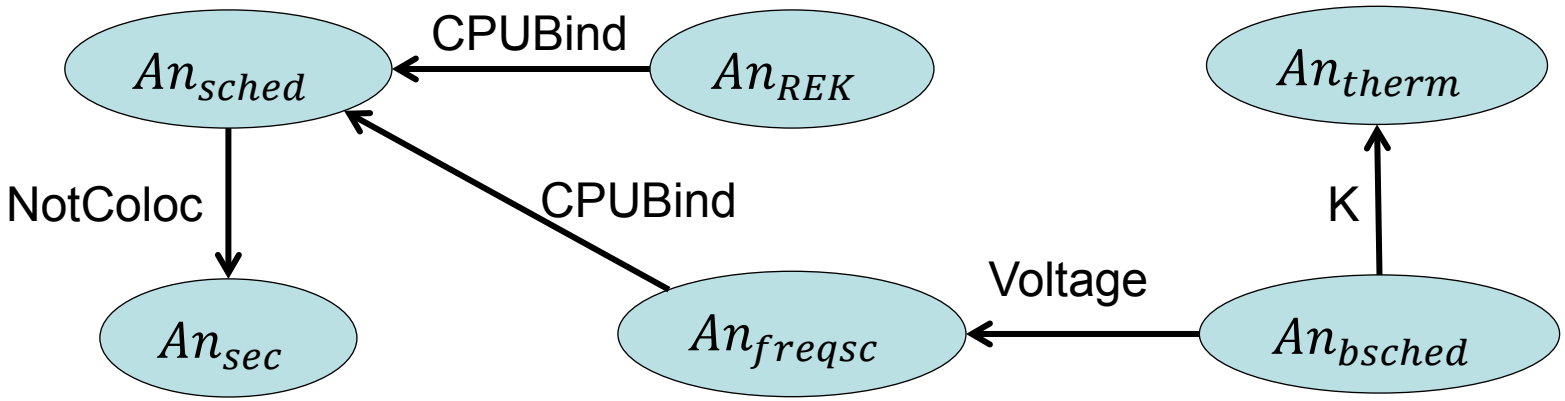
Ratio of cells with 0,1,2,3 neighbors: $1 \cdot TN(1) - 1 \cdot TN(2) + 10 \cdot TN(3) \geq 0$

$1 \cdot 4 - 1 \cdot 10 + 10 \cdot 2 = 14 \geq 0$

$1 \cdot 2 - 1 \cdot 14 + 10 \cdot 0 = -12 < 0$



Analyses Dependencies



Implementation

Models in the Architecture Analysis and Design Language (AADL)

- Supports multiple analysis
- Supports language extensions (subannexes)
- OSATE Implementation

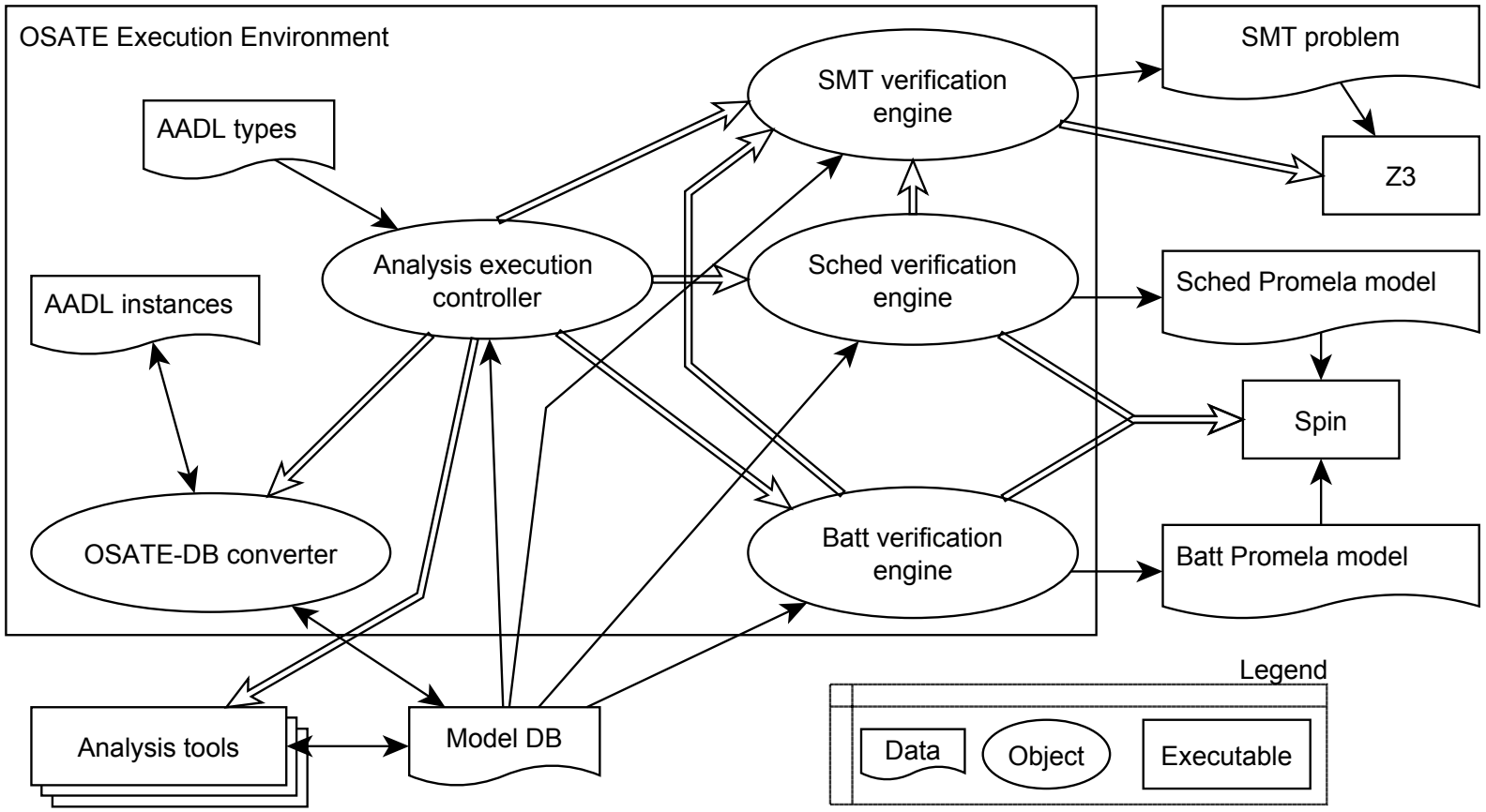
Analysis Contract Annex

- Implement contract language
- Generates model interpretation

Contract formulas verification

- First Order Logic (Static): SMT / Z3
- LTL (Runtime): Model checking / SPIN

Contract Verification Architecture



First-Order Logic Verification (Z3)

```
(define-sort thread () (Int))
(define-sort processor () (Int))
(declare-fun Actual_Processor_Binding (thread) Int)
(define-fun Not_Collocated (x1 thread) (x2 thread)) Bool
  (ite (and (= x1 0) (= x2 1)) true (ite (and (= x1 0)(= x2 2)) true
  (ite (and (= x1 1)(= x2 0)) true (ite (=x1 2)(= x2 0)) true false )))))
(assert (= (Actual_Processor_Binding 0) 0))
(assert (= (Actual_Processor_Binding 1) 1))
(assert (= (Actual_Processor_Binding 2) 1))
(assert (not (forall ((x1 thread) (x2 thread)) (=>
    (and (or (= x1 0) (= x1 1) (= x1 2))
       (or (= x2 0) (= x2 1)(= x2 2))
     (=> (and (not (= x1 x2)) (Not_Collocated x1 x2))
     (not (= (Actual_Processor_Binding x1) (Actual_Processor_Binding x2))))))
(check-sat)
```

Model Checking for Scheduling

Periodic Tasks

Multiple Processors Allows Global Scheduling

Priority Based Scheduling

- Fixed-Task Priorities: fixed at configuration time (RMS / DMS)
- Dynamic-Task (fixed job): changes at job arrival (EDF)

Tickless model

- Time advances by scheduling events
  - Deterministically: next event is a deterministic arrival
  - Non-deterministically: multiple possible events

Clock variable resets

- To the earliest event in variables

Model Checking for Battery Scheduler

Model battery cell scheduler

- Schedules cells to discharge, charge, rest
- Match required output through serial connections (efficient)
- Maximizes battery lifetime

Matrix of battery cells

Connections between cells change dynamically

- Reflects needs to provide output voltage (serial connections)
- at certain current (parallel connections)

Cell charge changes to reflect charge discharge

Evaluation

Real-Time Scheduling

Threads	(R/D)MS Time	EDF Time
3	0.01	0.01
4	0.01	0.52
5	0.07	33.4
6	0.37	2290.0
7	2.18	Out Mem
8	12.4	Out Mem
9	71.2	Out Mem
10	421	Out Mem
11	Out Mem	Out Mem

Battery Scheduling

Cells	FGURR	FGWRR	GPWRR
9	0.13	0.15	0.15
12	0.61	2.34	3.94
16	44	31.4	127
20	1060	619	Out Mem
25	Out Mem	Out Mem	Out Mem

Time in seconds. Amazon EC2 virtual Machine with 8 cores and 30 GB of mem.

Conclusions

Analyses are key for development of CPS

- But inconsistent assumptions may compromise results

Analysis contracts to automatically verify assumptions

- Analysis contract language & verification framework
- Implementation in AADL sub-annex

Example

- Two domains
- Five analyses

Analysis contracts: sound and scalable

- Single multi-domain analysis intractable