FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Progress Towards System-Security Co-design

Janos Sztipanovits
David Lindecker
ISIS-Vanderbilt

Berkeley UNIVERSITY OF CALIFORNIA
Massachusetts Institute of Technology
NSF
MICHIGAN
VANDERBILT UNIVERSITY

Content

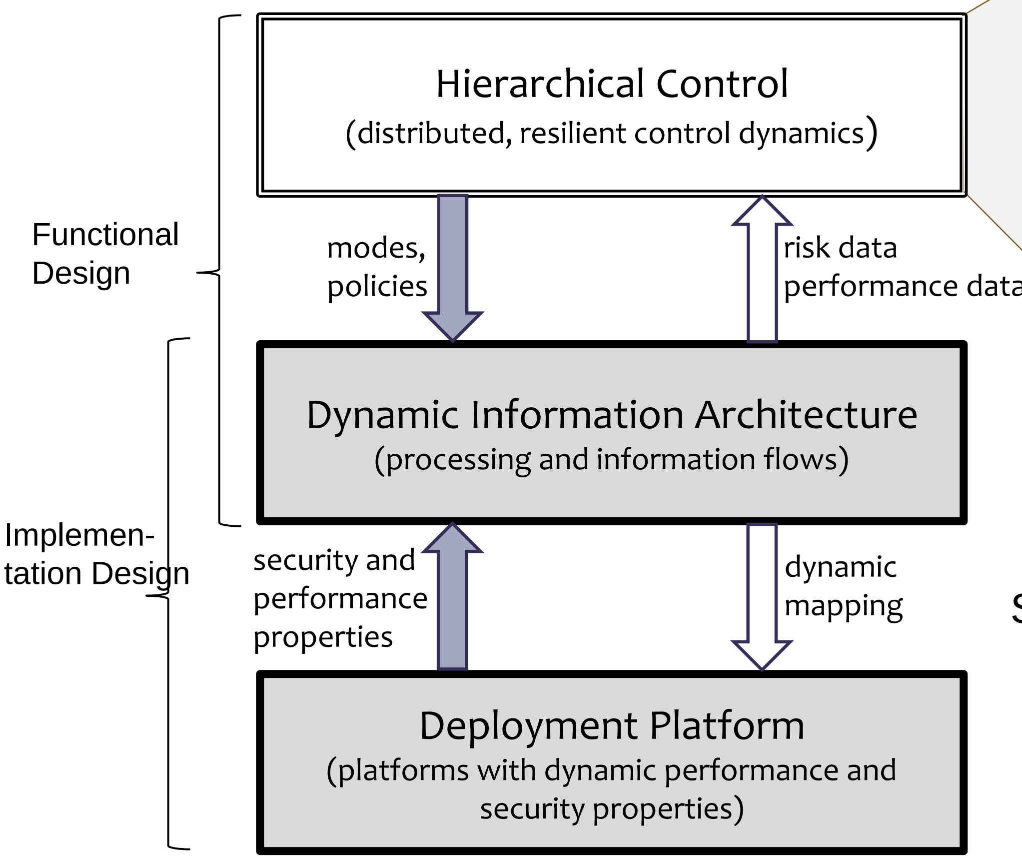
1. **Goals**
2. Decentralized Label Model
3. Formal Framework
4. System-level Synthesis Framework
5. Next Steps

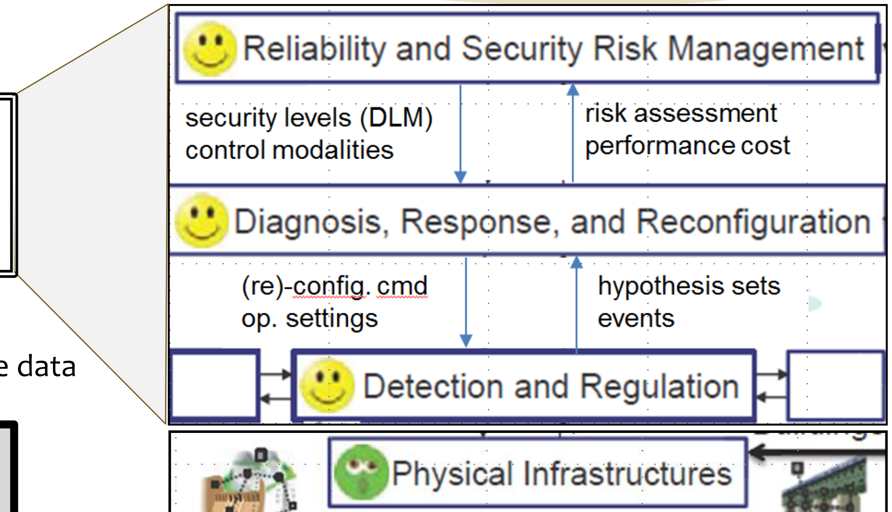
FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

What Is Our Goal?

- **The traditional system-level synthesis problem for the "cyber" side of CPS:**
  - Derive specification for the behavior of the system components that will be implemented using networked computing
  - Derive a functional model for the information architecture and componentize the system
  - Select computing/networking platform
  - Derive deployment model assigning components of the information architecture to processing and communication platforms
  - Generate code for software components and derive WCET and WCCT
  - Perform timing analysis
- **Making security part of system-level co-design**
  - Mitigation of security vulnerabilities cost performance, timing, even functionality
  - Our goal is to address security requirements as part of the design trades embedded in the system-level design process

Co-design Problem in FORCES

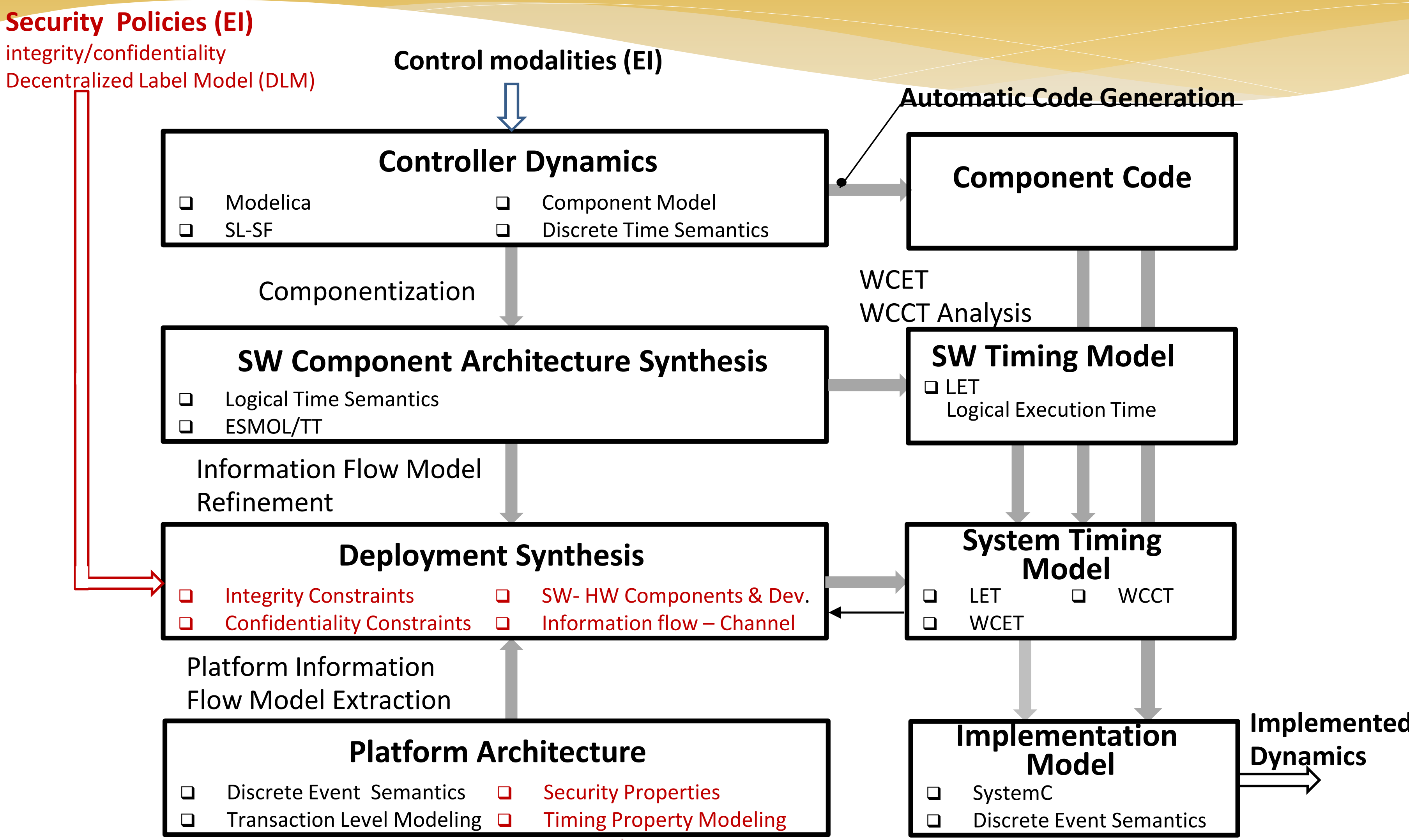




System-Security Codesign

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Design Flow



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Challenges

* Modeling language suite
  (behavior, information flows, SW components, architecture, timing, platform, deployment) - reuse previous work
* Security Requirement Modeling
  (need to be composable with other modeling aspects)
* Common Semantic Domain and Formal Framework
  (functional, performance and security models need to be anchored to a semantic domain suitable for synthesis)
* Synthesis Framework and Co-design flow
  (mapping system-level synthesis problem on the formal framework and tools)
* Integrated Tool Suite and Validation
  (target domain rich enough for testing the co-design tool suite)

Content

1. Goals
2. **Decentralized Label Model**
3. Formal Framework
4. System-level Synthesis Framework
5. Next Steps

Security Concerns Addressed

* **Integrity attacks**
  - Manipulate data (value, timestamp, source identity,..)

* **Confidentiality attack**
  - Leak critical data to unauthorized persons/systems

* **Integrity and confidentiality restrictions impose constraints on information flows.**
  - How to model these restrictions?
  - How to integrate these restrictions with others (functional and timing) and formulate a co-design problem?

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Decentralized Label Model (DLM) for Information Flow Control

* Myers, Liskov (1997): Introduced security-typed languages by labeling variables with information flow security policies
* Method was developed for programming languages, the result is *Jif, a security-typed version of Java.*
* DLM provides mechanism for static/dynamic type checking of security labels in information flows to detect policy violations.
* Example: *Jif*, a security-typed version of Java

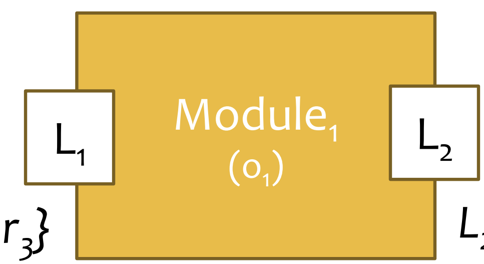
DLM Concepts

- New semantic concepts introduced:
  - *Principles* that represent authority entities.
  - *Labels expressing* security classes encountered in most information flow models.
  - *Policies* that are elementary security primitives used in *labels.*
  - *Labeled entities* that have attached labels, such as *values*, *slots* (*variables, objects, i/o channels*). Copies of *values* can be relabeled, *slots* cannot.
  - *Operators* that can *relabel* or *declassify* values in information flows.
- The model can be naturally applied to system-level information flow modeling languages by assigning security types to input/output ports

Working With Security Labels

- *Labels* contain a set of *policies*. Each *policy* includes an *owner* and a set of *readers* allowed by the *owner*. The *effective reader set* for a *label* is the intersection of every *reader set* in it.
  $L = \{o_1: r_1, r_2; o_2: r_2, r_3\}$
- *Processing blocks* running under the authority of an *owner* can *declassify* the *owner's policy* by adding *readers.*

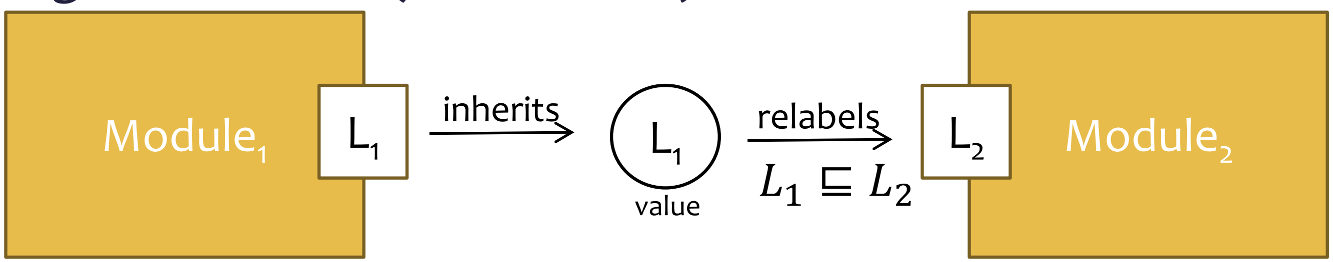
$L_1 = \{o_1: r_1, r_2; o_2: r_2, r_3\}$



$L_2 = \{o_1: r_1, r_2, r_3, ; o_2: r_2, r_3\}$

Propagation Rules

* *Propagation rule-1 (restriction):*

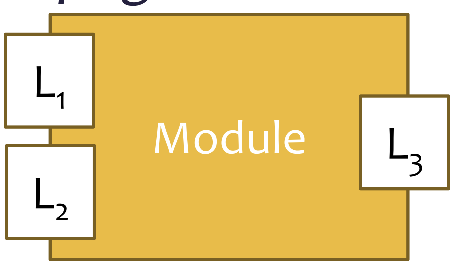


$owners(L_1) \subseteq owners(L_2)$

$\forall o \in owners(L_1), readers(L_1, o) \supseteq readers(L_2, o)$

(L_1 has more readers and fewer owners than L_2)

* *Propagation rule-2 (join):*



L_3 is the *join* of L_1 and L_2

$L_3 = L_1 \sqcup L_2$

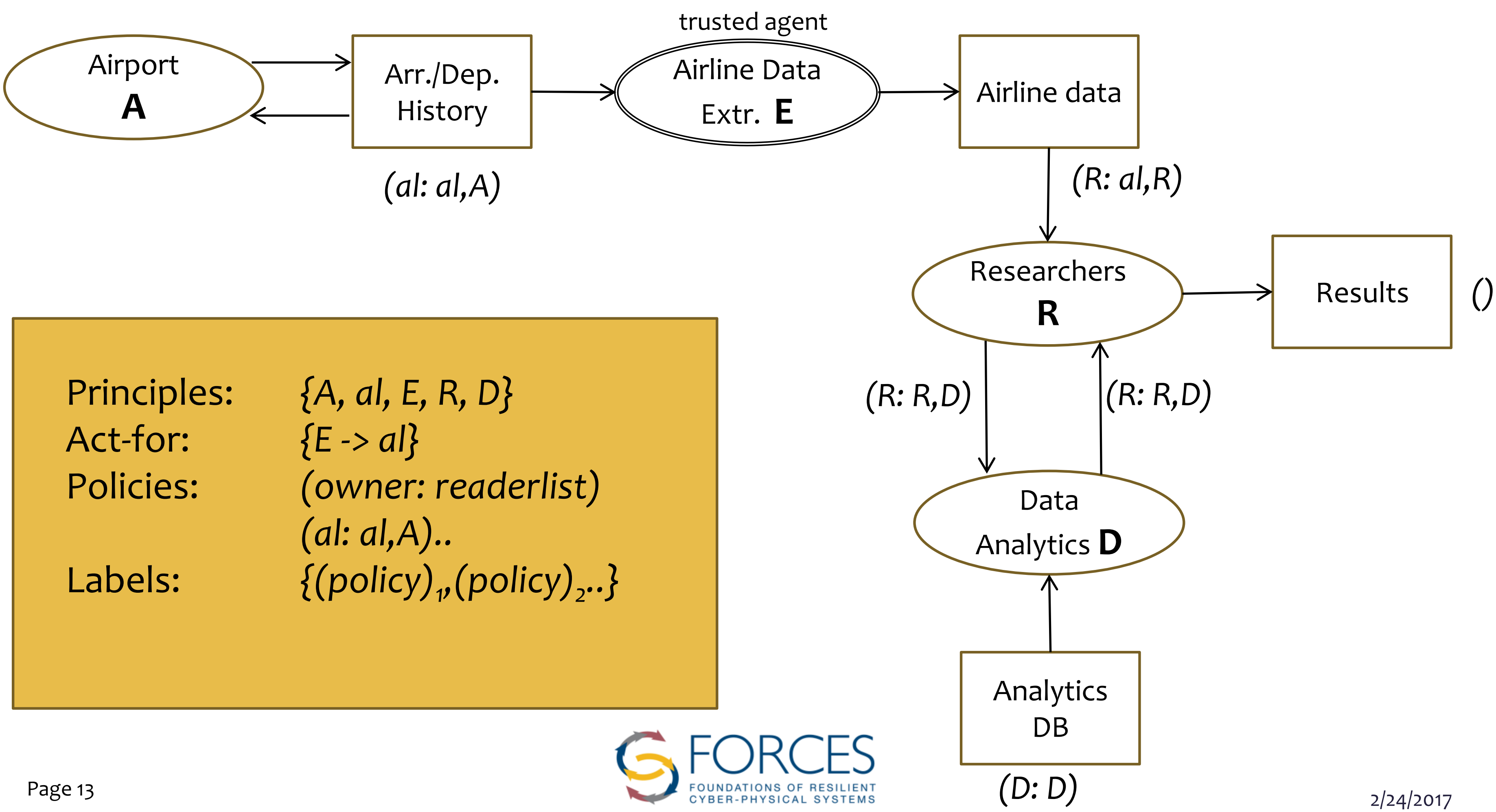
$owners(L_1 \sqcup L_2) = owners(L_1) \cup owners(L_2)$

$readers(L_1 \sqcup L_2, o) = readers(L_1, o) \cap readers(L_2, o)$

(*join* L_1 and L_2 is the least restrictive label that maintains all the flow restrictions specified by L_1 and L_2)

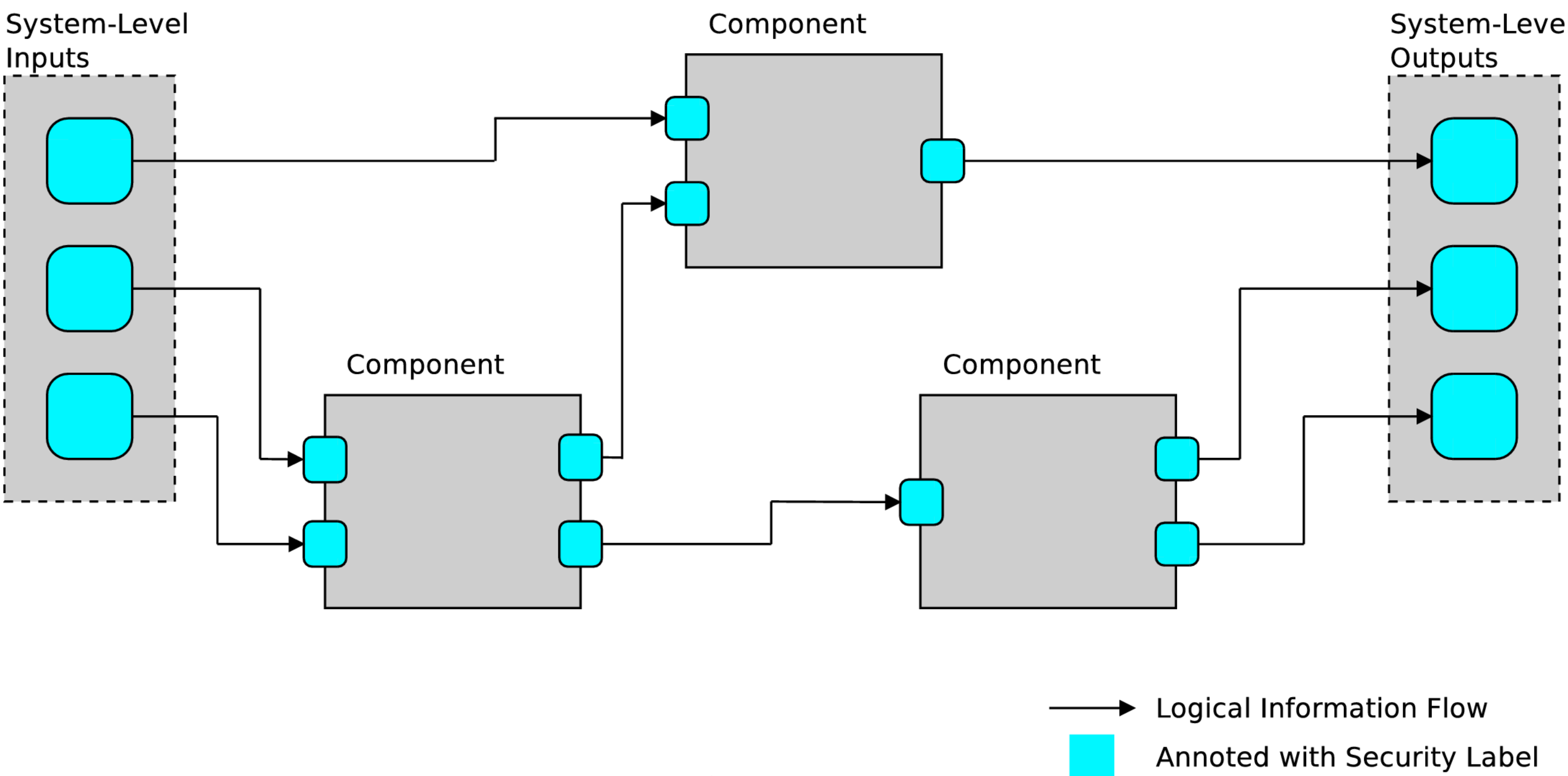
FORCES FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Simple Example



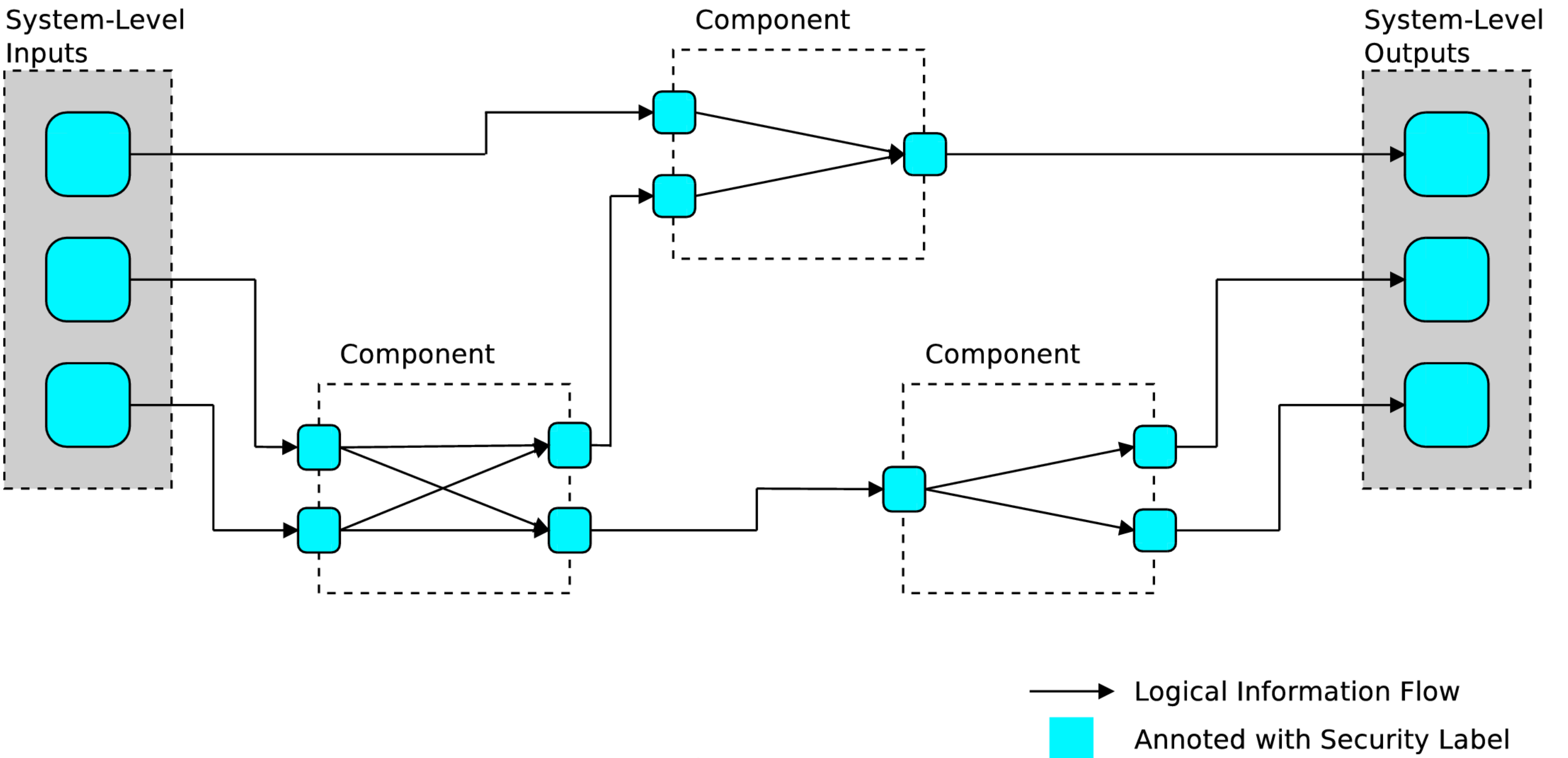
Principles:	*{A, al, E, R, D}*
Act-for:	*{E -> al}*
Policies:	*(owner: readerlist)*
	(al: al,A)..
Labels:	$\{(policy)_1,(policy)_2..\}$

Information Flow Over SW Component Model

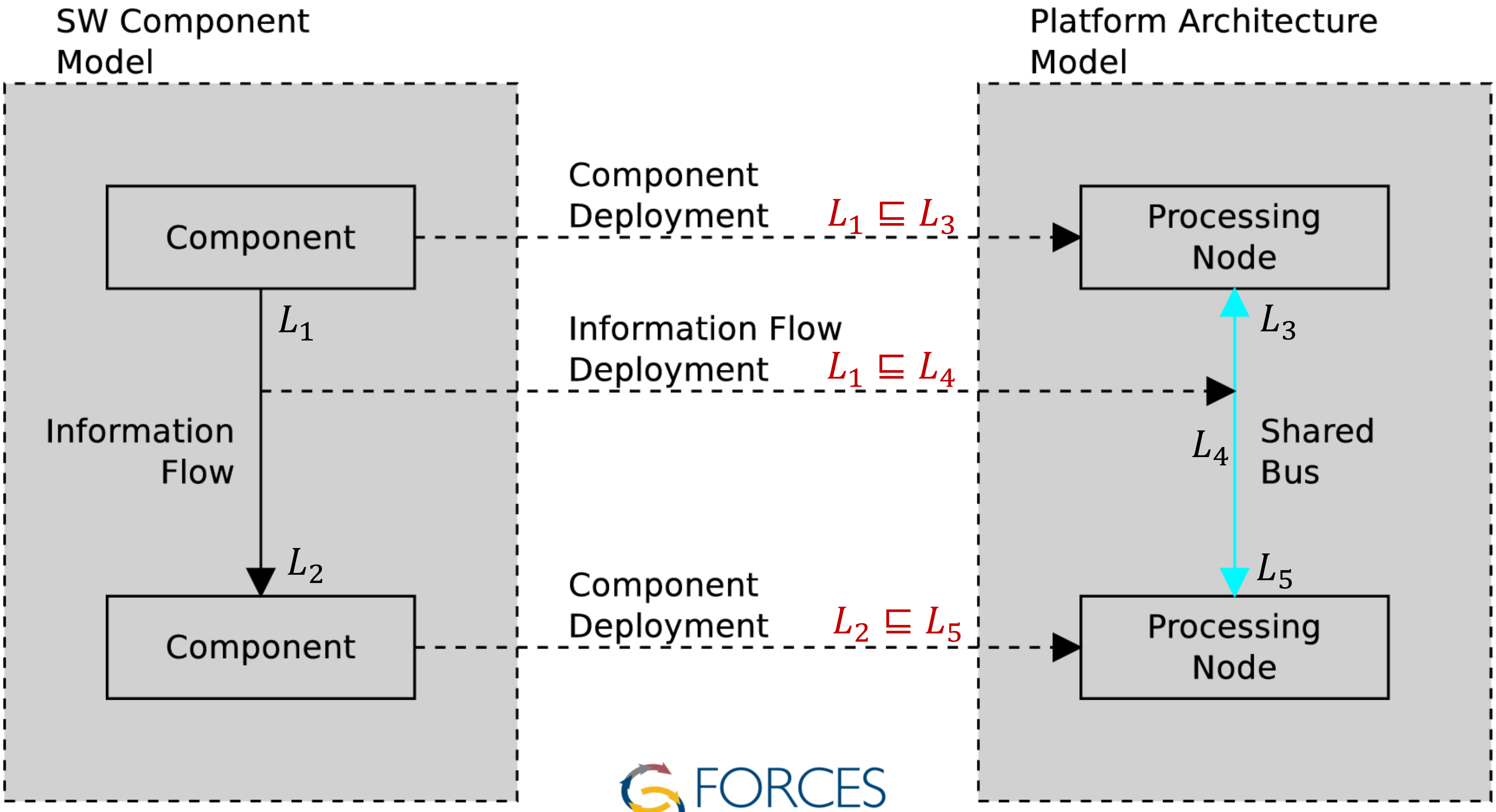


FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Information Flow Over SW Component Model



Information Flow Over System Platform -s



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Content

1. Goals
2. Decentralized Label Model
3. **Formal Framework**
4. System-level Synthesis Framework
5. Next Steps

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

FORMULA

* Ethan Jackson (ISIS grad student 2004-2008; MSR 2009 – Present)
* Algebraic Data Types (ADT) Open World Logic Programs (OLP) provide common semantic domain for DSMLs and model transformations.
* Constraint Logic Programming provides execution semantics for model transformations.
* Z3 backend for model finding.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Example: Deployments Domain

```
domain Deployments
{
  Service ::= new (name: String).
  Node ::= new (id: Natural).
  Conflict ::= new (s1: Service, s2: Service).
  Deploy ::= fun (s: Service => n: Node).

  conforms no { n | Deploy(s, n), Deploy(s', n),
         Conflict(s, s') }.
}
```

Example: Partial Model

```
partial model SpecificProblem of Deployments
{
    requires Deployments.conforms.

    sVoice is Service("Voice Recognition").
    sDB is Service("Big Database").
    n0 is Node(0).
    n1 is Node(1).
   Conflict(sVoice, sDB).
}
```

Content

1. Goals
2. Decentralized Label Model
3. Formal Framework
4. **System-level Synthesis Framework**
5. Next Steps

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Principal Hierarchy

Stakeholders are denoted by principals, each uniquely identified by a name:

```
Principal ::= new (name:String).
```

A relation over principals:

```
ActsFor   ::= new (Principal, Principal).
```

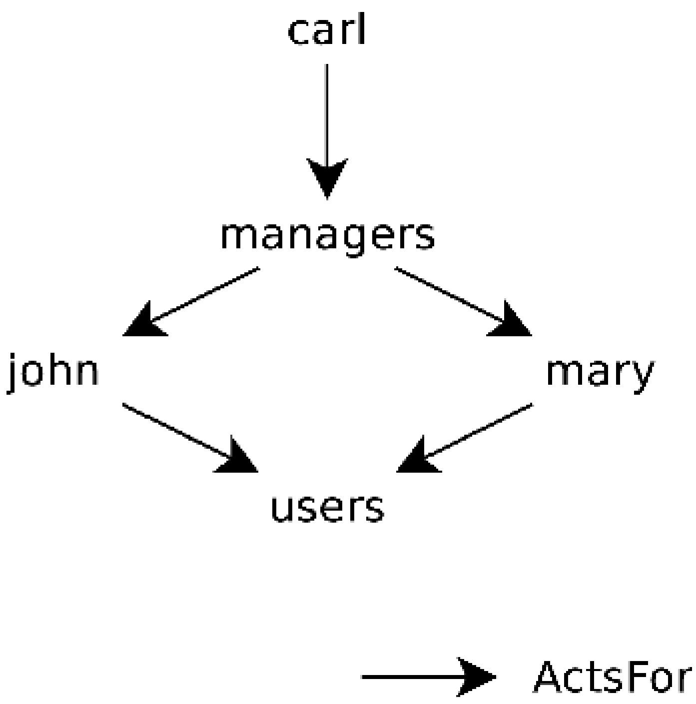
The term `ActsFor(A,B)` indicates that principal A is allowed to perform actions as if it were principal B.

The "ActsFor" relation is transitive and reflexive:

```
ActsForTR ::= (Principal, Principal).
ActsForTR(x,x) :- x is Principal. //reflexivity
ActsForTR(x,y) :- ActsFor(x,y).
ActsForTR(x,z) :- ActsForTR(x,y), ActsFor(y,z). //transitivity
```

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Principal Hierarchy Example




```
carl is Principal("carl").
managers is Principal("managers").
john is Principal("john").
mary is Principal("mary").
users is Principal("users").
ActsFor(carl, managers).
ActsFor(managers, john).
ActsFor(managers, mary).
ActsFor(john, users).
ActsFor(mary, users).
```

Policies and Labels

A policy consists of an owner principal and a set of allowed reader principals:

owner: reader1 reader2

A label is a (possibly empty) set of policies:

L = { policy1; policy2; ...}

Our encoding views a label as a tree where the label's identifier is the root, the policy owners make up the second level, and the corresponding readers make up the third level :

```
Label  ::= new (name:String).
Policy ::= new (lbl:Label, owner:Principal).
Reader ::= new (pl:Policy, reader:Principal).
```


FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Label Encoding Example

L1 = { sam: bob amy; john: bob }

```
L1 is Label(“L1”).
P1 is Policy(L1, Principal(“sam”)).
Reader(P1, Principal(“bob”)).
Reader(P1, Principal(“amy”)).
P2 is Policy(L1, Principal(“john”)).
Reader(P2, Principal(“bob”)).
```

Inferring Label Information

We can compute the effective readers set for each label:

```
EffReader(lbl, reader) :-
  lbl is Label, reader is Principal, no CantRead(lbl, reader).
CantRead(pl.lbl, r) :-
  pl is Policy, r is Principal,
  no { r' | ActsForTR(r, r'), Reader(pl, r') }.
```

We can compare the restrictiveness of labels based on their effective reader sets:

```
AtLeastAsRestrictive(lbl1, lbl2) :-
  lbl1 is Label, lbl2 is Label,
  no { x | EffReader(lbl1, x), CantRead(lbl2, x) }.
```

We can also “propagate” policies by computing the join (⨆) of two labels: the least restrictive label that is at least as restrictive as both labels.

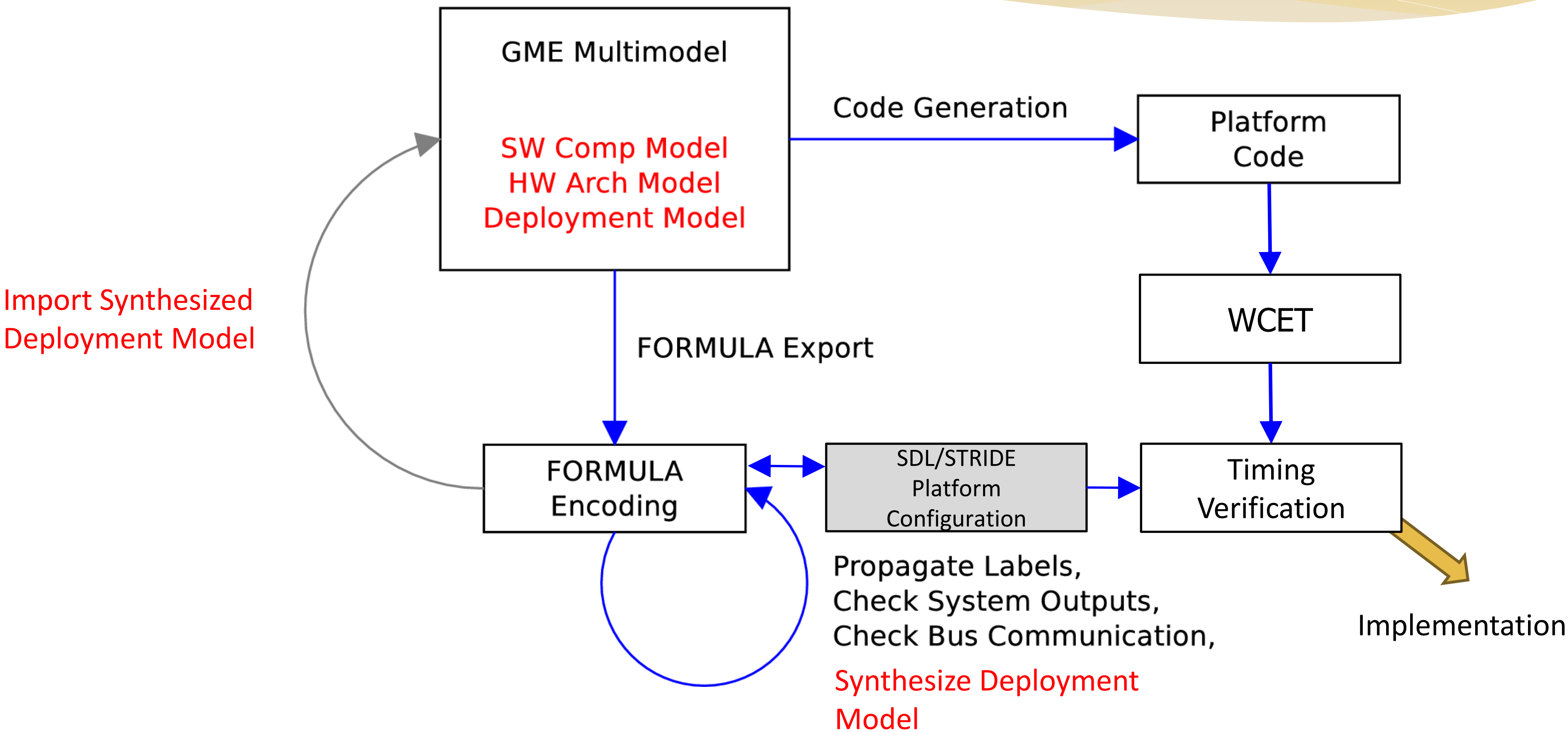
FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Content

1. Goals
2. Decentralized Label Model
3. Formal Framework
4. System-level Synthesis Framework
5. **Next Steps**

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

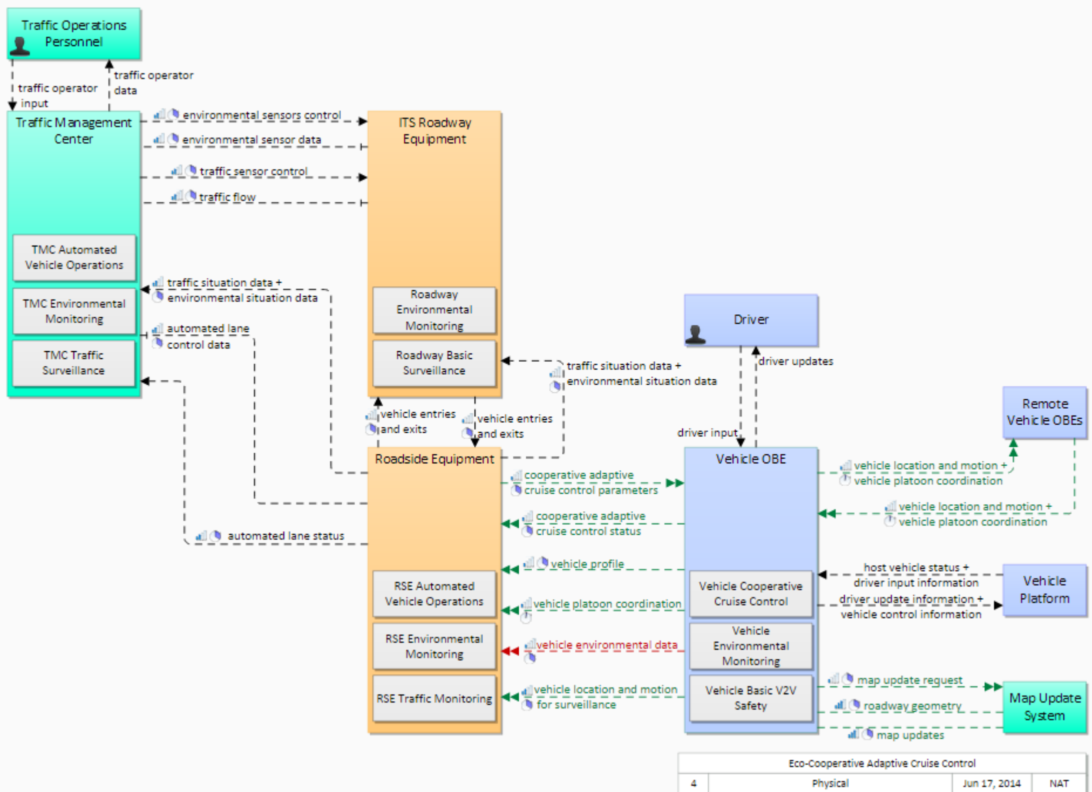
Workflow for Designing Secure Distributed Embedded Systems



Test Cases – DOT/CVRIA

Physical

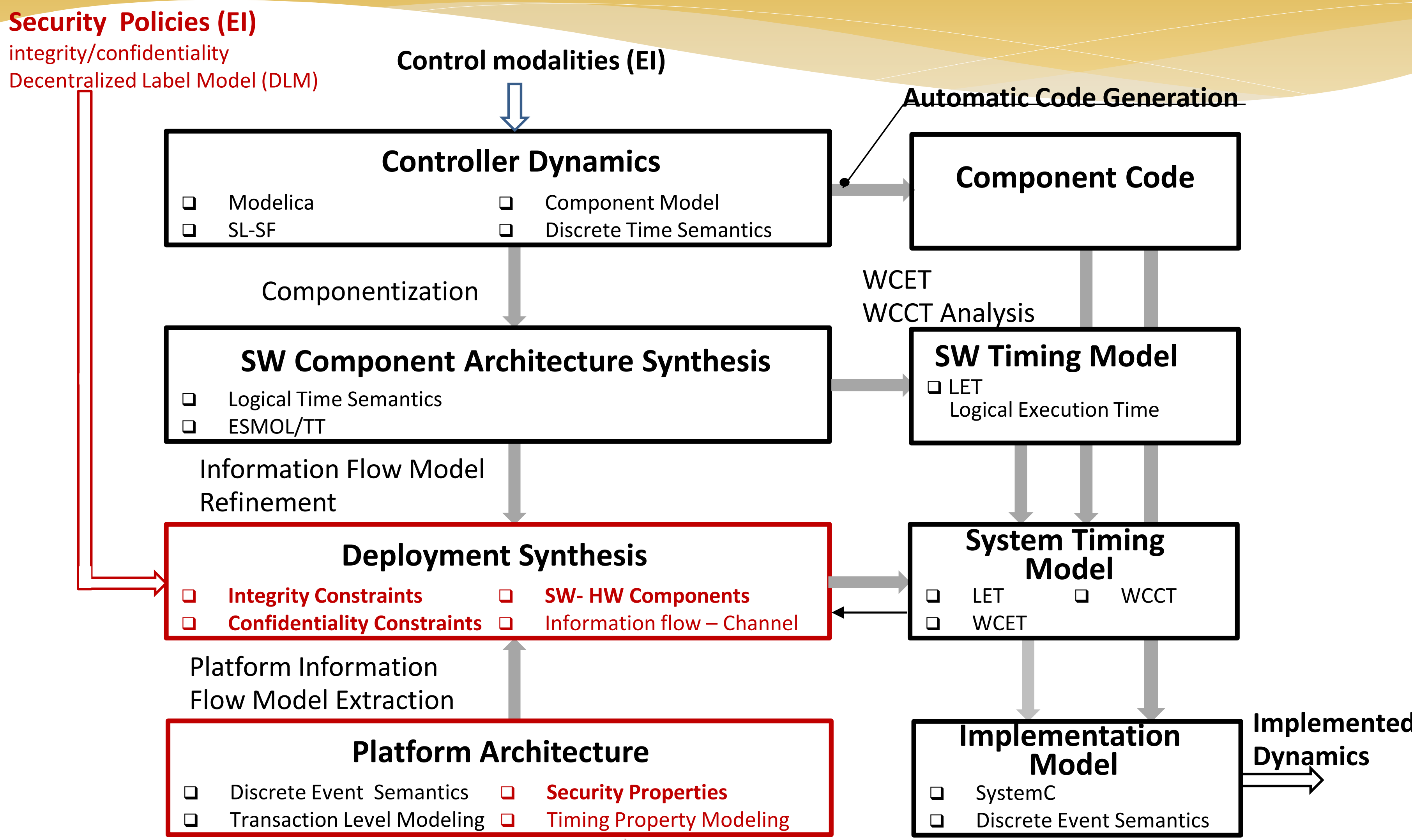
The physical diagram can be viewed in SVG or PNG format and the current format is SVG.
SVG Diagram
PNG Diagram



- Modeling Language Capture in GME
- Semantics in FORMULA
- DLM mapping
- Analysis Studies

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Summary



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS