Formal Methods for IT Security

ISACA – CETIC Meeting
May 23th 2007

Objectives of the talk

What are formal methods ?

What to expect from their application ?

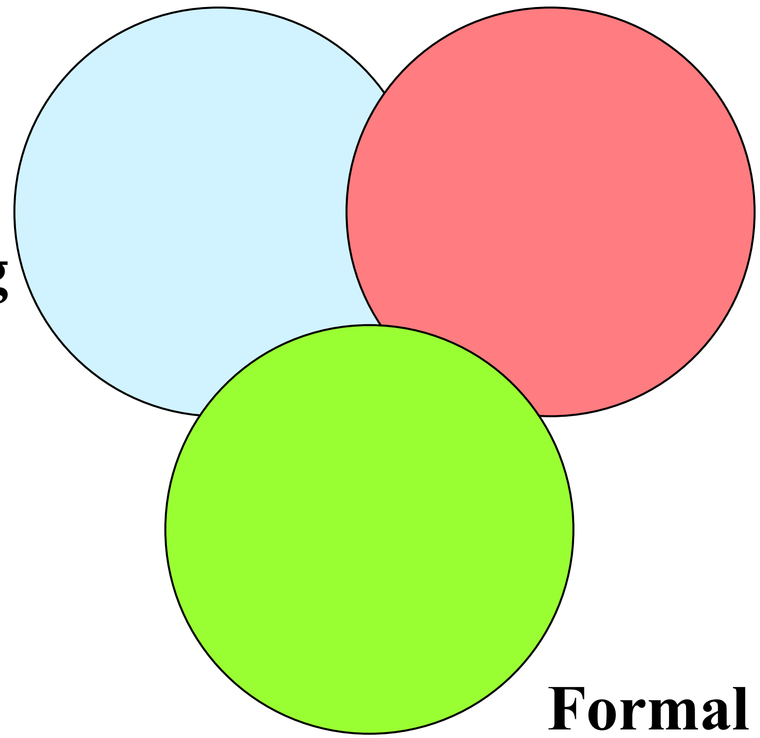
Our experience with some formal modeling tools

Context : IT Security

The roadmap

- **Introduction**
- The added value brought by Formal methods
- Formal models in C.C. certification
- Formal modeling tools
- Cetic experience with formal tools
- Conclusion

Formal Methods & IT Security



What are formal methods ?

Formal methods are mathematically-based techniques for the specification, development and verification of software and hardware systems

Idea : performing mathematical analyses can contribute to the reliability and robustness of a design.

IT Security

IT / Computer Security aims at preventing, or at least detecting unauthorized actions by agents in a computer system.

IT security complements :
Safety : absence of damage due to mistakes or other unintentional failure

IT Security as a Software Engineering Problem

Situation : security loopholes in IT systems actively exploited
Objective : thwart attacks by absence of vulnerabilities
Difficulty : security is interwoven with the whole system.
IT systems are very complex, security flaws are hard to find.

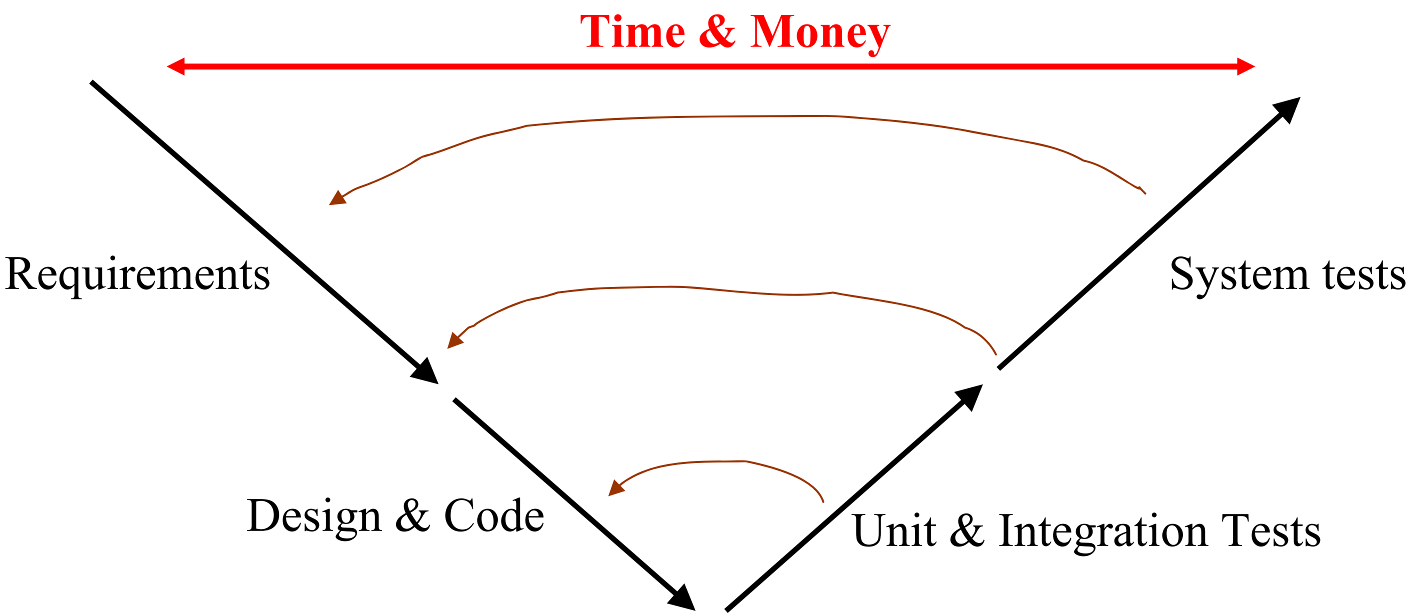
Remedy :

- **address security in all development phases**
- **do review and tests**
- **make use of formal modeling / analysis**

The roadmap

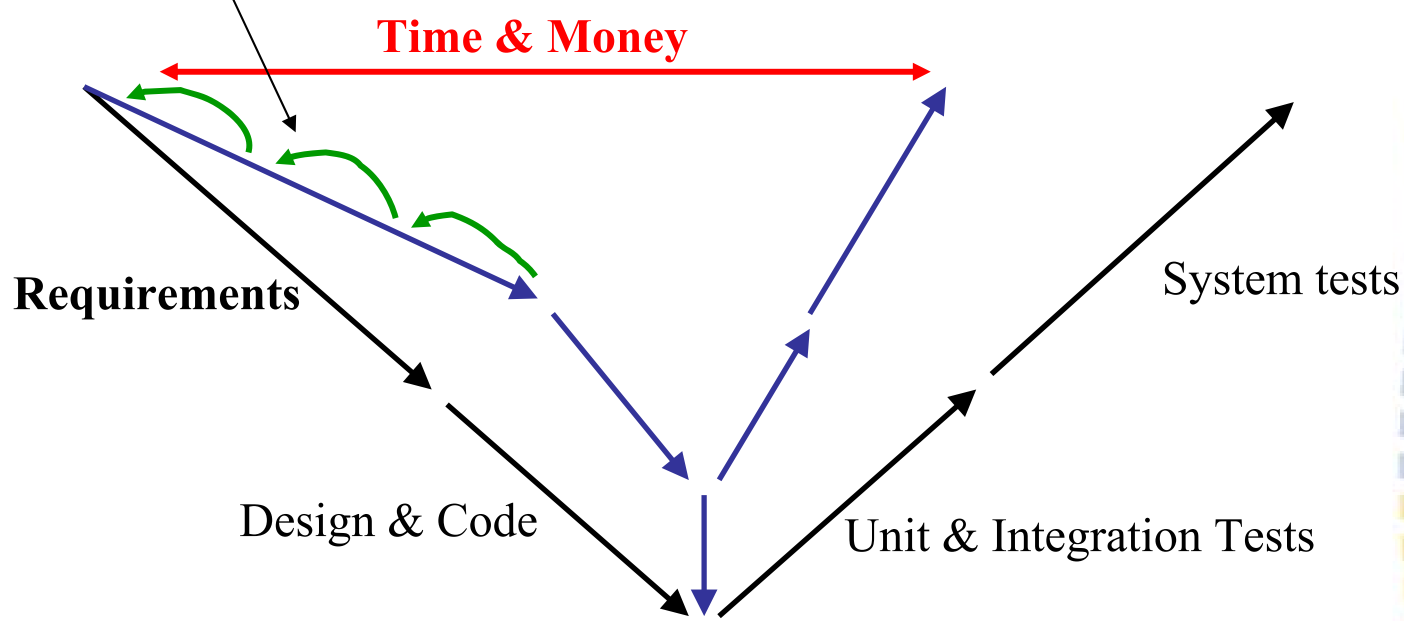
- Introduction
- **The added value brought by Formal metho**
- Formal models in C.C. certification
- Formal modeling tools
- Cetic experience with formal tools
- Conclusion

V Life Cycle

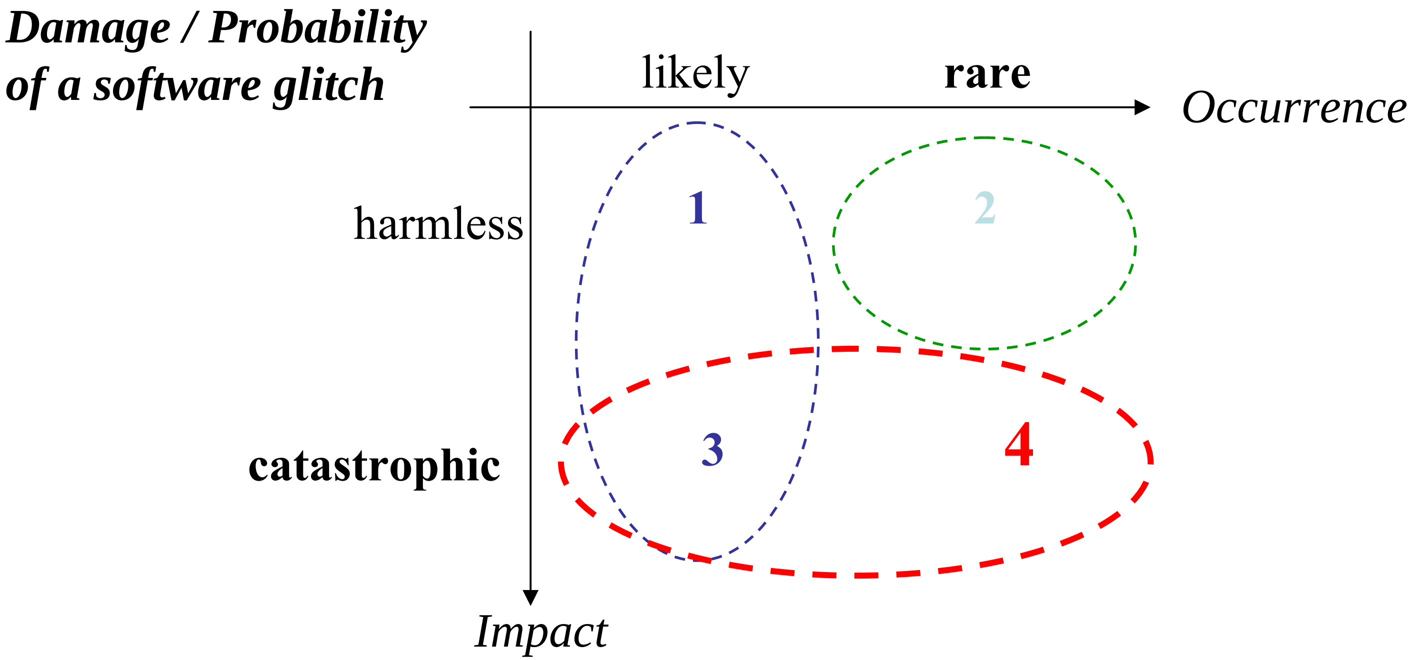

Time & Money
Requirements
System tests
Design & Code
Unit & Integration Tests

V Life Cycle with Formal Methods

get at formal artifacts early enough in the lifecycle to apply useful analysis within the design loop.



Added Value of Formal Methods



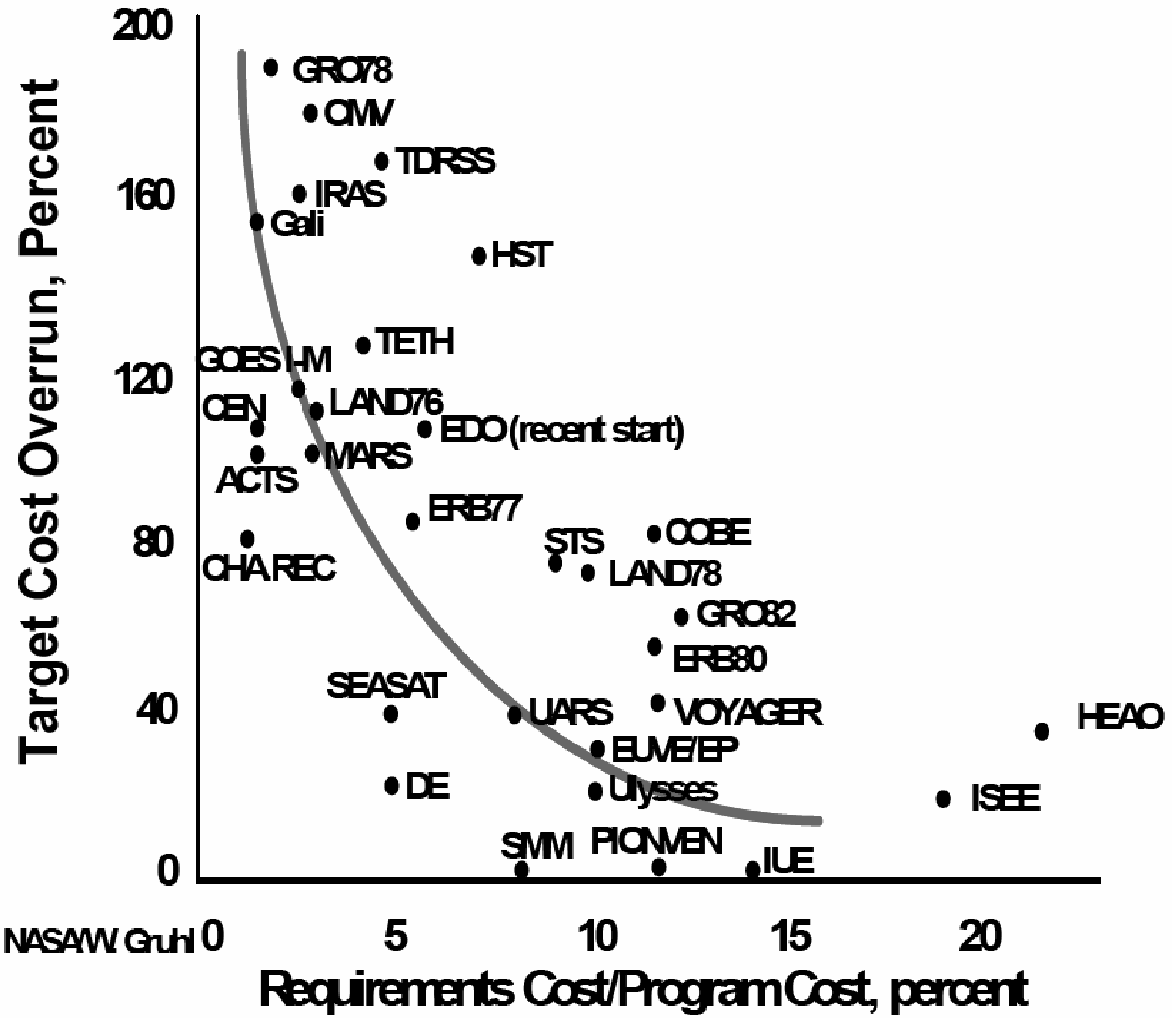
1 & 3 : covered by traditional testing

2 : not very important

3 & 4 : target of formal methods techniques

4 : is important and not adequately covered by traditional sw testing

NASA Feedback :
Invest in your requirements !



Ref: NASA/W. Gruhl

The roadmap

- Introduction
- The added value brought by Formal methods
- **Formal models in C.C. certification**
- Formal modeling tools
- Cetic experience with formal tools
- Conclusion

Certification Goals & General Approach

Goal : gain **confidence** in the security of a system

- What are the goals to be achieved ?
- Are the measures employed appropriate to achieve the goal ?
- Are the measures implemented correctly ?

Approach : **assessment** of system security **by neutral experts**

- **Understanding** the security functionality of the system
- Gaining evidence that functionality is correctly implemented
- Gaining evidence that the integrity of the system is kept

Result : Successful evaluation is awarded a **certificate**

Common Criteria

International standard :

- Version 2.1 : ISO / IEC 15408:1999
- Version 3.1 : ISO / IEC 15408:2006

Generic approach :

- full range of IT systems
- scalable level of assurance

CC process : Build a Security Target

- Definition of the Target of Evaluation (TOE) and separation from its environment
- Definition of the security threats and objectives for the TOE
- Introduction of TOE Security Functions (TSF) : measures intended to counter the threats
- Determination of Evaluation Assurance Level (EAL)

⇒ The **Security Target** is the central document to which all subsequent evaluation activities and results refer !

Evaluation Assurance levels

EAL1 : functionally tested
EAL2 : structurally tested
EAL3 : methodically tested and checked
EAL4 : methodically designed, tested and reviewed

EAL5 : semiformally designed and methodically tested including **formal security policy model**
EAL6 : semiformally verified and methodically tested
EAL7 : formally verified design and methodically tested

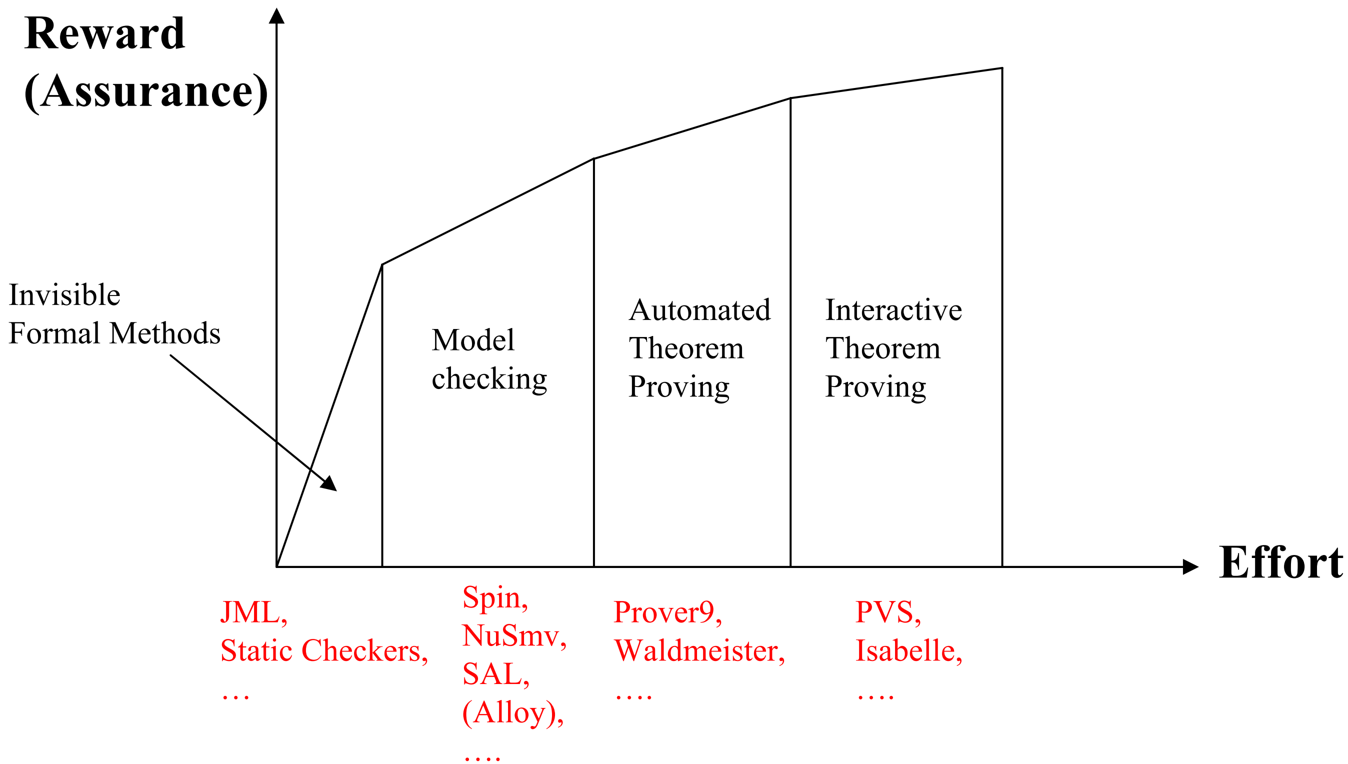
Increasing requirements on scope, depth and rigor

The roadmap

- Introduction
- The added value brought by Formal methods
- Formal models in C.C. certification
- **Formal tools & examples**
- Cetic experience with formal modeling tools
- Conclusion

Formal tools

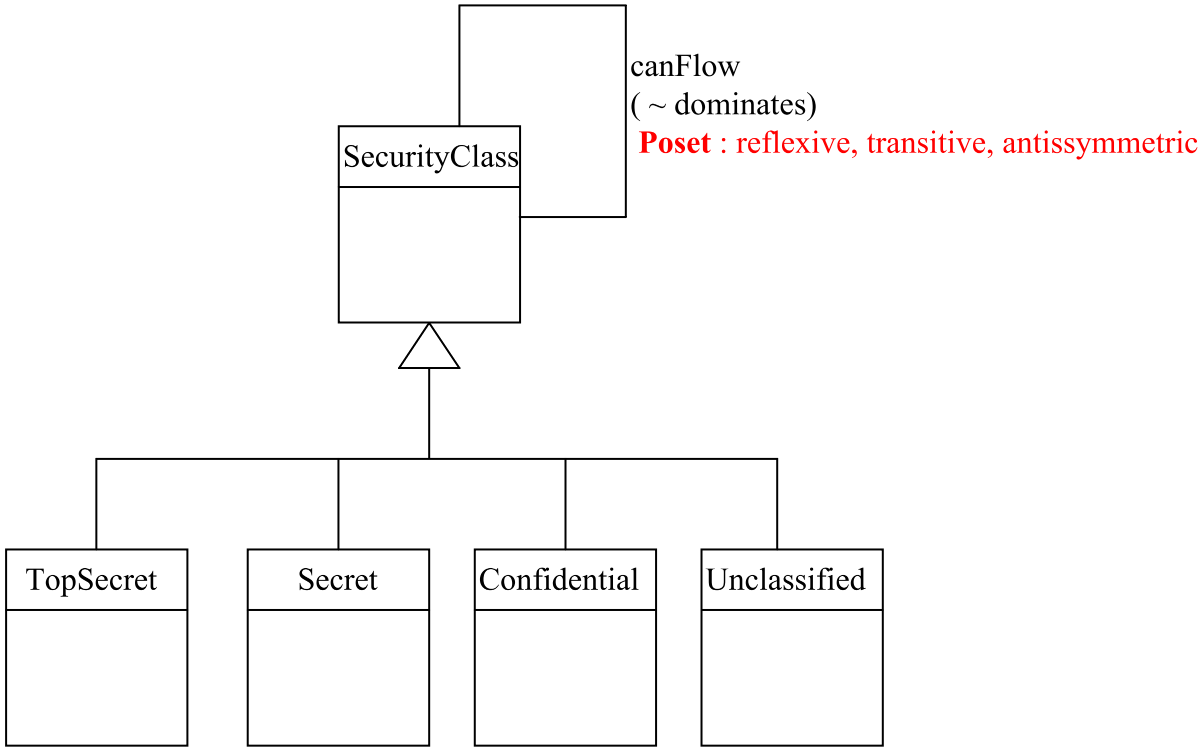
Which formal tools ?

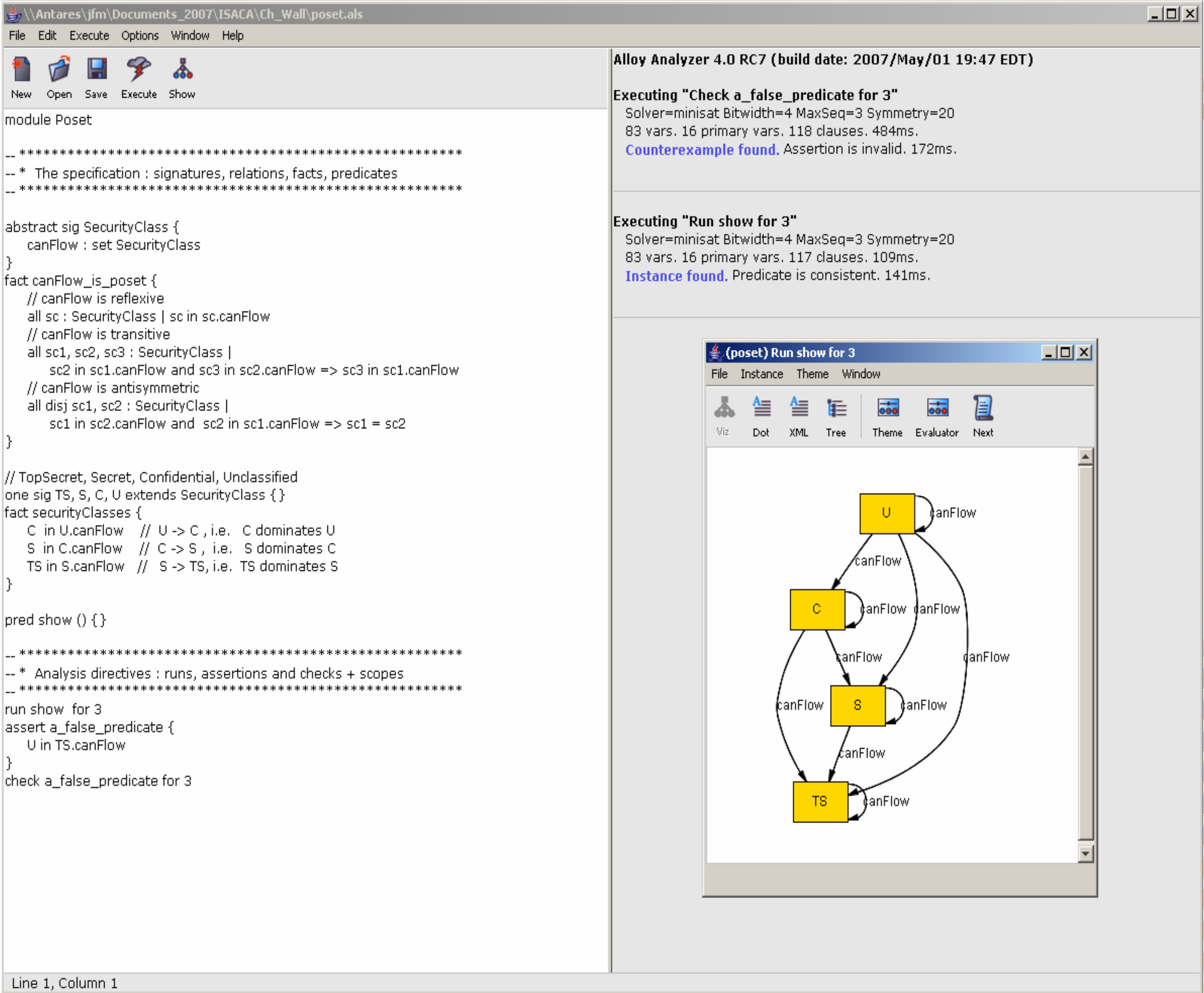


Alloy Analyzer

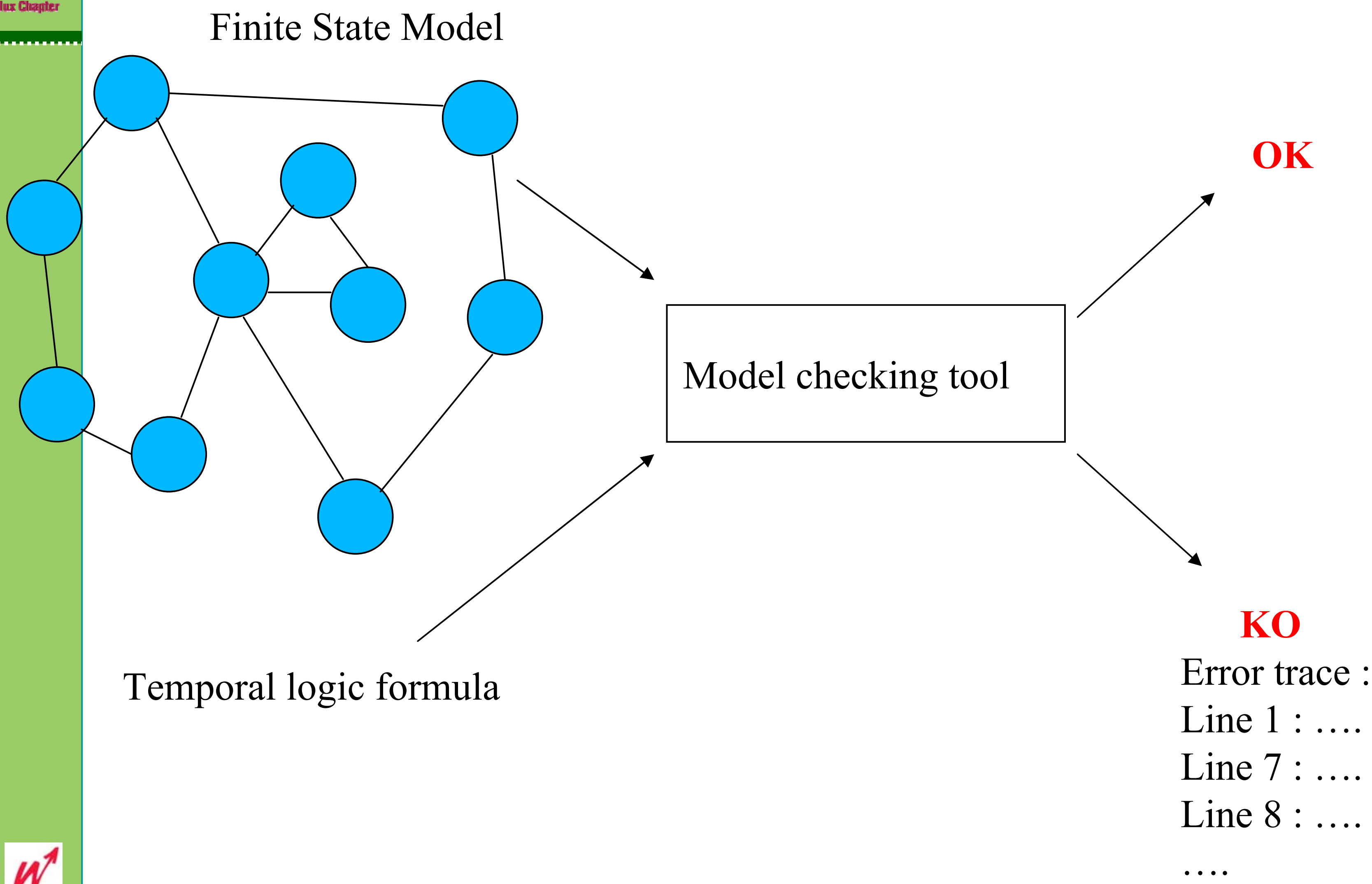
- A model finder
- Based on SAT technology : given a propositional formula, finds an assignment of the propositional variables that satisfies the formula
- Input : a first-order relational logic specification, analysis directives with scopes
- Automatic push-button technology, no expert knowledge required
- Output : examples and counterexamples

Alloy Language




\\Antares\jfm\Documents_2007\ISACA\Ch_Wall\poset.als
File Edit Execute Options Window Help
New Open Save Execute Show
module Poset
-- ***
-- * The specification : signatures, relations, facts, predicates
-- ***
abstract sig SecurityClass {
canFlow : set SecurityClass
}
fact canFlow_is_poset {
// canFlow is reflexive
all sc : SecurityClass | sc in sc.canFlow
// canFlow is transitive
all sc1, sc2, sc3 : SecurityClass |
sc2 in sc1.canFlow and sc3 in sc2.canFlow => sc3 in sc1.canFlow
// canFlow is antisymmetric
all disj sc1, sc2 : SecurityClass |
sc1 in sc2.canFlow and sc2 in sc1.canFlow => sc1 = sc2
}
// TopSecret, Secret, Confidential, Unclassified
one sig TS, S, C, U extends SecurityClass { }
fact securityClasses {
C in U.canFlow // U -> C , i.e. C dominates U
S in C.canFlow // C -> S , i.e. S dominates C
TS in S.canFlow // S -> TS, i.e. TS dominates S
}
pred show () { }
-- ***
-- * Analysis directives : runs, assertions and checks + scopes
-- ***
run show for 3
assert a_false_predicate {
U in TS.canFlow
}
check a_false_predicate for 3
Line 1, Column 1
Alloy Analyzer 4.0 RC7 (build date: 2007/May/01 19:47 EDT)
Executing "Check a_false_predicate for 3"
Solver=minisat Bitwidth=4 MaxSeq=3 Symmetry=20
83 vars. 16 primary vars. 118 clauses. 484ms.
Counterexample found. Assertion is invalid. 172ms.
Executing "Run show for 3"
Solver=minisat Bitwidth=4 MaxSeq=3 Symmetry=20
83 vars. 16 primary vars. 117 clauses. 109ms.
Instance found. Predicate is consistent. 141ms.
(poset) Run show for 3
File Instance Theme Window
Viz Dot XML Tree Theme Evaluator Next
U
C
S
TS
canFlow

Model Checking



Model Checking

- Calculates whether a system satisfies a certain behavioural property :
  - is the system deadlock free ?
  - whenever a packet is sent, will it eventually be received ?
- Is it like testing ? No, the major difference is : Looks at **all** possible behaviors of a system
- Automatic push-button technology, no expert knowledge required
- Output : examples and counterexamples that help in understanding, communicating and that can be animated

Model Checking

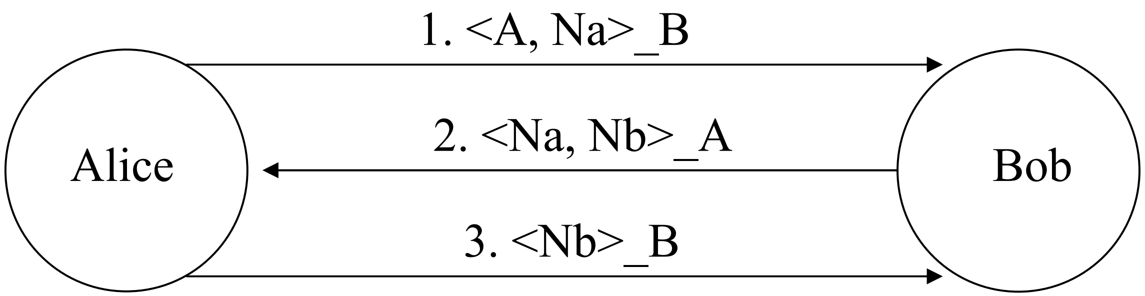
- How do we express the finite state model ?
- How do we express the behavioural property?

Example : The Needham Schroeder protocol with the SAL model checking tool (SRI).

Needham –Schroeder protocol (circa 1978)

Protocol’s purpose :

mutual authentication between principals A and B
in the presence of an intruder who can intercept, delay, read, copy, and generate messages but who does not know the secret keys of the principals.



Needham Schroeder Protocol with SAL

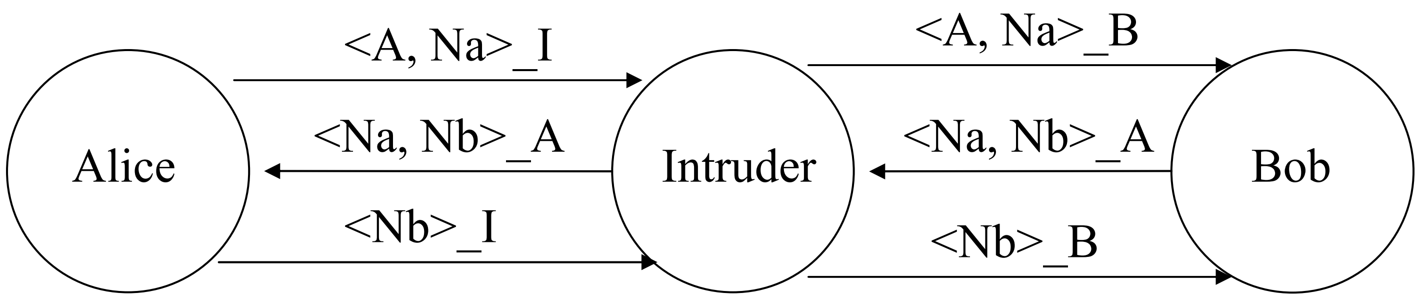
```
Network {msg : TYPE;}:CONTEXT =
BEGIN
... network : MODULE =
      ...INITIALIZATION ...TRANSITION...
END

Needhamschroeder : CONTEXT =
BEGIN
... net: CONTEXT = network{msg;} ...
  principal[i: principals] : MODULE
      ...INITIALIZATION ...TRANSITION...
  intruder[x: intruders] : MODULE
      ...INITIALIZATION ...TRANSITION...

System : MODULE = (([] (id: principals): principal[id]) [] intruder[e])
            || (RENAME buffer TO imsg, inms TO omsg IN net!network);

Prop: THEOREM system |- G ((FORALL (x, y: principals) :
            (pc[x]=responding AND responder[x]=y) =>
                  ((pc[y]=waiting OR pc[y]=engaged) AND responder[y]=x)));
END
```

17 years later ... : man in the middle attack

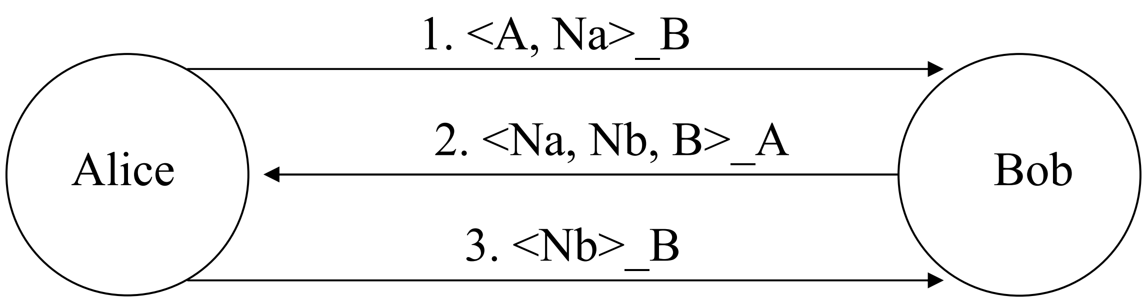


At the end of the attack, Bob falsely believes
that Alice is communicating with him,
and that NA and NB are known only to Alice and Bob.

Needham –Schroeder protocol

Protocol’s fix :

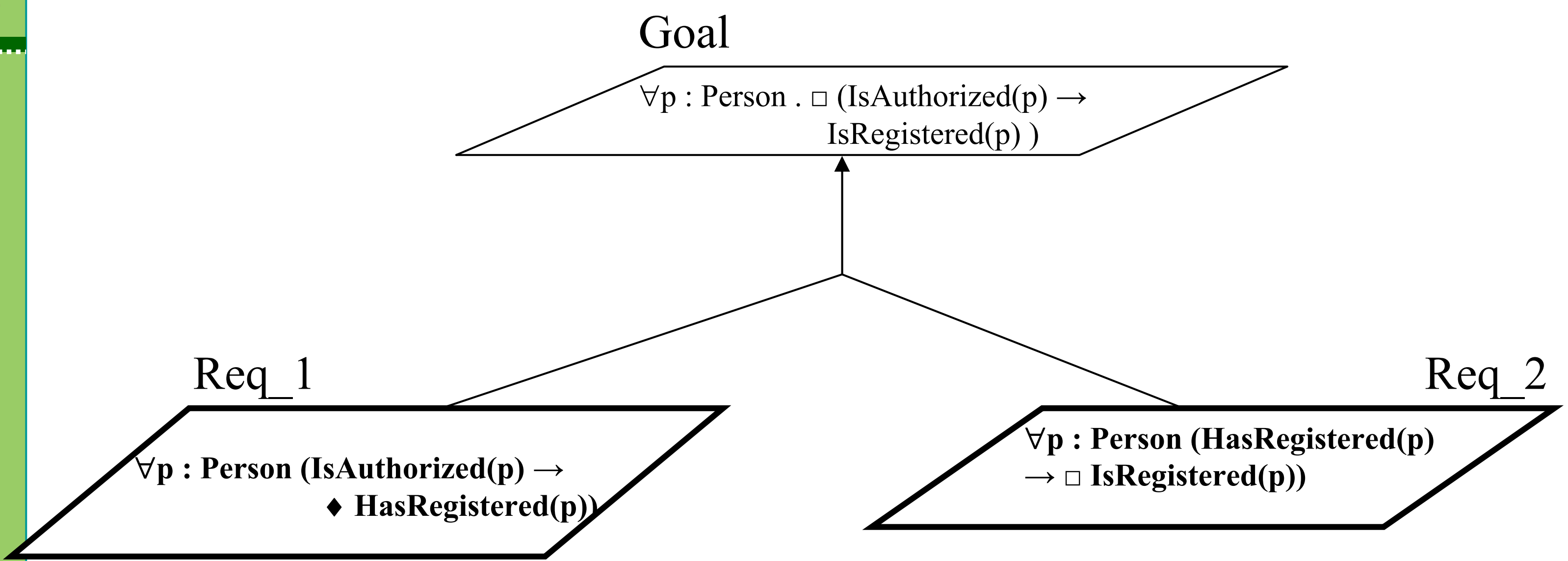
includes B’s identity in msg 2
Now an intruder cannot anymore replay the message since
Alice would expect the intruder’s identity.



The roadmap

- Introduction
- The added value brought by Formal methods
- Formal models in C.C. certification
- Formal tools
- **Cetic experience with formal tools**
- Conclusion

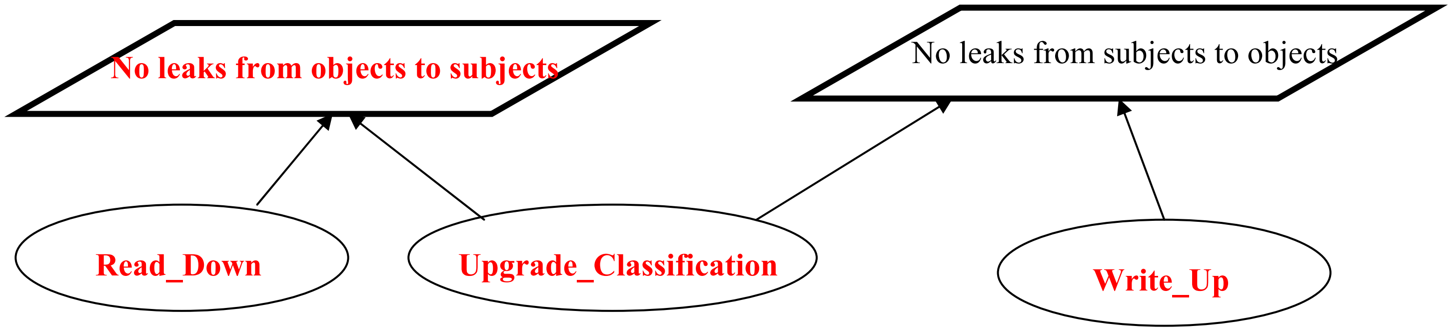
Formal Requirements Models



Check :

$\bigwedge(\text{Req\_1}, \text{Req\_2}) \Rightarrow \text{Goal}$

Formal Requirements Models



fact Initial_knowledge_empty { all s : Subject | no s.knows.(TO/Ord.First)
all o : Object | no o.accessed.(TO/Ord.First) }

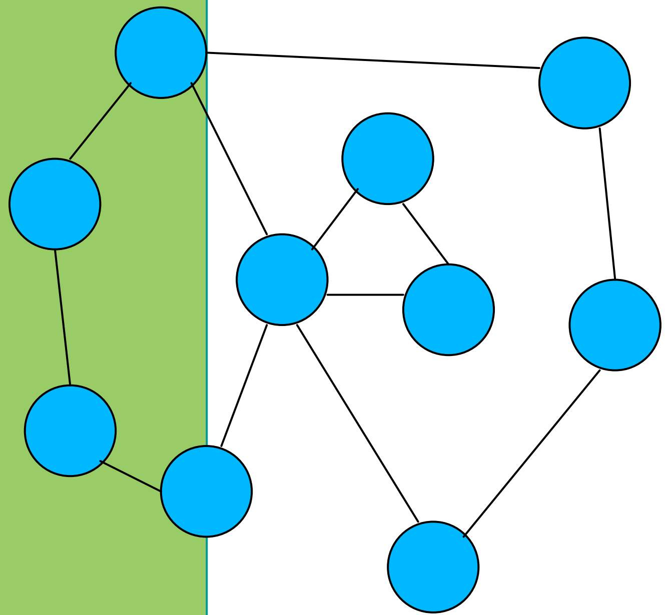
fact Progress {
all t : Tick – TO/Ord.Last | let t' = t.nextTick | some s : Subject | some o : Object |
ReadDown[s,o,t,t'] || WriteUp[s,o,t,t'] || UpgradeClassification[s,o,t,t']
…. + frame conditions !!! …}

assert no_leaks_from_objects_to_subjects {
all t : Tick | all s : Subject | all o : Object | o -> t in s.knows =>
s.clearance in (o.classification.t).canFlow }

check no_leaks_from_objects_to_subjects ….

Formal Requirements Models : Acceptance Test Generation

Finite State Model



Model checking tool

Witness Trace (acceptance test objective)

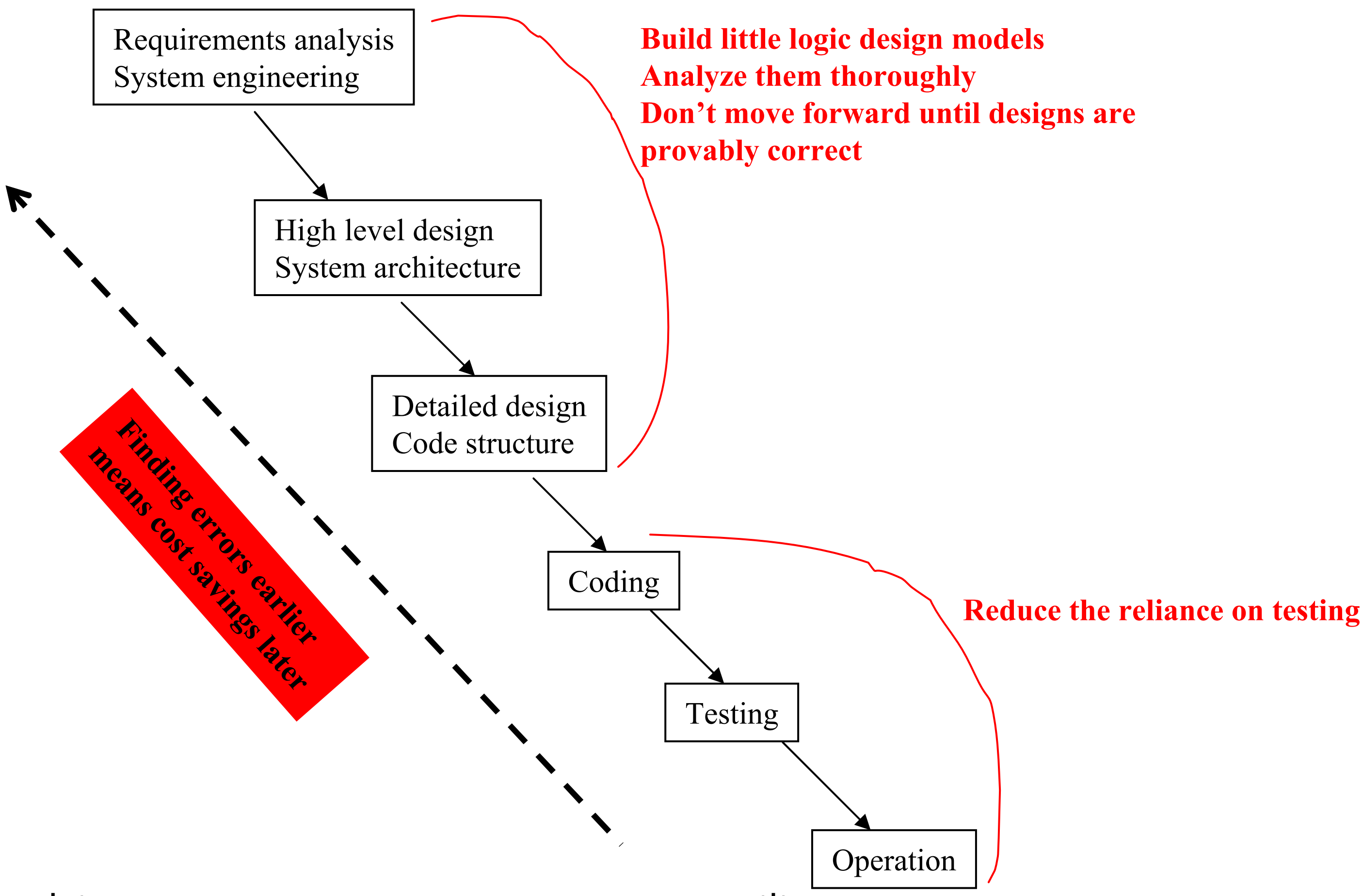
Trap Property : obstacle or any negated property of interest
◊ (some t : Train | moving(t) /\ doors_open(t))

The roadmap

- Introduction
- The added value brought by Formal methods
- Formal models in C.C. certification
- Formal tools
- Cetic experience with formal tools
- **Conclusions**

Conclusions

Basic idea of formal methods :



Conclusions

What is the challenge ?

Use the technology of formal methods :

- to augment traditional methods and tools
- to automate traditional processes (e.g. testing !)

To do this :

- unobstrusively extract formal specification & properties
- deliver results in a familiar form

Conclusions

Main benefits from Req. Engineering experience :

- **a formal model is the best critics you can find:**
  it helps to formulate the right questions
  and checks that you get the right answers

- **a formal model is an invaluable communication tool**
  with the stakeholders

IT security context :

access control, information flows, protocols, PKI, ….

Bibliography

Role Based Access Control Models (R.S. Sandhu, E.J. Coyne)

Lattice-Based Access Control (R.S. Sandhu)

Spin course : http://spinroot.com/spin/Doc/course (G. Holzmann)

Model checking : a Tutorial Overview (Stephen Merz)

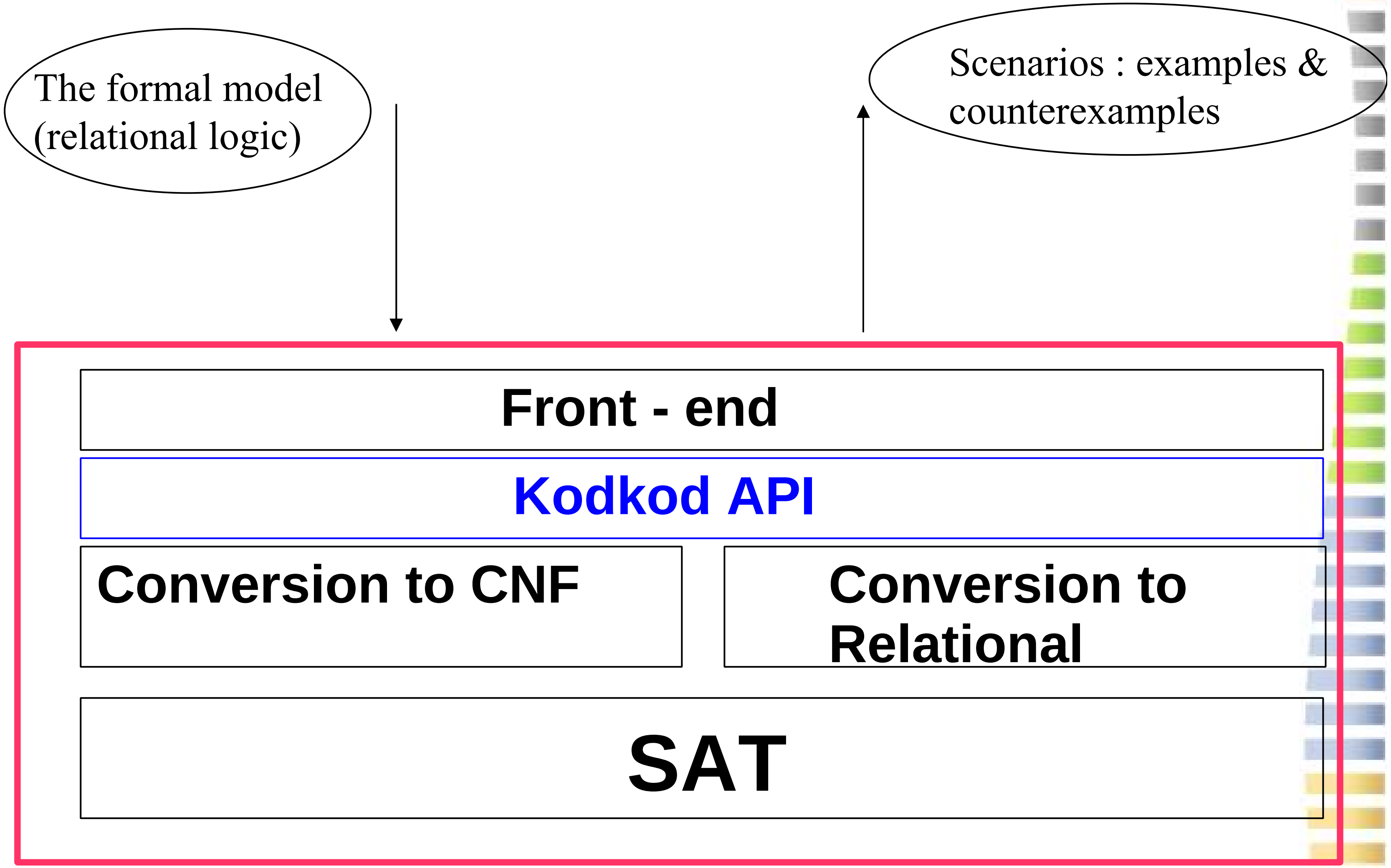
Software Abstractions (D. Jackson)

Effective Test Generation (J. Rushby) :
http://www.csl.sri.com/users/rushby/slides/efftestgen.pdf

Formal Security Analysis (D. von Oheimb) :
http://david.von-oheimb.de/cs/talks/index.html

The Needham Schroeder protocol in SAL (J. Rushby) :
http://www.csl.sri.com/users/rushby/abstracts/needham03

The Alloy analyzer



Logic model checker : how does it work ?

- system : L(S) (the set of all possible behaviors of S)
- property : L(p) (the set of valid/desirable behaviors)
- prove that : $L(S) \subseteq L(p)$ (everything possible is valid)
- method :

To prove $L(S) \subseteq L(p)$ we can prove

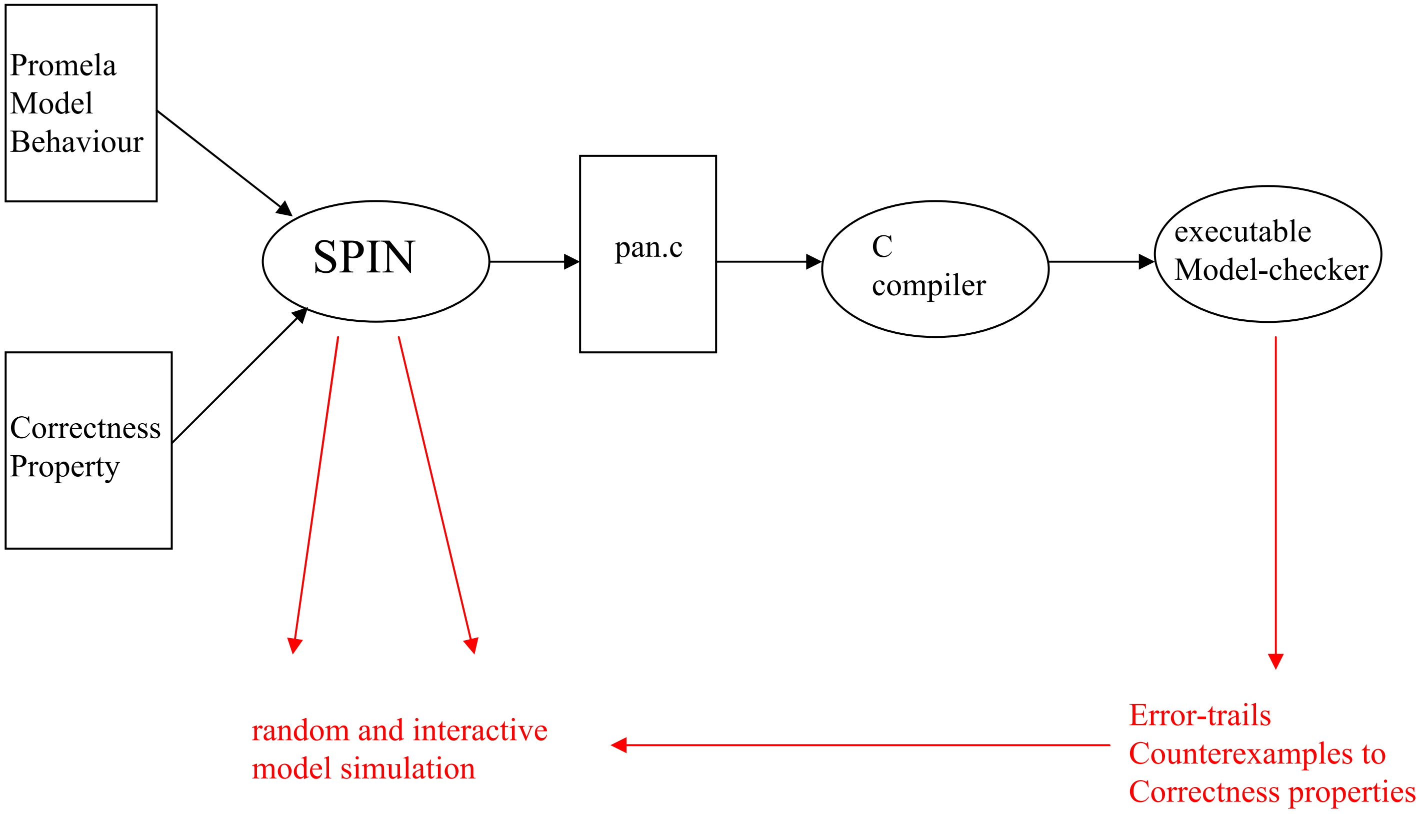
$$L(S) \cap (\Sigma_\omega \setminus L(p)) = \emptyset$$

which is the same as

$$L(S) \cap L(\neg(p)) = \emptyset$$

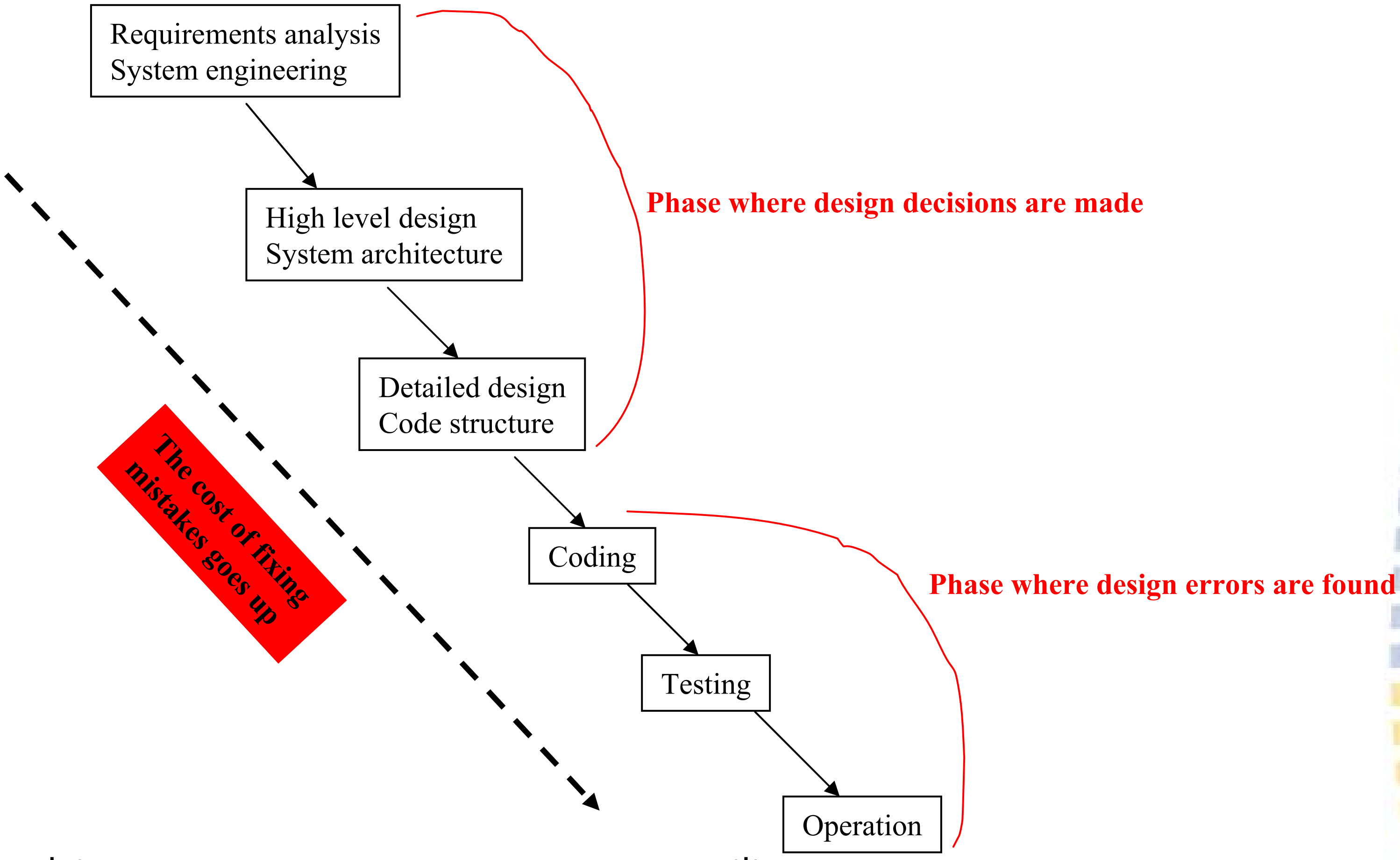
Spin's verification engine

Logic model checker : how does it work ?



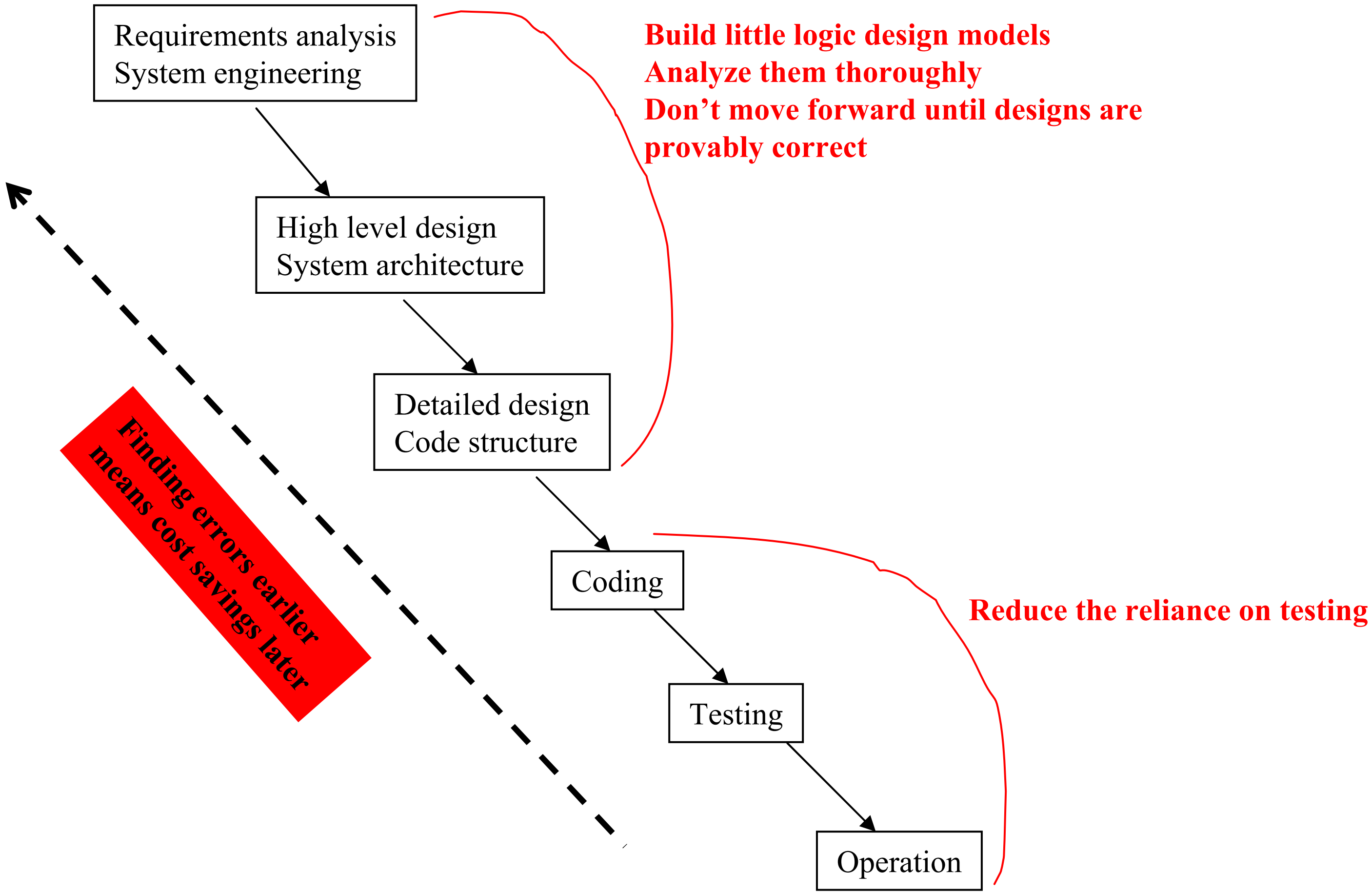
Conclusions

Basic idea of formal methods :



Conclusions

Basic idea of formal methods :



BLP : the Object Model