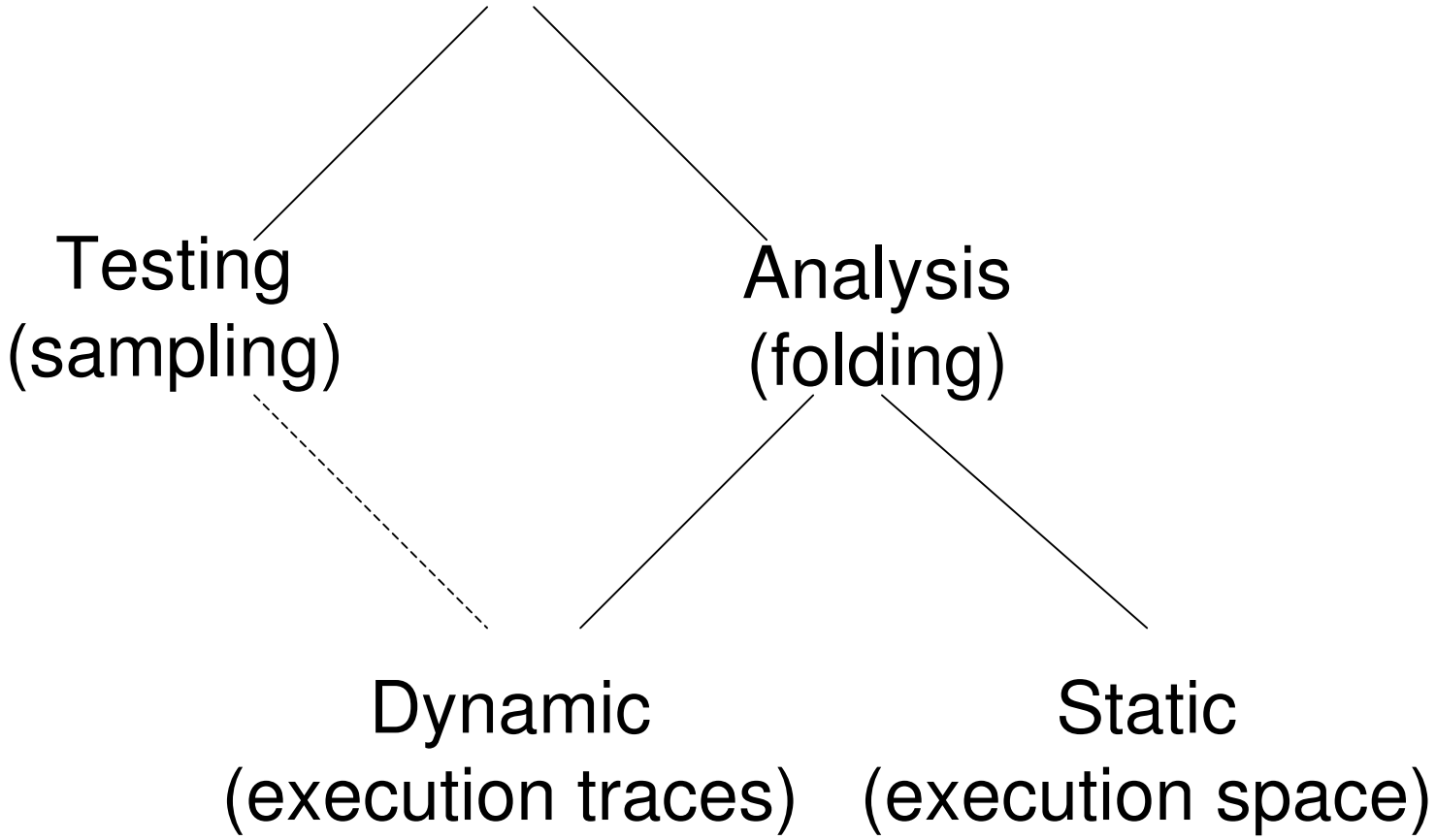
Dynamic Analysis for Self-Healing Software

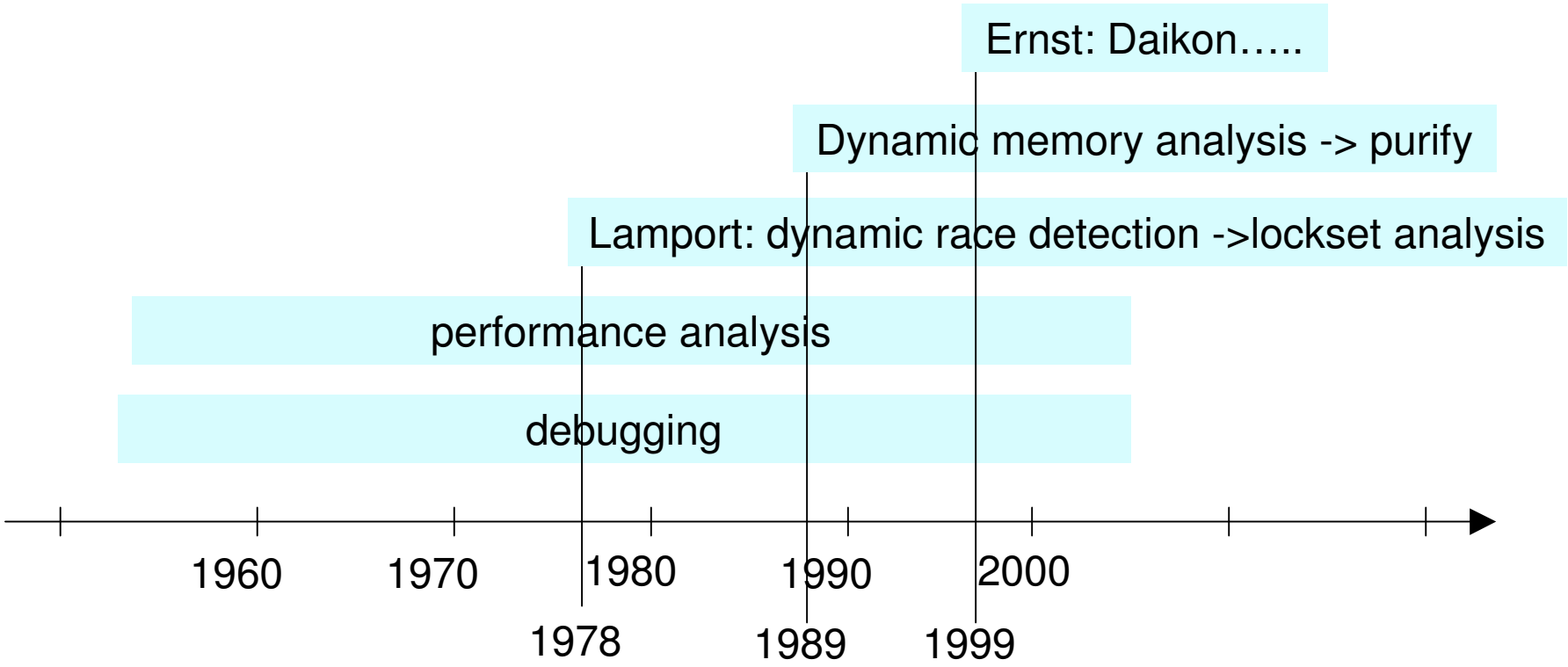
Mauro Pezzè
Università degli Studi di Milano Bicocca
University of Lugano

PART I: Dynamic Analysis

Program verification



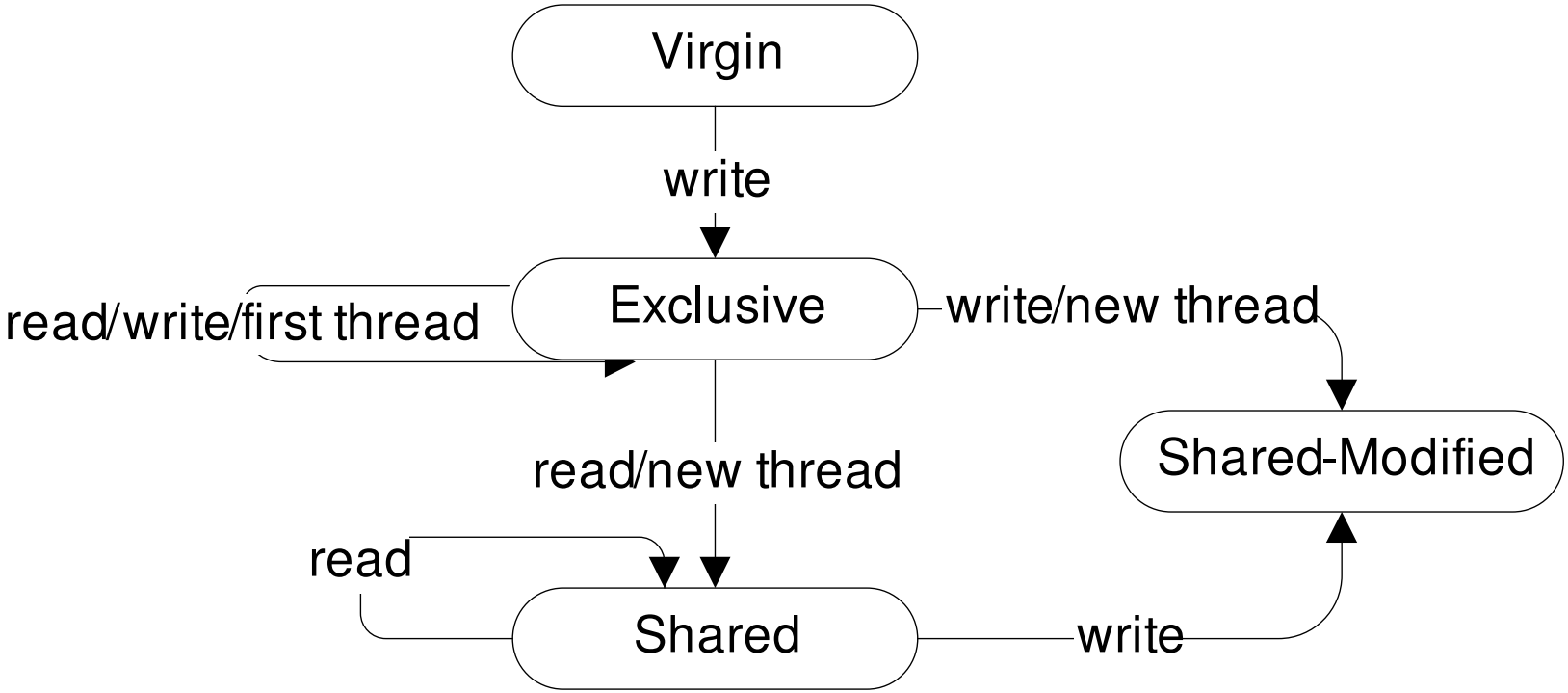
Dynamic analysis



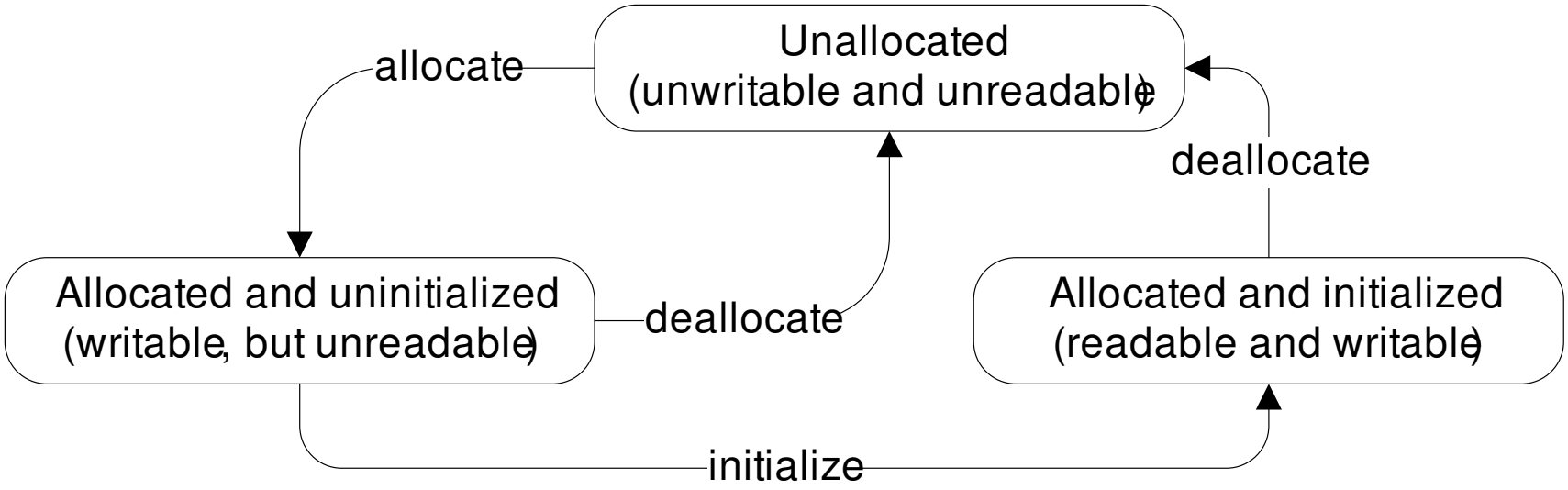
Dynamic lockset analysis

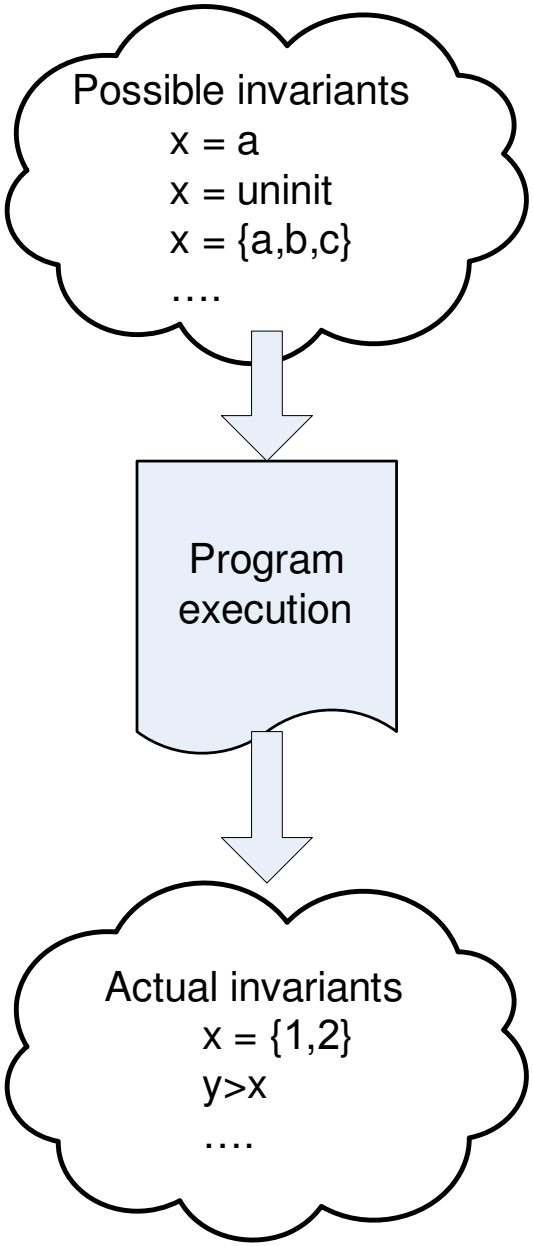
Tread	Program trace	locks held	lockset(x)
thread A	lock(lck1)	{}	{lck1, lck2}
	x=x+1	{lck1}	{lck1}
	unlock(lck1)	{}	
thread B	lock(lck2)	{lck2}	
	x=x+1		{}
	unlock(lck2)	{}	

Lockset analysis



Dynamic memory analysis





Daikon

POSSIBLE INVARIANTS

over any variable *x*:

x = *a*

x = *uninit*

x = {*a*,*b*,*c*} for a small set of values

over a single numeric variable *x*:

x >= *a*, *x* <= *b*, *a* <= *x* <= *b*

x ¬ = 0

x = *a* (mod b)

x ¬ = *a* (mod b)

over two numeric variables *x* and *y*:

y = *ax*+*b*

x <= *y*, *x* < *y*, *x* = *y*, *x* ¬ = *y*

x = *f n*(*y*)

over the sum of two numeric variables *x*+*y*:

x+*y* >= *a*, *x*+*y* <= *b*, *a* <= *x*+*y* <= *b*

x+*y* ¬ = 0

x+*y* = *a* (mod b)

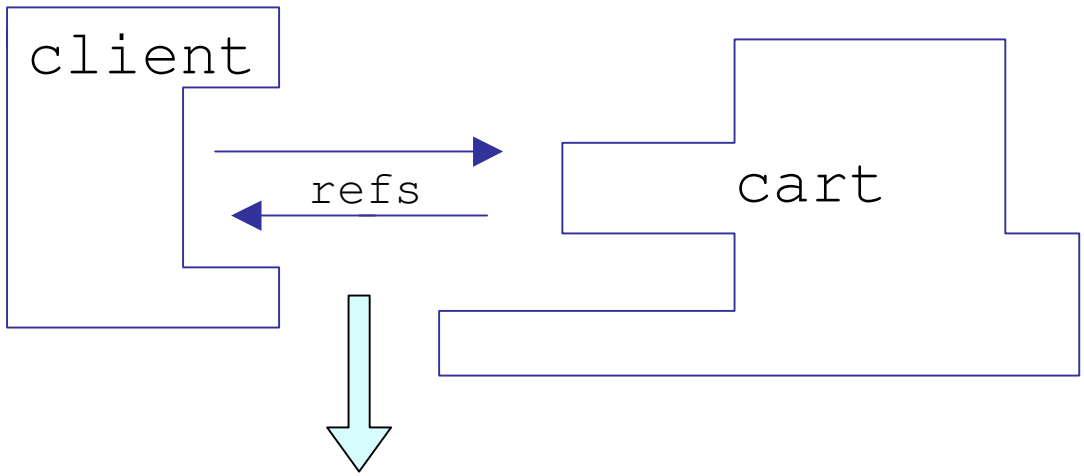
x+*y* ¬ = *a* (mod b)

……

Dynamic analysis of COTS components

From simple variables to complex objects

- Extract information with aspect programming
- Identify field values with reflection
- Derive invariant on objects' fields (Daikon)



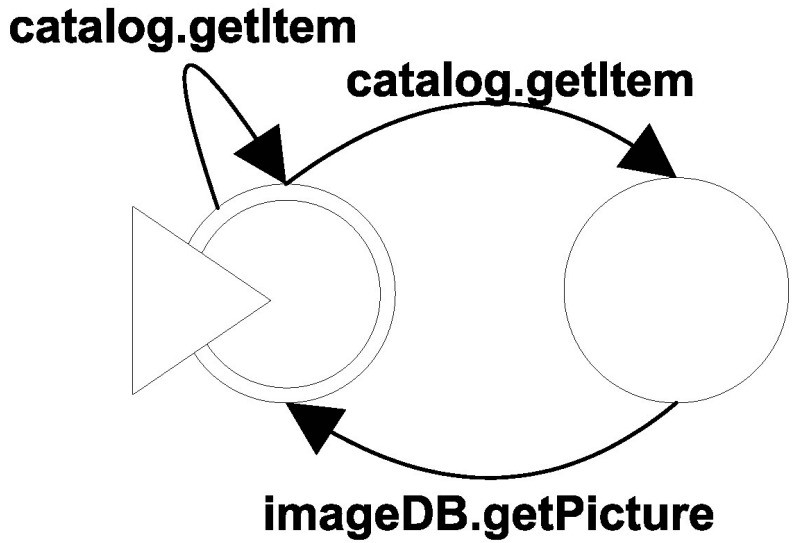

```
cartitem.getUnitCost <= cartitem.getTotal
```

Dynamic analysis of subsystems' interactions

single interactions == words over a regular language

```
getitem getitem getpicture
getitem getitem getitme getpicture
```

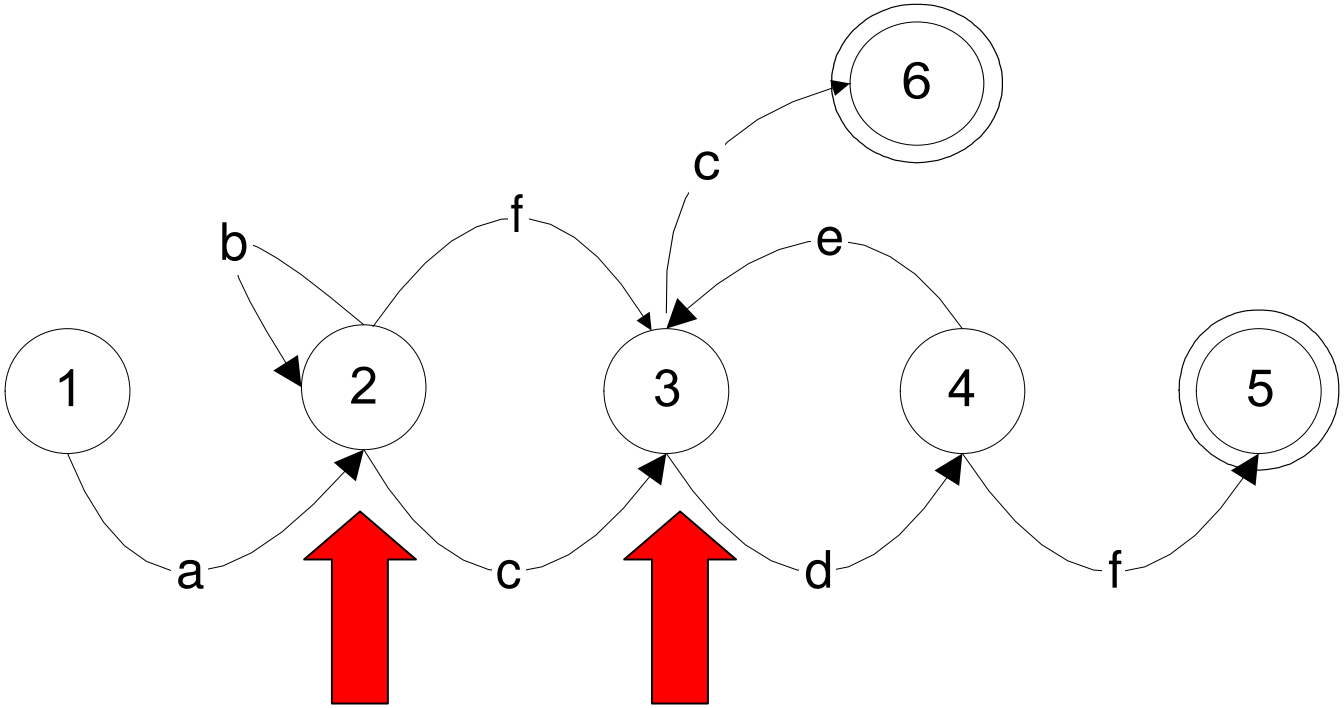
interaction models == FSA



Deriving Interaction Models

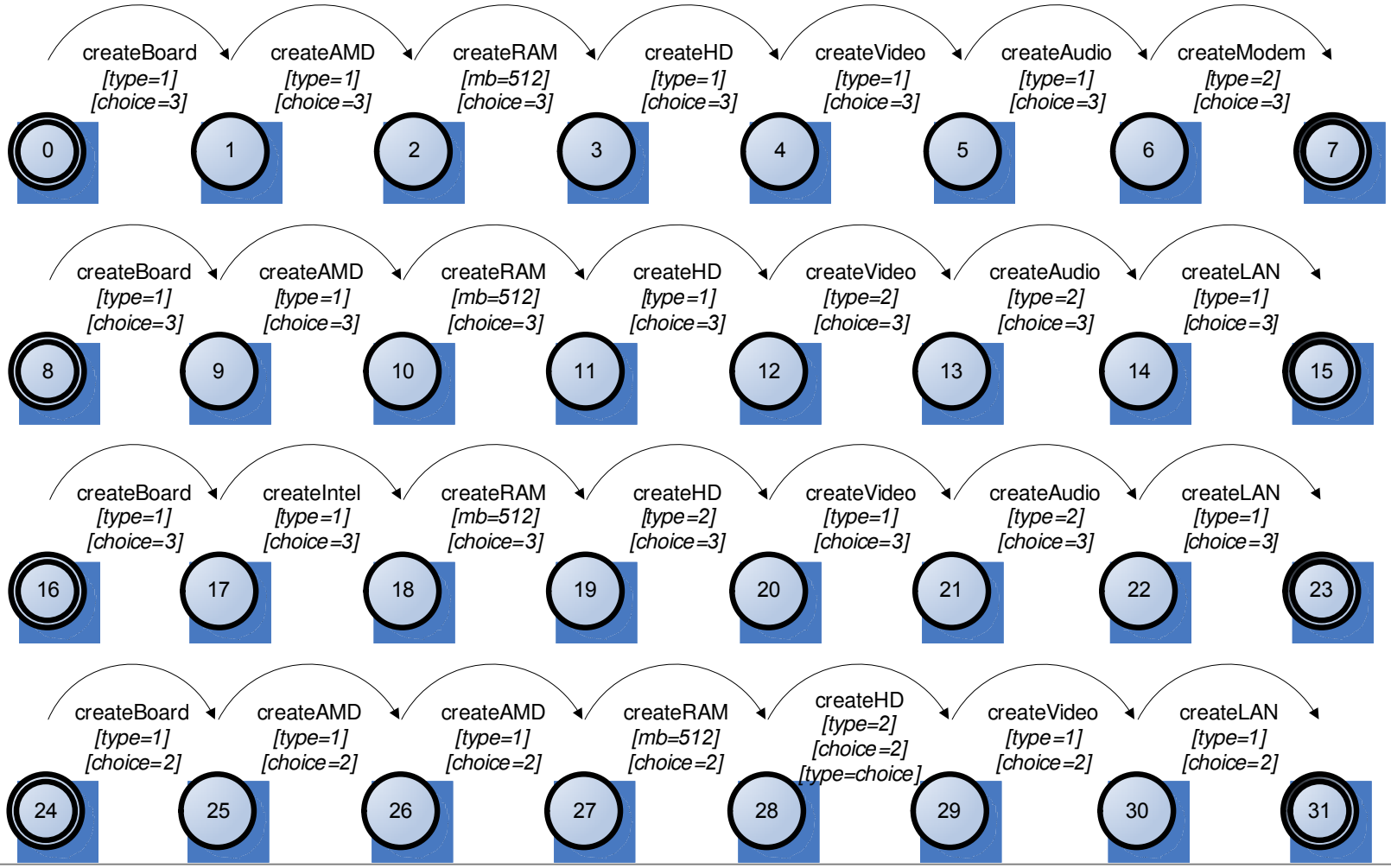
- From samples to FSA
  - only positive samples
  - shared sub-behaviors
  - no teacher
  - incremental algorithm
  - add-and-delete sequence

kBehavior

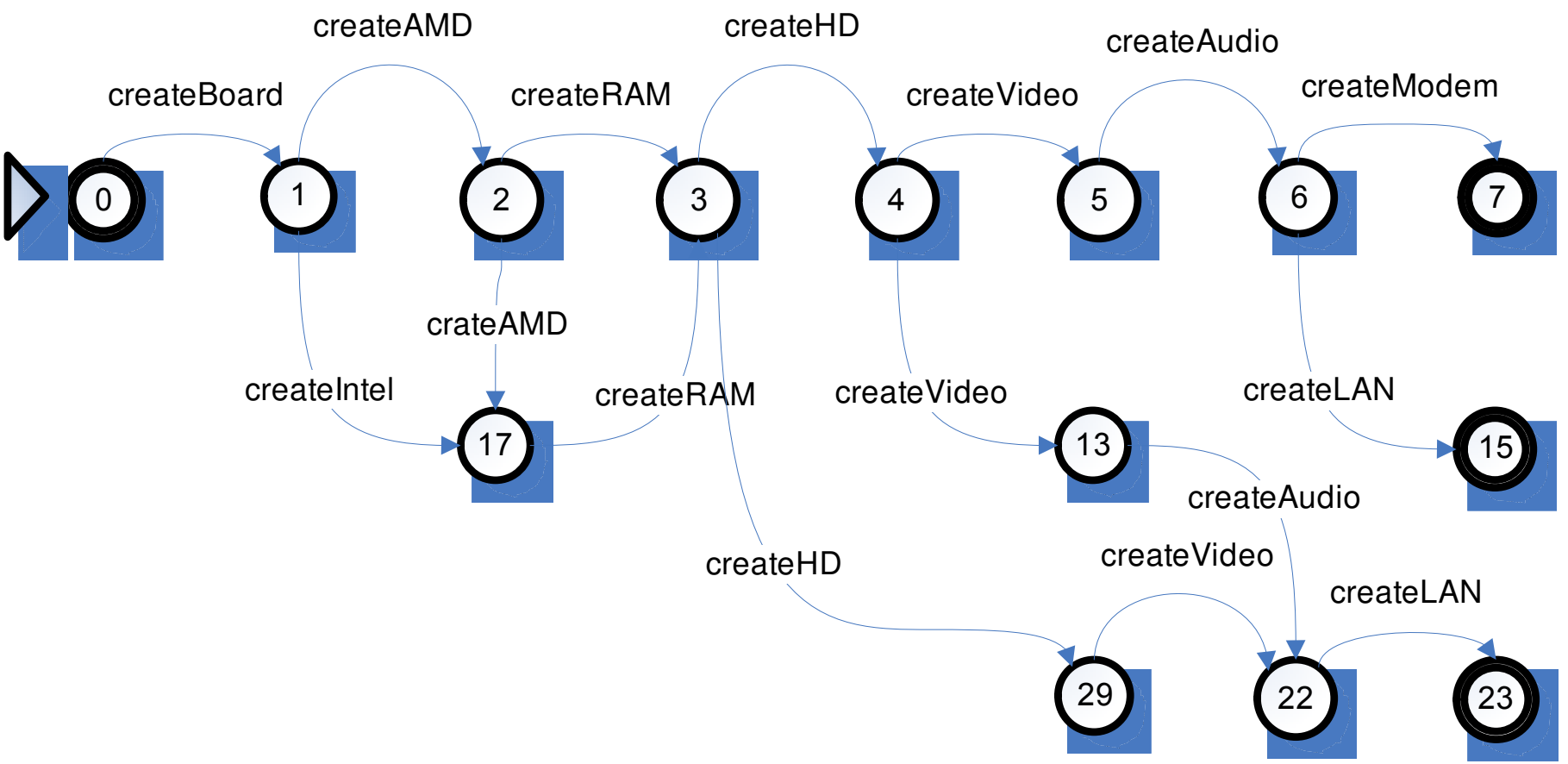


a b b f d e d e c
a b b f d e d e c

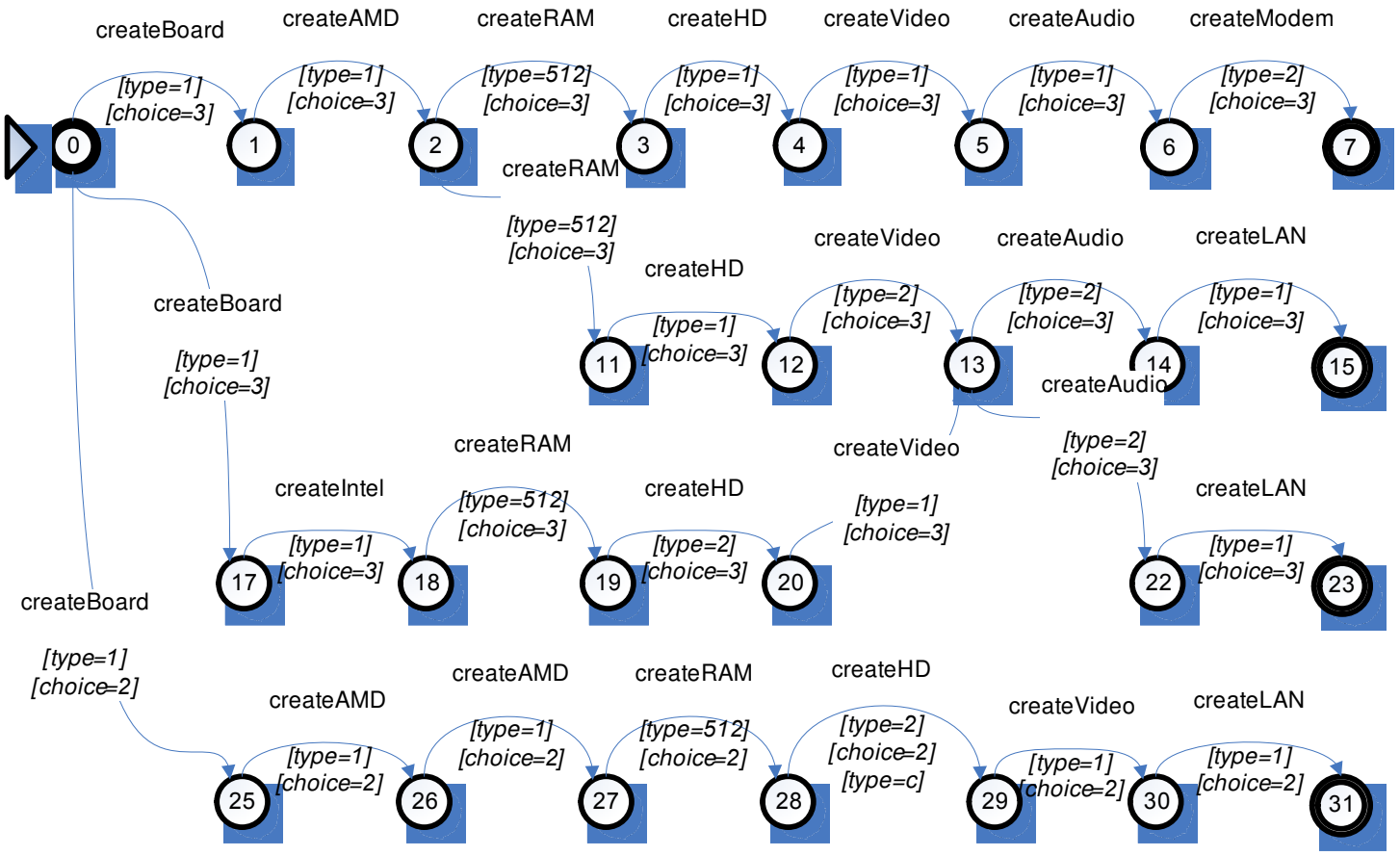
From FSA to extended FSA: a simple set of traces



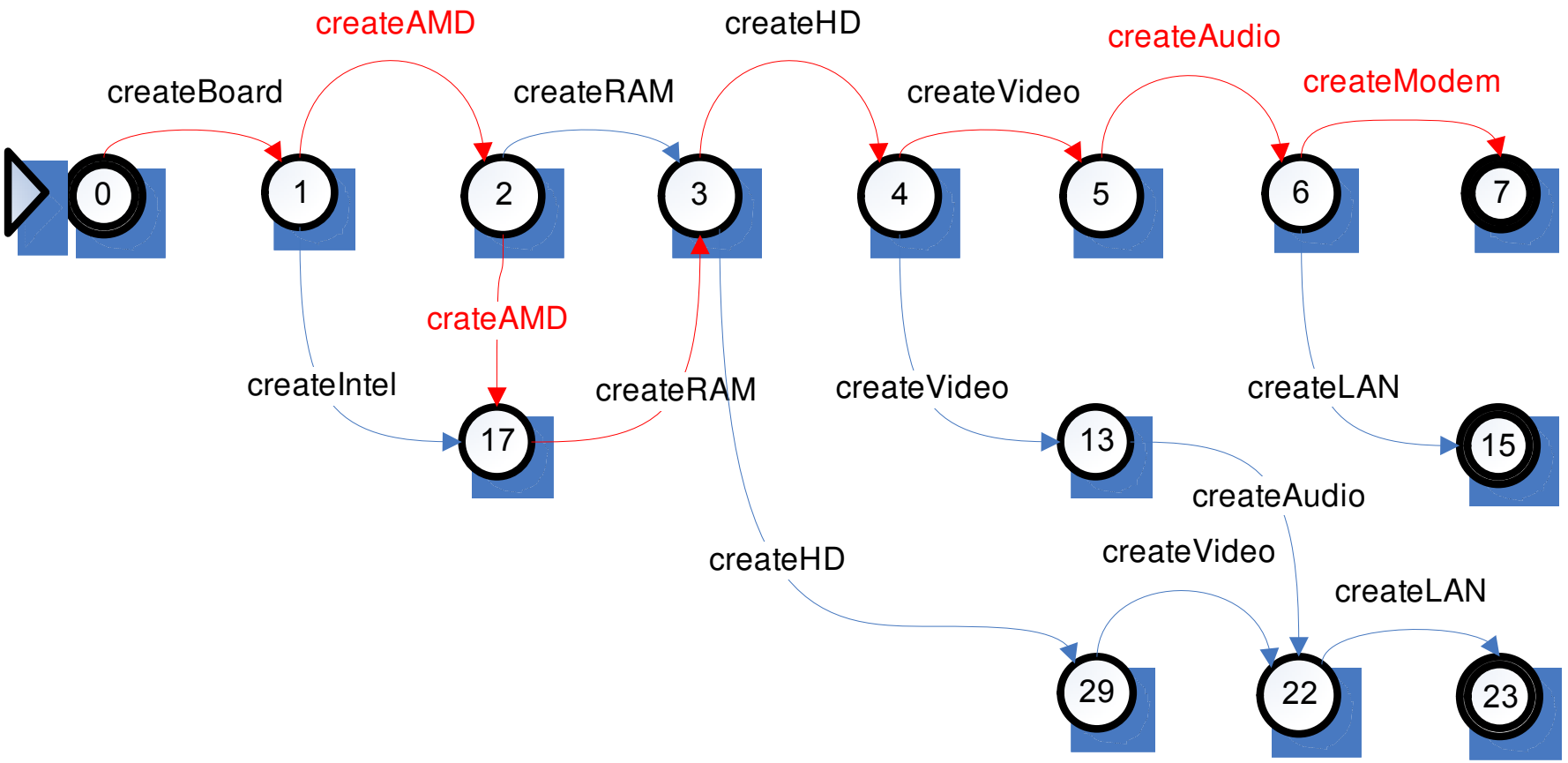
A simple FSA


createAMD
createHD
createAudio
createBoard
createRAM
createVideo
createModem
0
1
2
3
4
5
6
7
crateAMD
createIntel
createRAM
createVideo
createLAN
17
13
15
createAudio
createHD
createVideo
createLAN
29
22
23

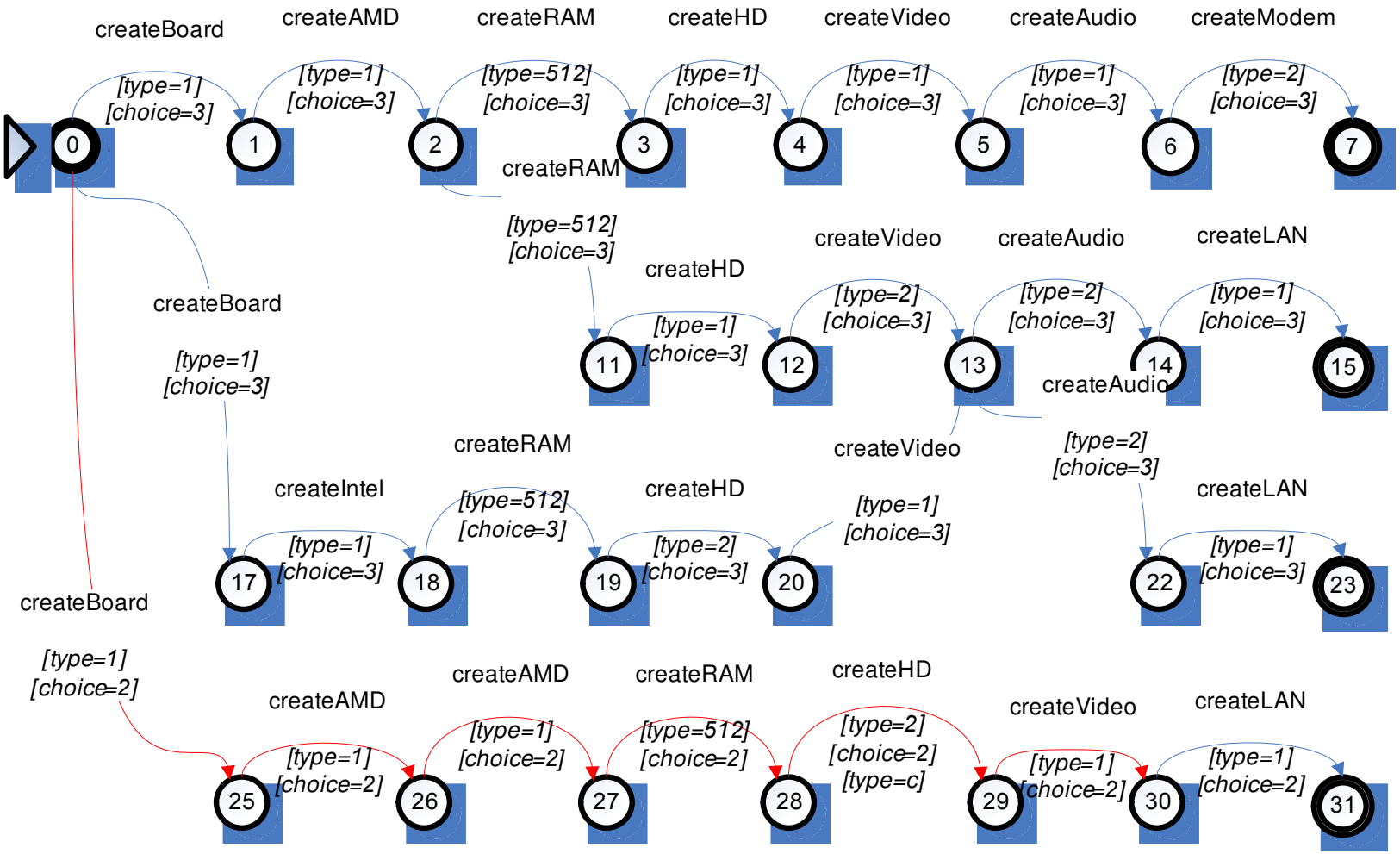
An annotated FSA



Precision: the simple FSA

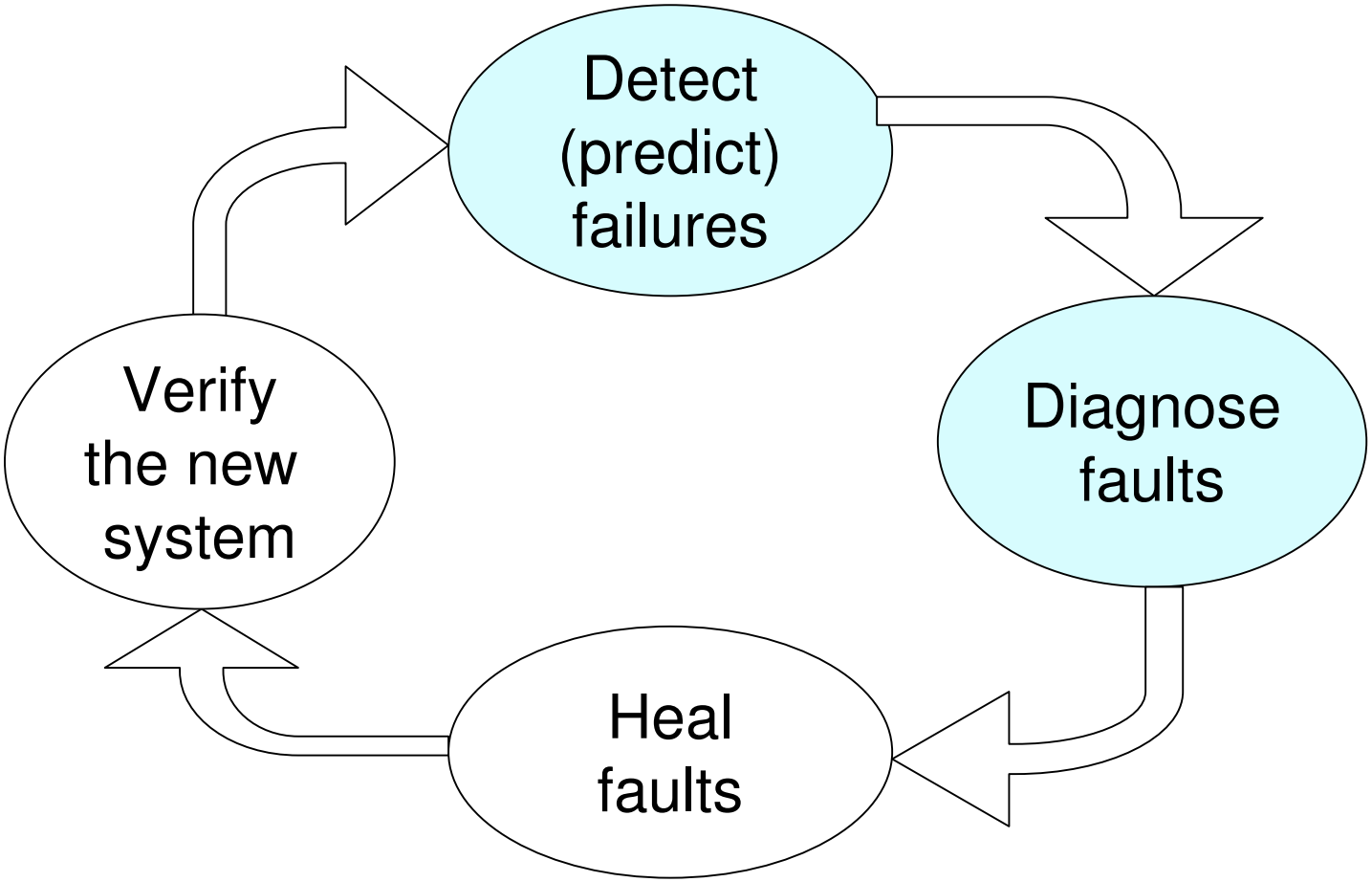


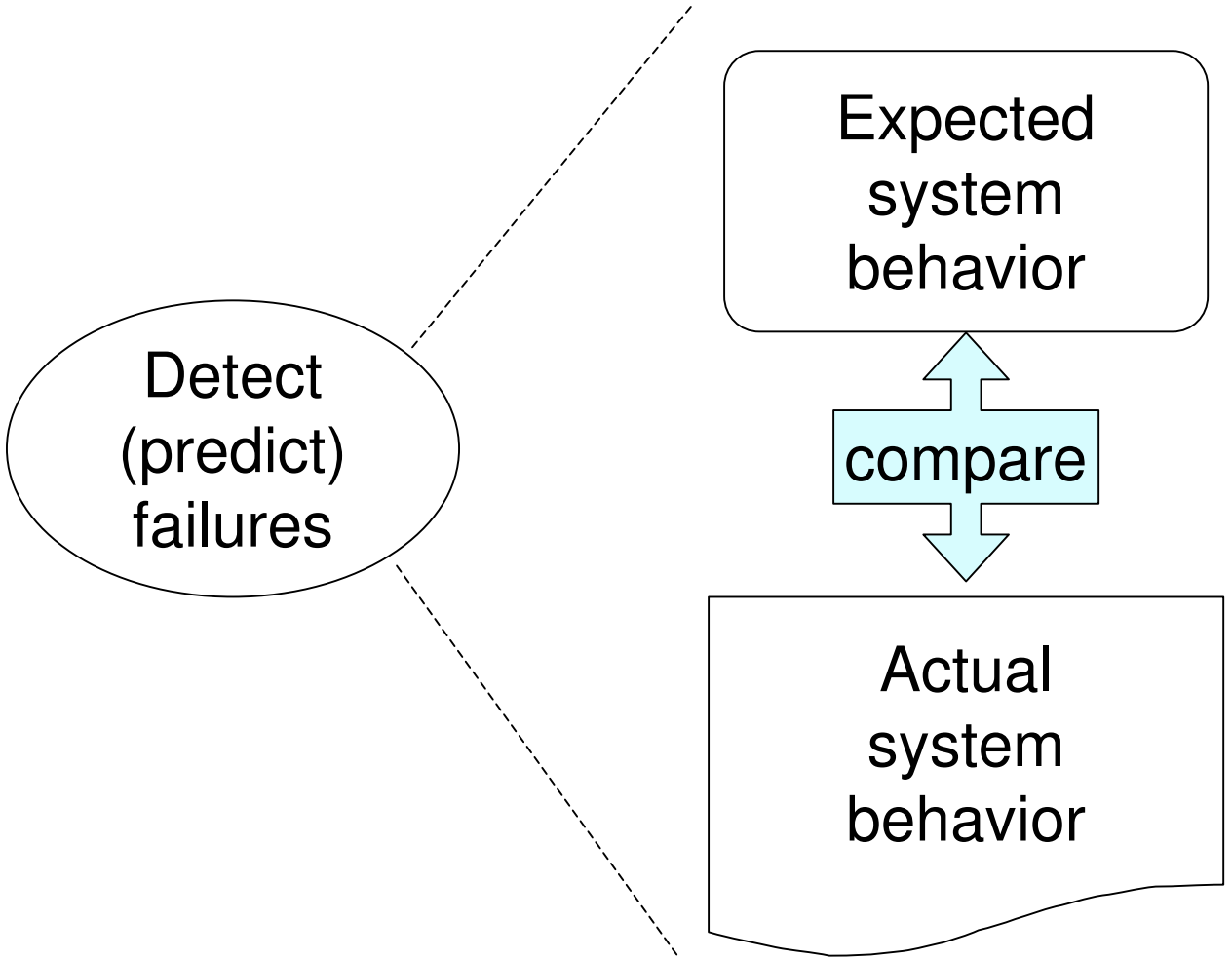
Precision: the annotated FSA



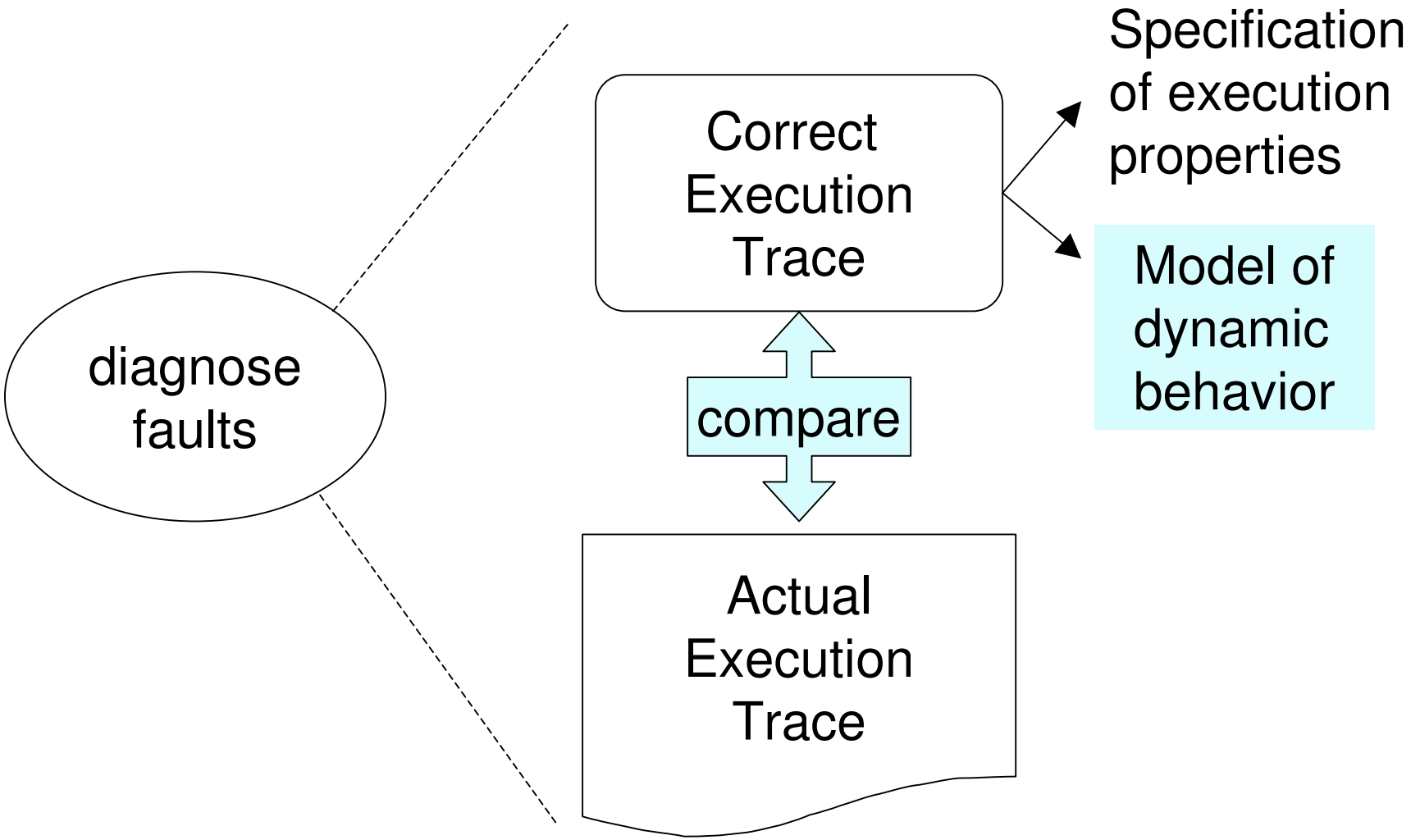
PART II: Self-Healing Software

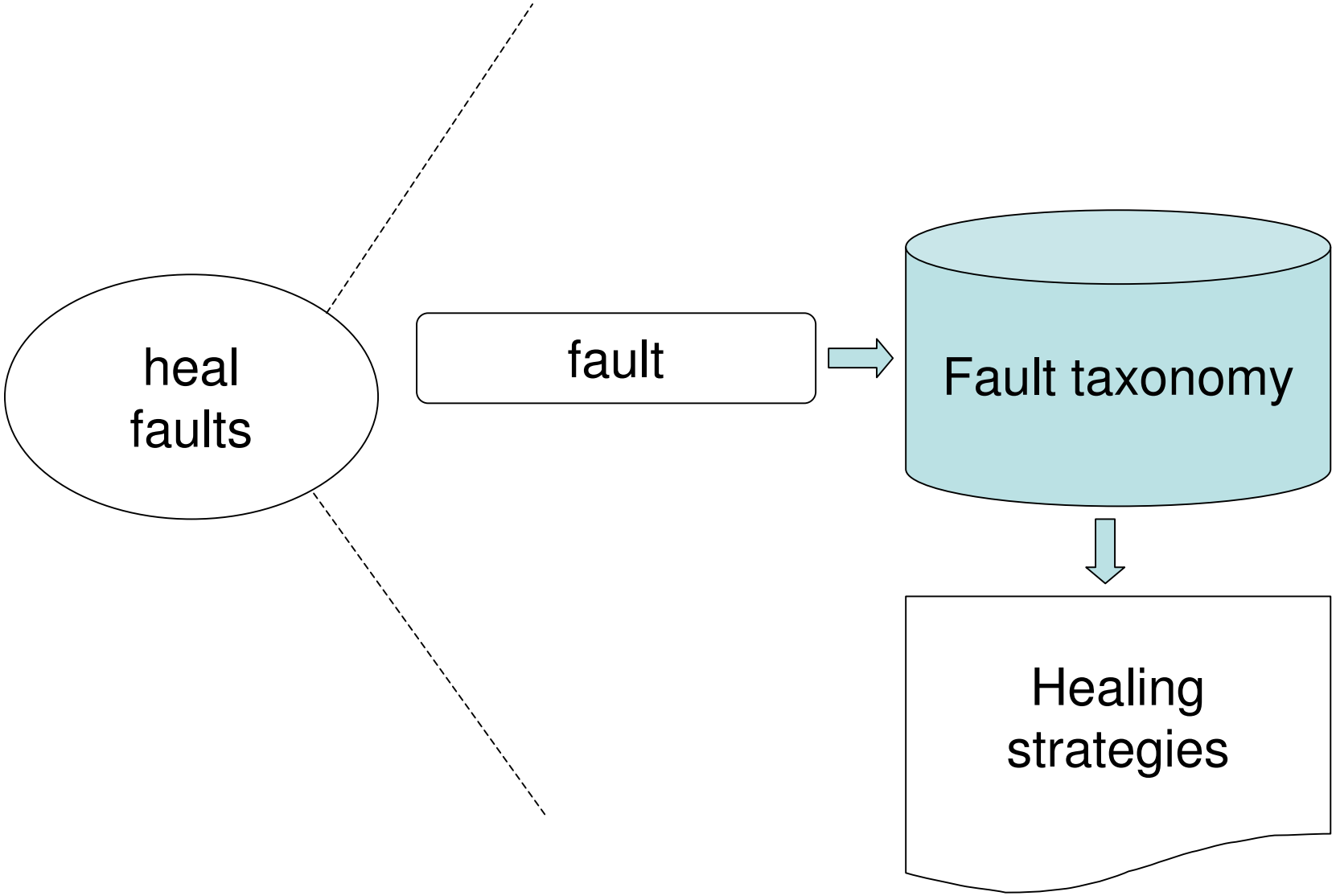
The self-healing cycle



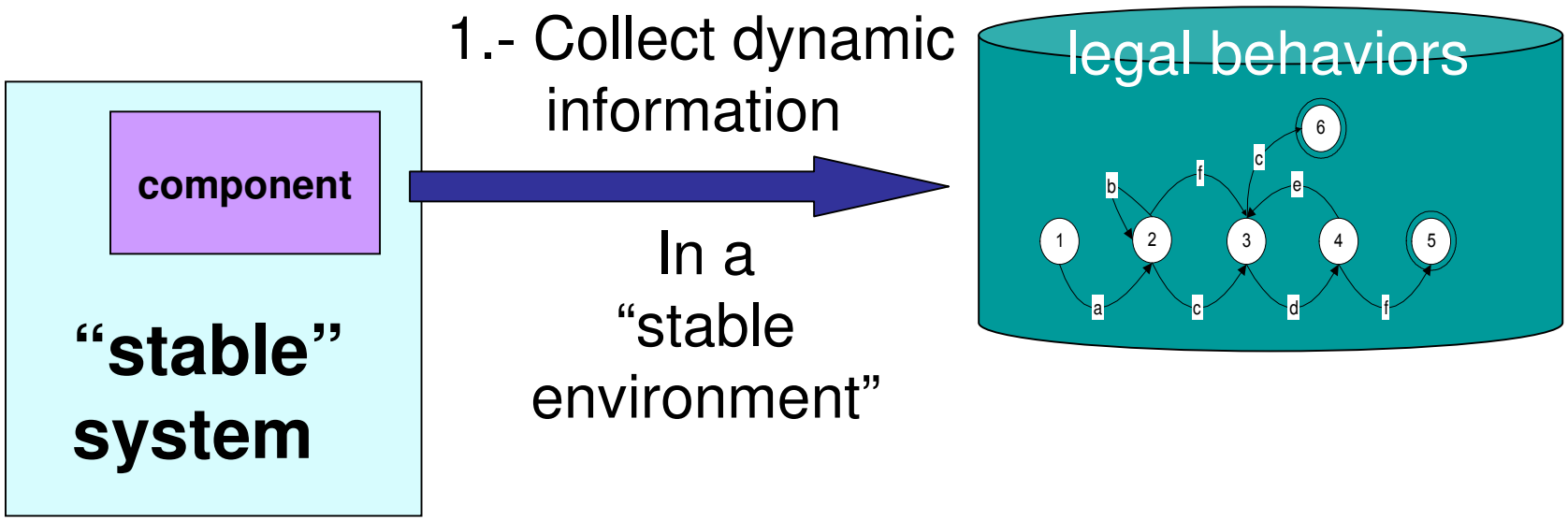


We need a model of the system expected behavior

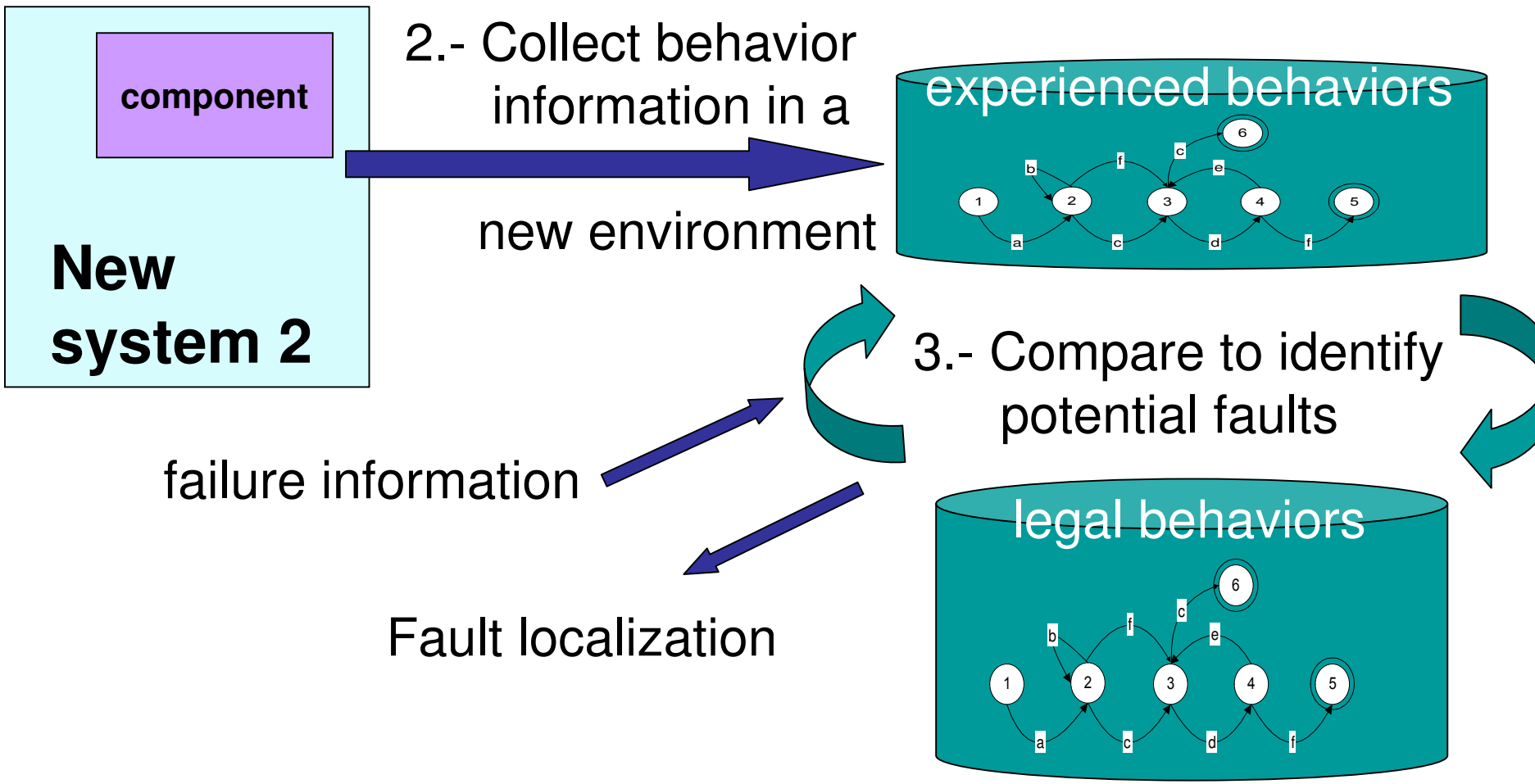

diagnose faults
Correct Execution Trace
Specification of execution properties
Model of dynamic behavior
compare
Actual Execution Trace


heal
faults
fault
Fault taxonomy
Healing
strategies

Dynamic models to diagnose faults

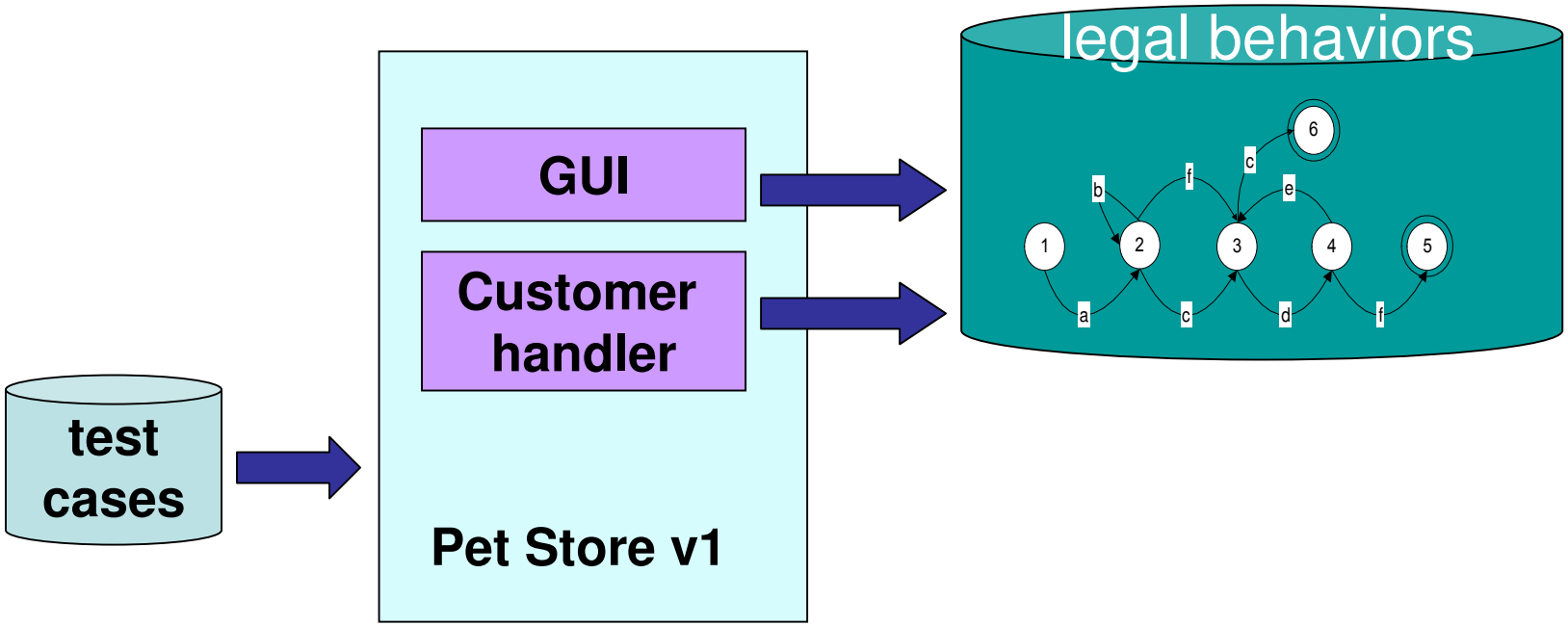


Dynamic models to diagnose faults

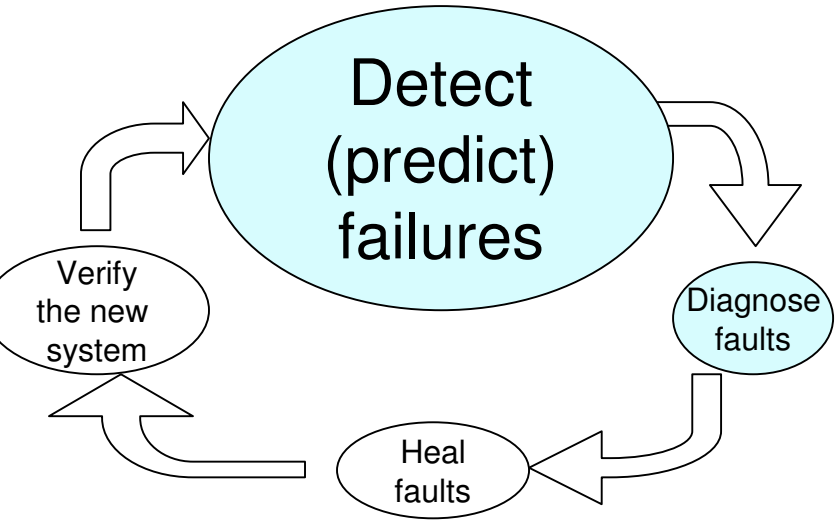


Sun's Pet store

generate information for fault diagnosis



Pet Store



Failure condition: Required(e-mail address)

Detect failures

No e-mail address

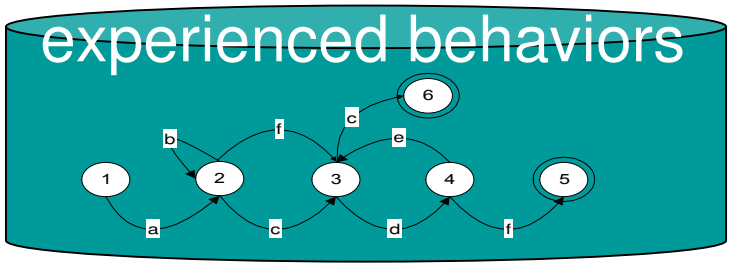
inputs

New GUI

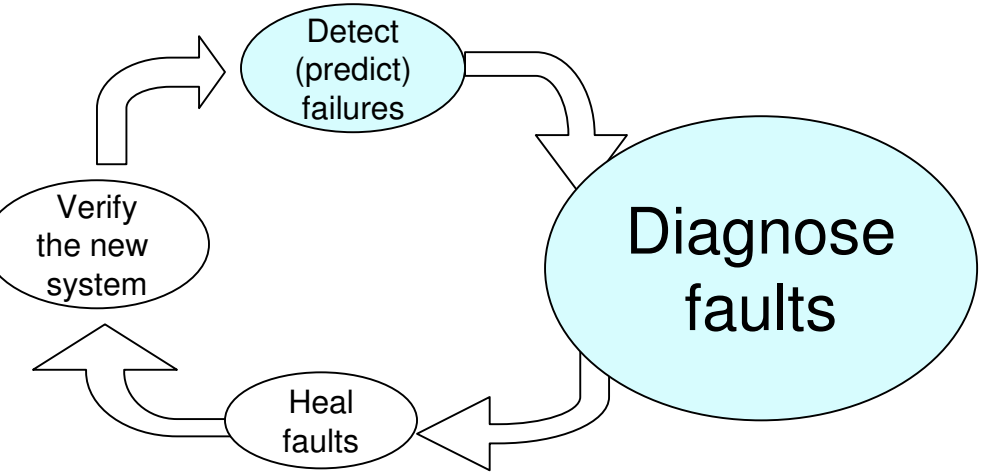
Modified Customer handler

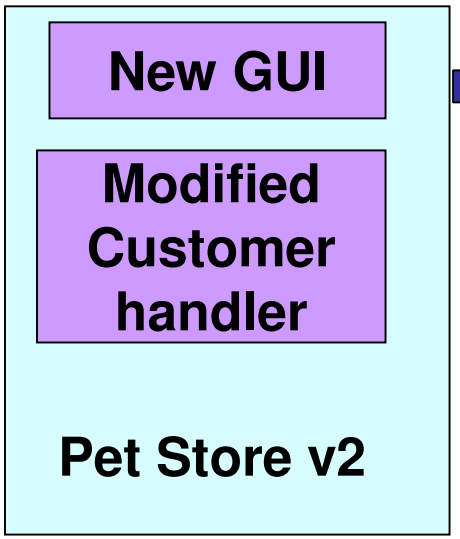
Pet Store v2

Record behaviors

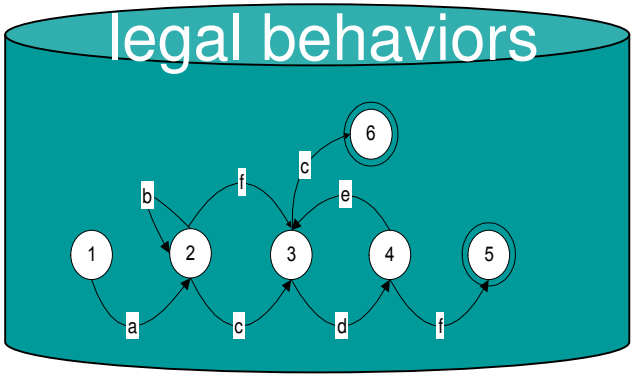


Pet Store

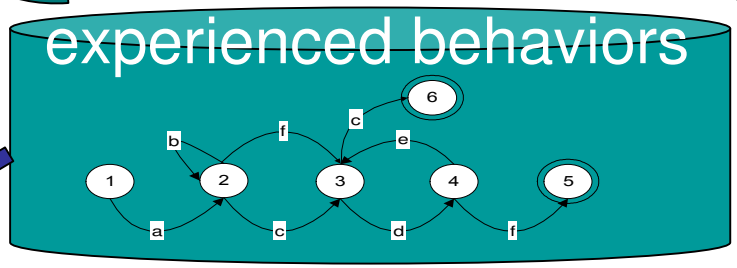




failure



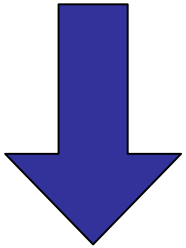
differences



CustomerEvent.getContactInfo.getEmail.length>5

Pet Store: fault localization

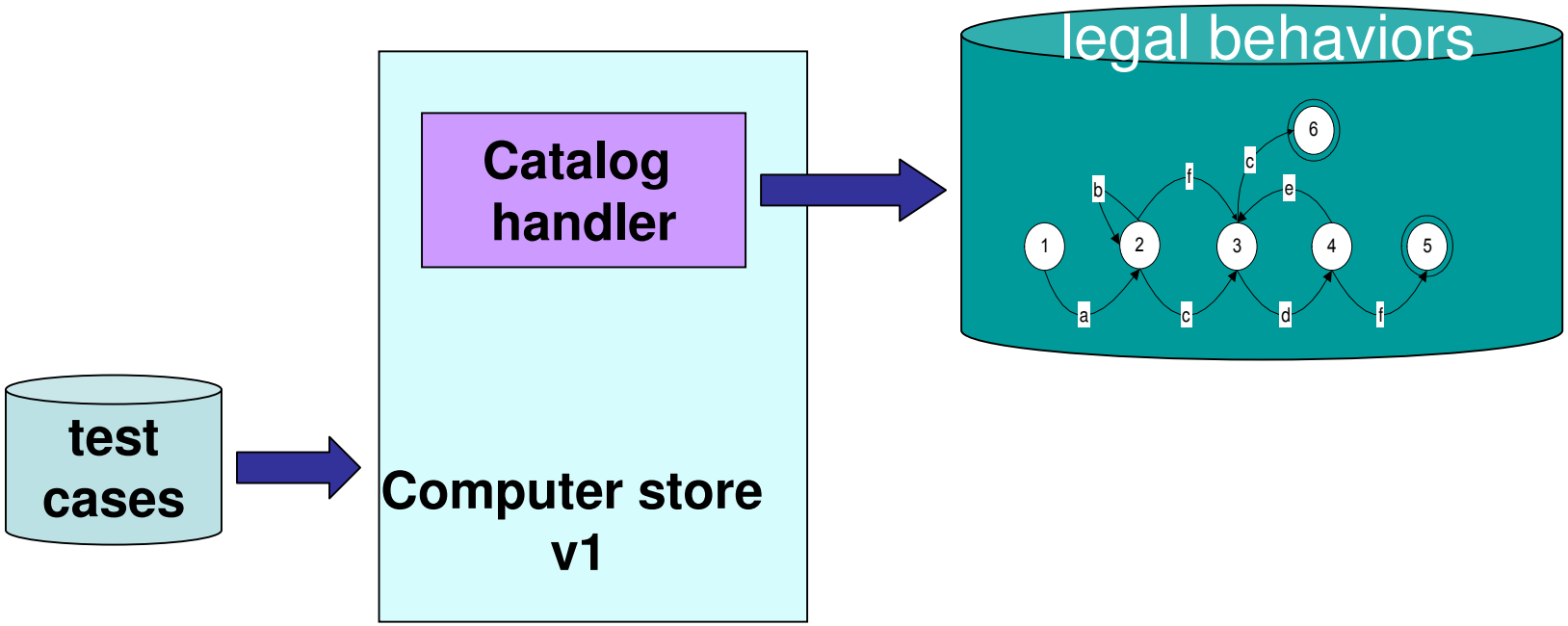
CustomerEvent.getContactInfo.getEmail.length>5



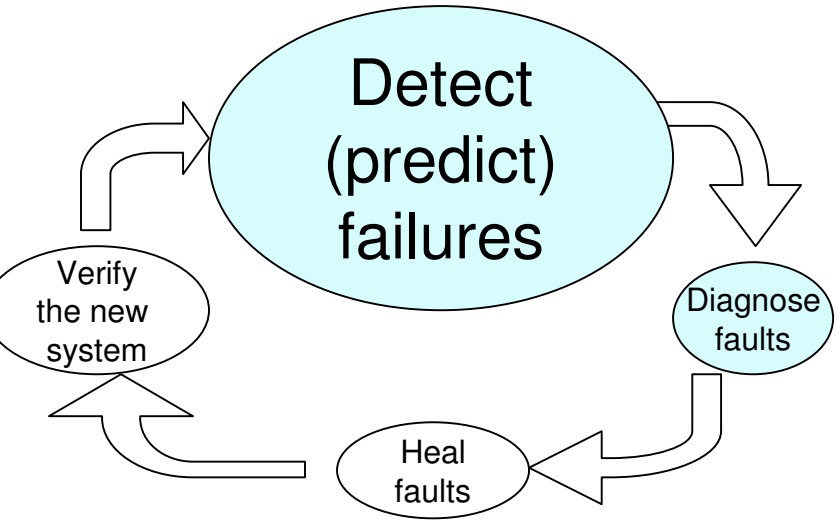
The value of field length
is not checked for
Non emptiness

Computer store

generate information for fault diagnosis



Computer Store



Failure condition: Return (complete catalog)

Detect failures

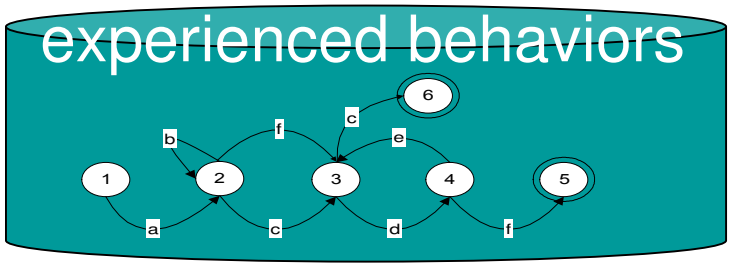
Incomplete catalog

inputs

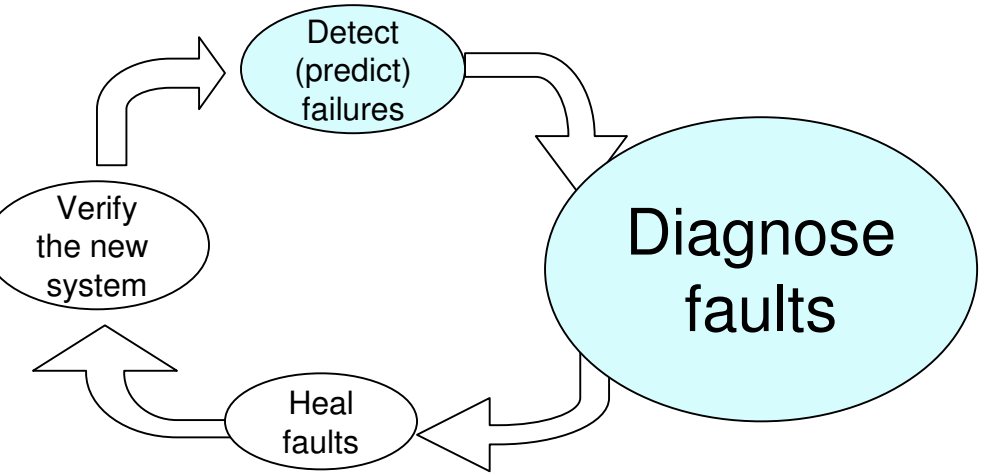
New Catalog handler

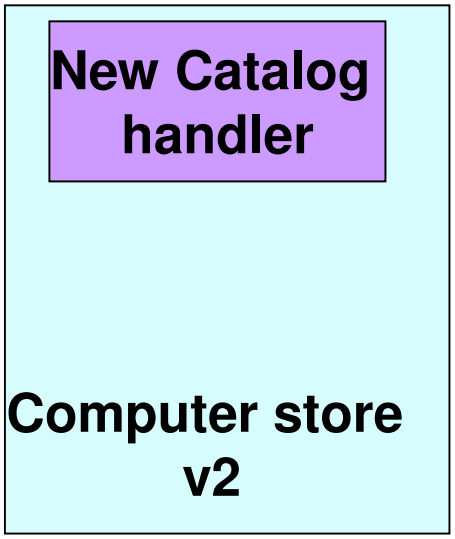
Computer store v2

Record behaviors

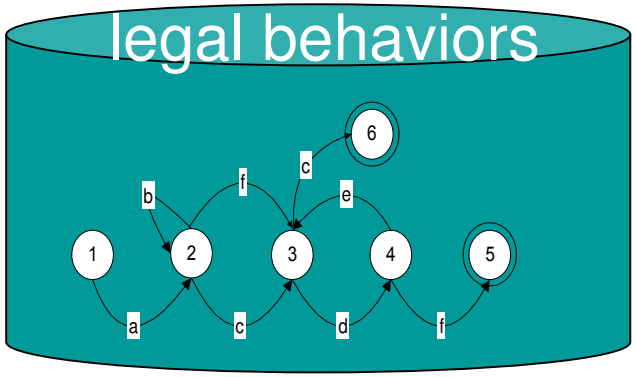


Computer Store

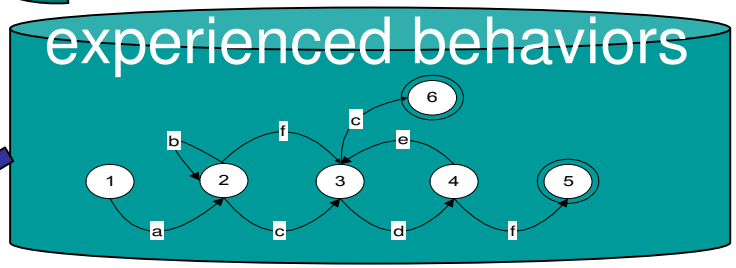




failure



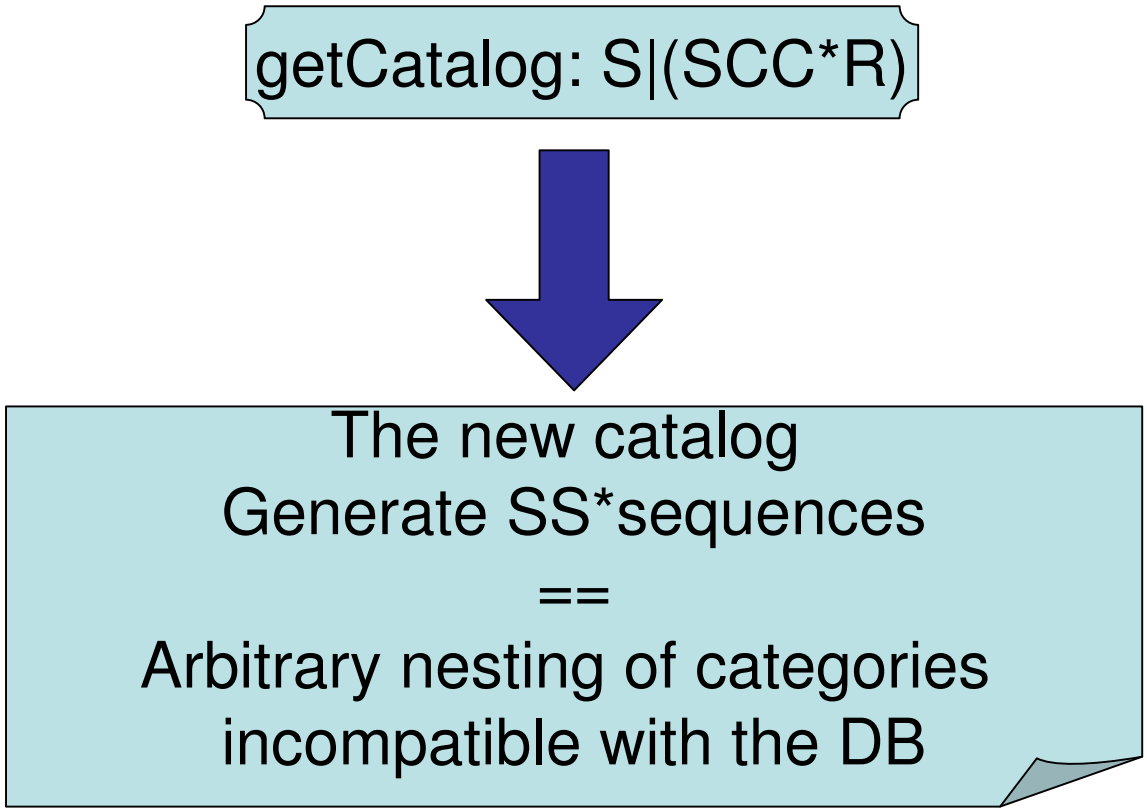
differences



getCatalog: S|(SCC*R)

S = locate service
C = extract catalog
R= produce results

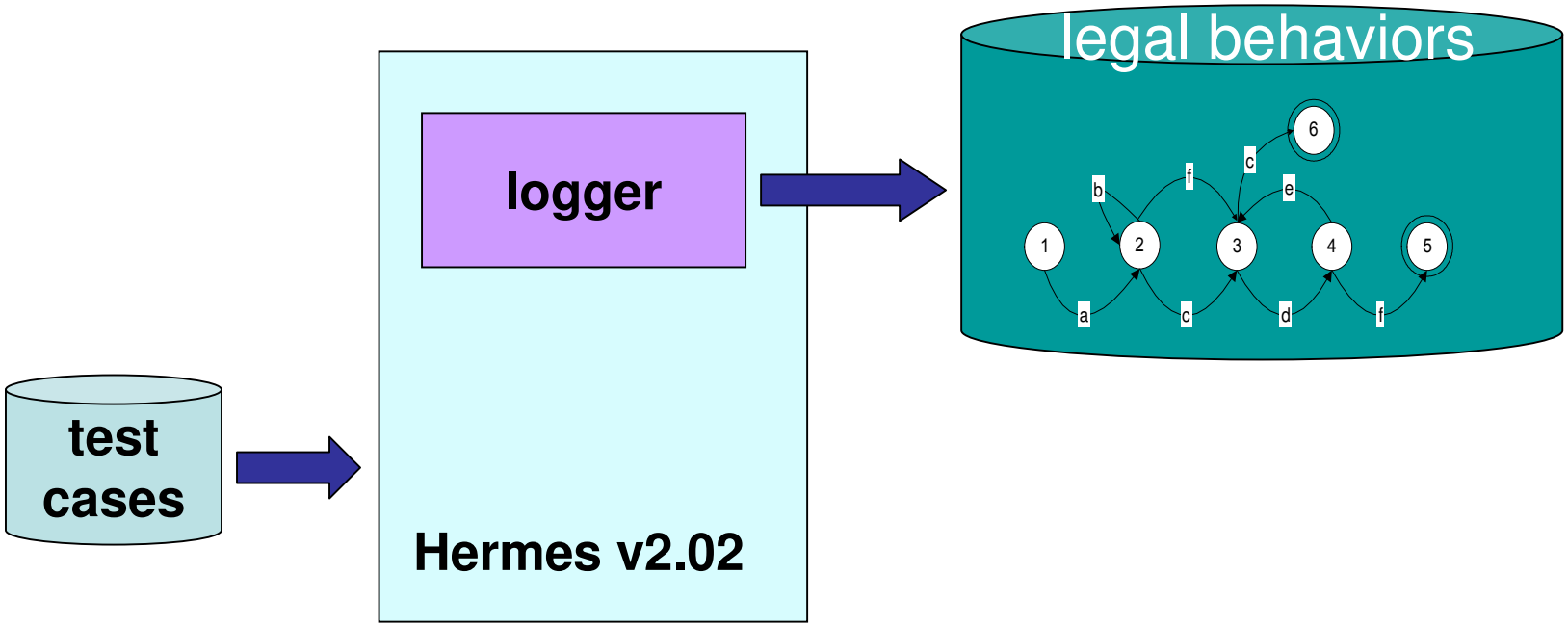
Computer Store: fault localization



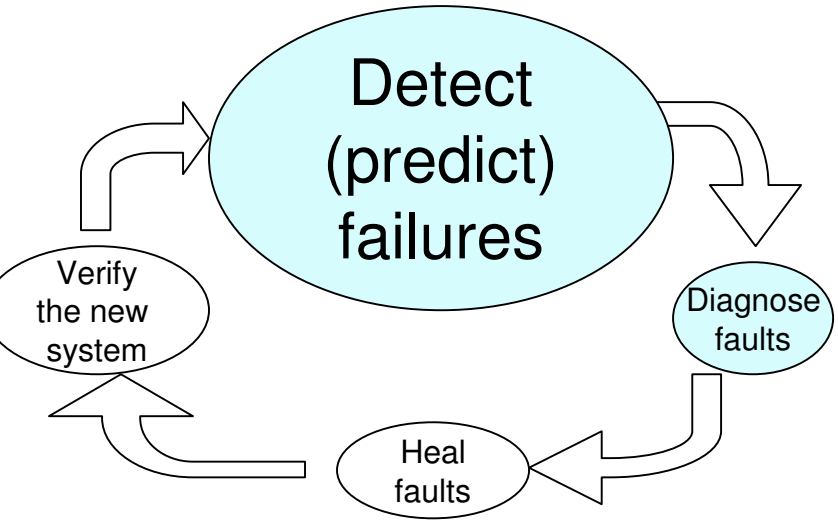
S = locate service
C = extract catalog
R= produce results

Hermes mobile middleware

generate information for fault diagnosis



Hermes



Failure condition: Close all threads

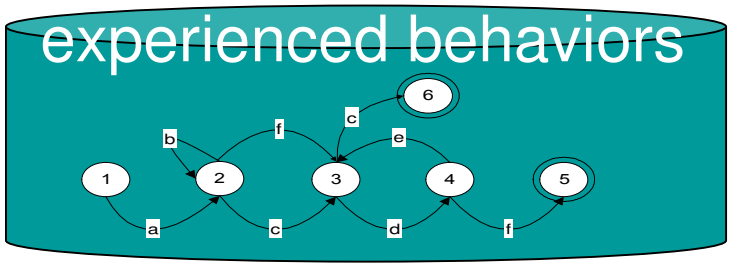
Detect failures

Open threads

inputs

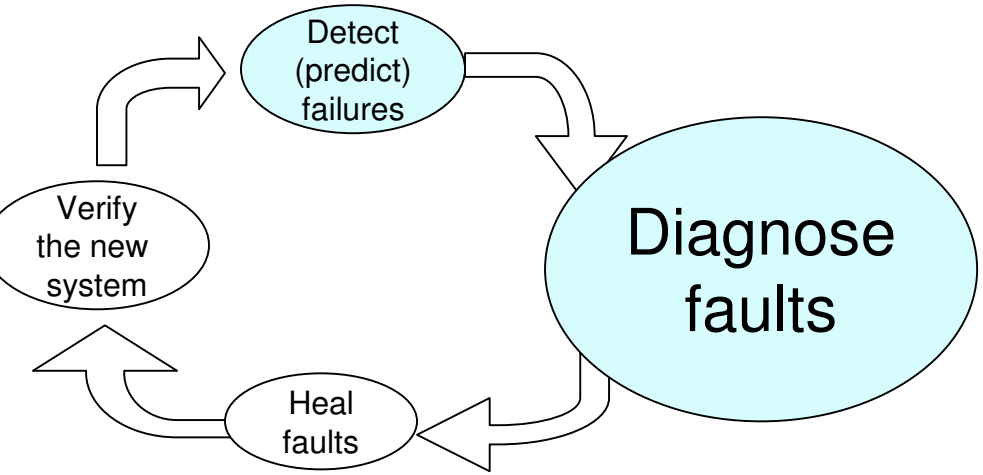
New logger

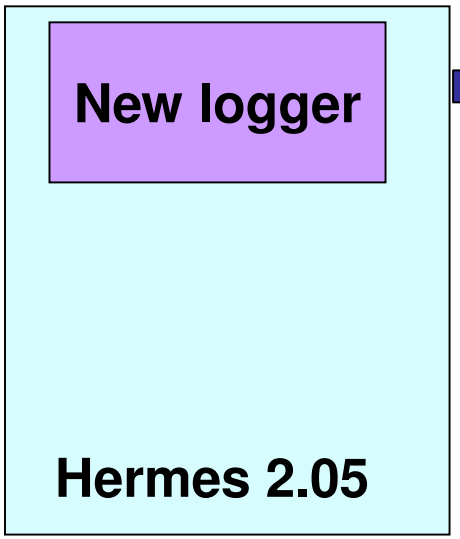
Hermes 2.05



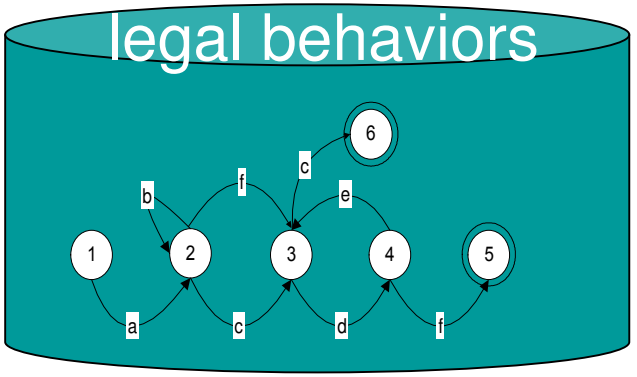
Record behaviors

Hermes

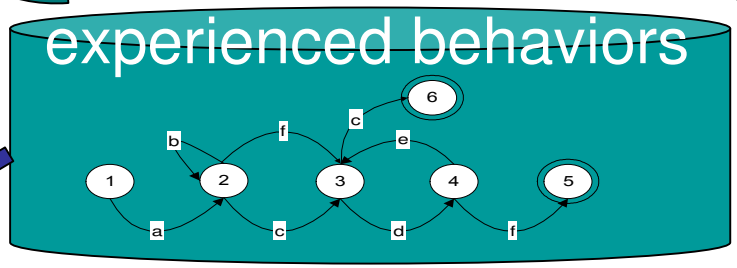




failure

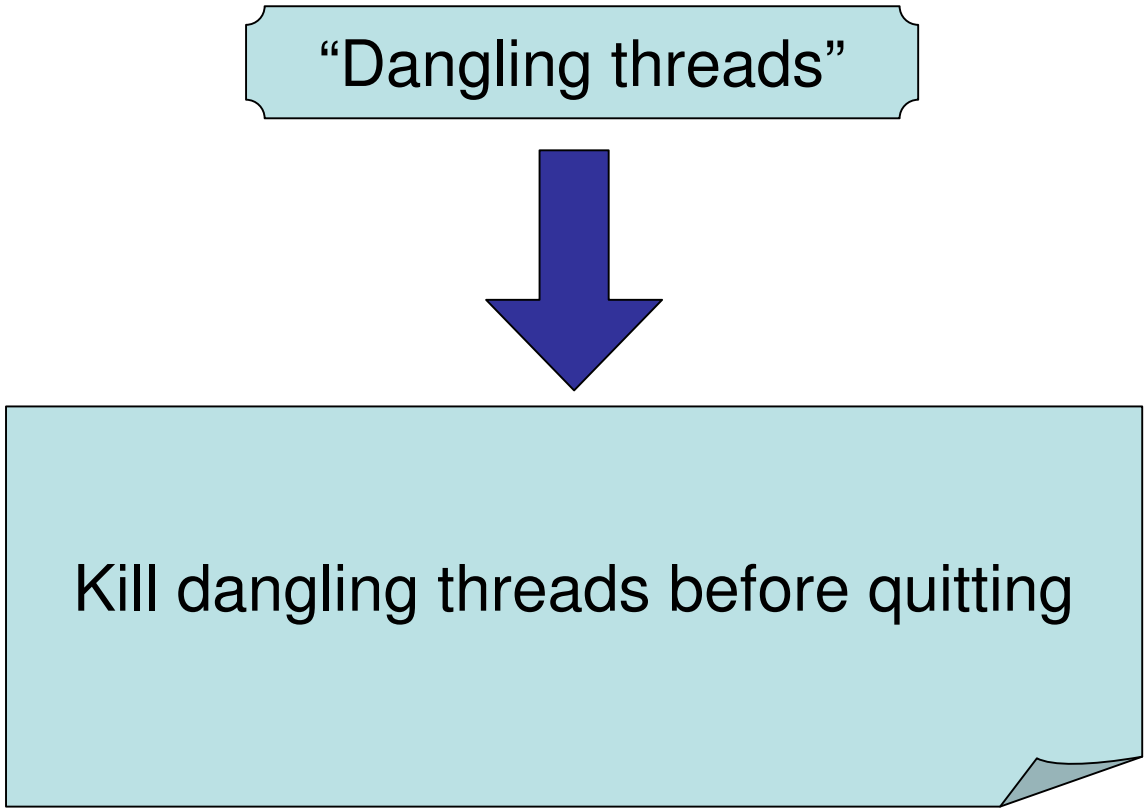


differences



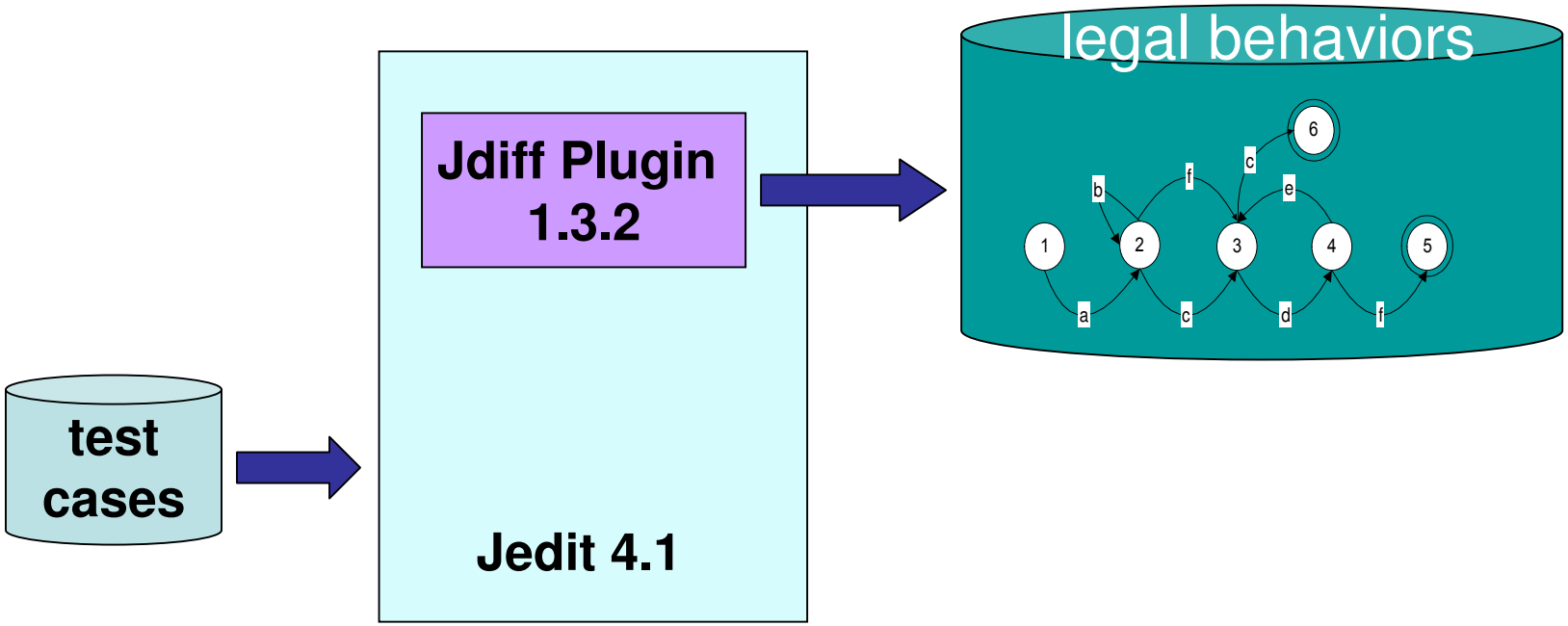
"dangling thread"

Hermes: fault localization

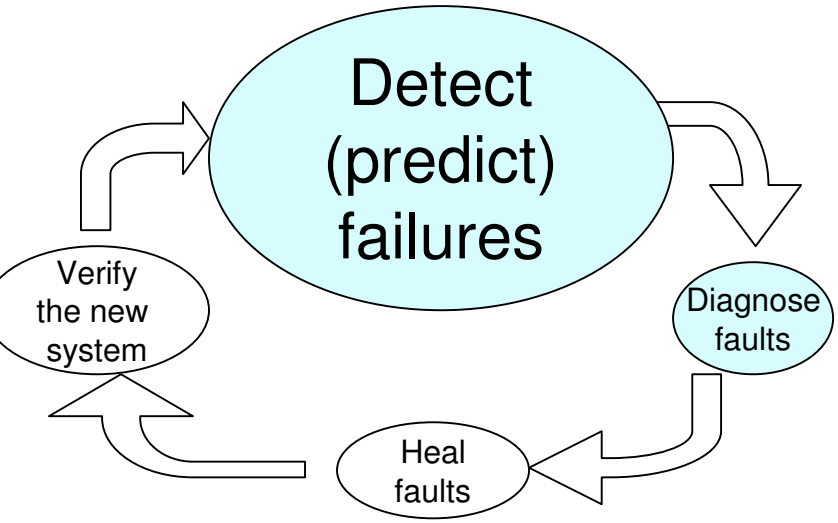


Jedit

generate information for fault diagnosis



Jedit



Failure condition:
Display all GUI elements

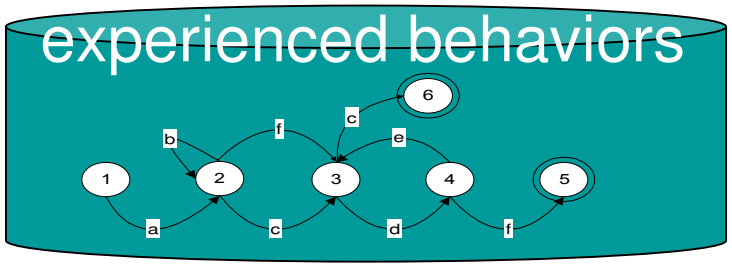
Detect failures

No scrollbar

inputs

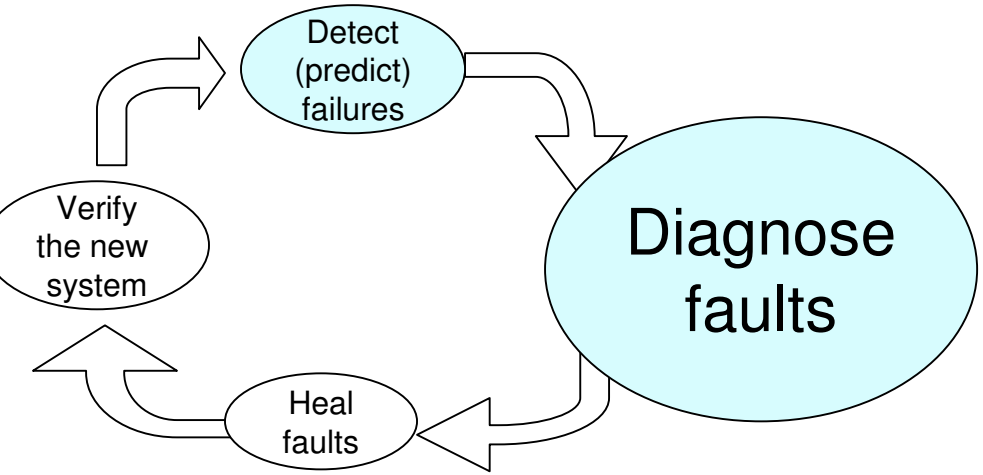
Jdiff Plugin 1.3.2

Jedit 4.2 Pre9



Record behaviors

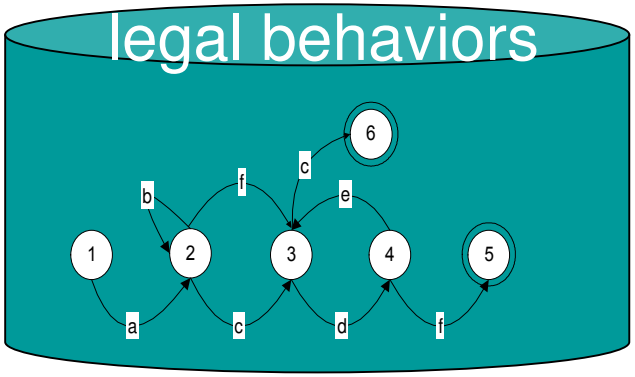
Jedit



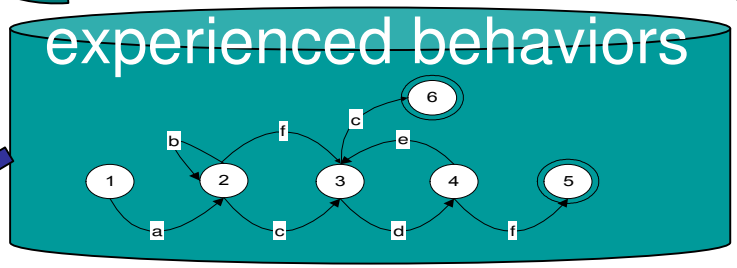
Jdiff Plugin 1.3.2

Jedit 4.2 Pre9

failure

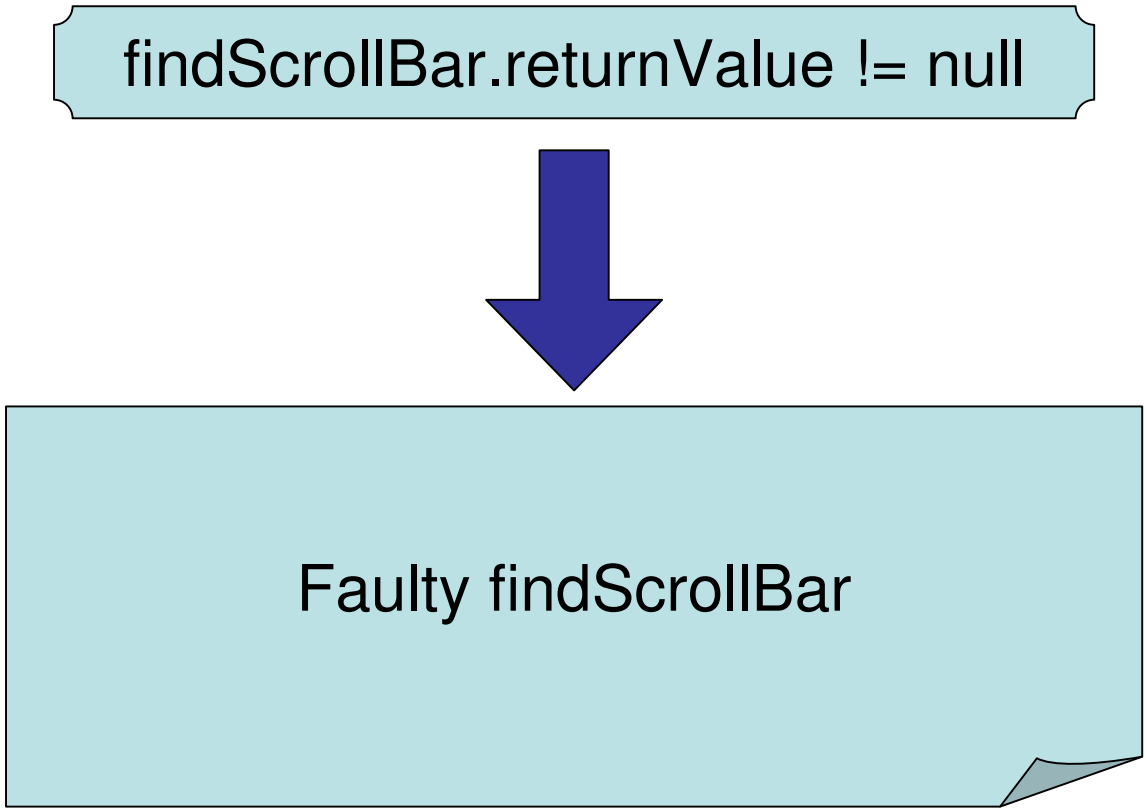


differences



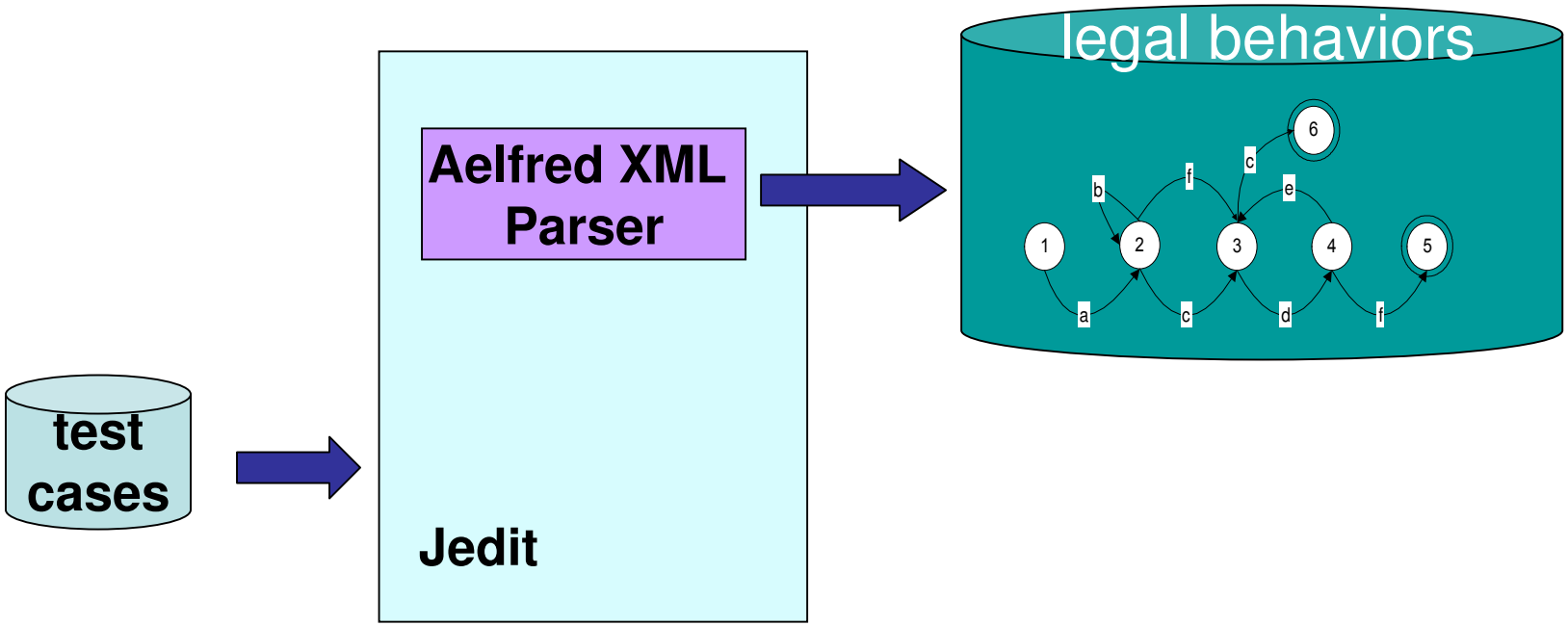
findScrollBar.returnValue != null

Jedit: fault localization

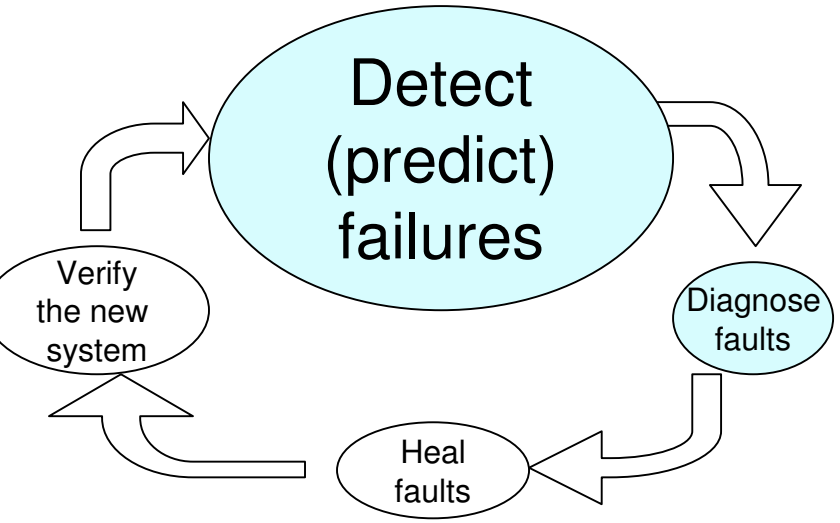


Aelfred XML Parser

generate information for fault diagnosis



Aelfred XML Parser



Failure condition: Unhandled exception

Unhandled exception

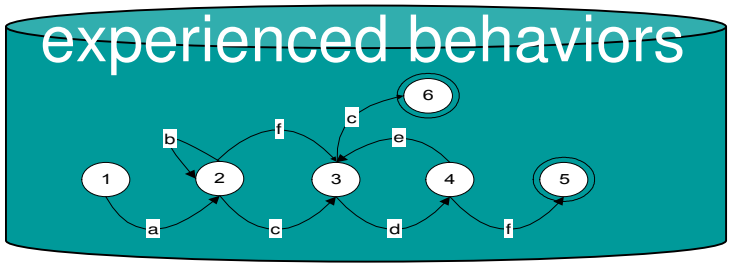
Detect failures

inputs

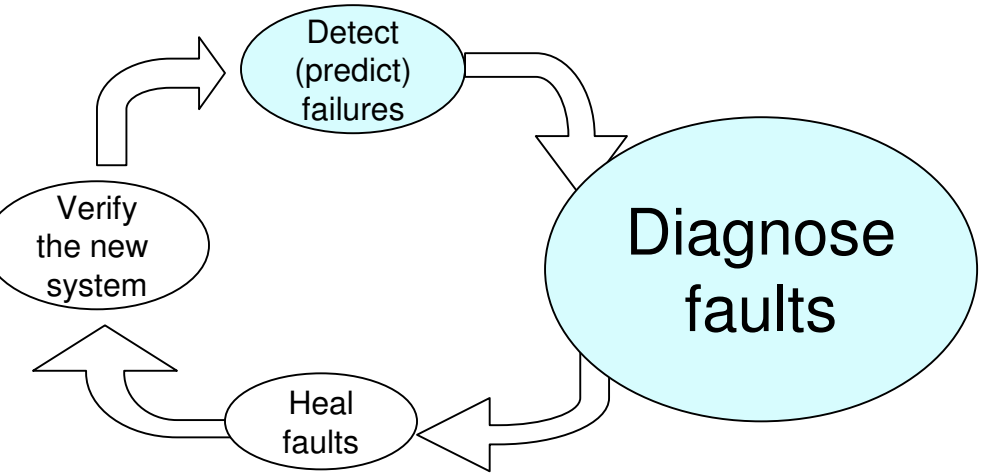
Aelfred XML Parser

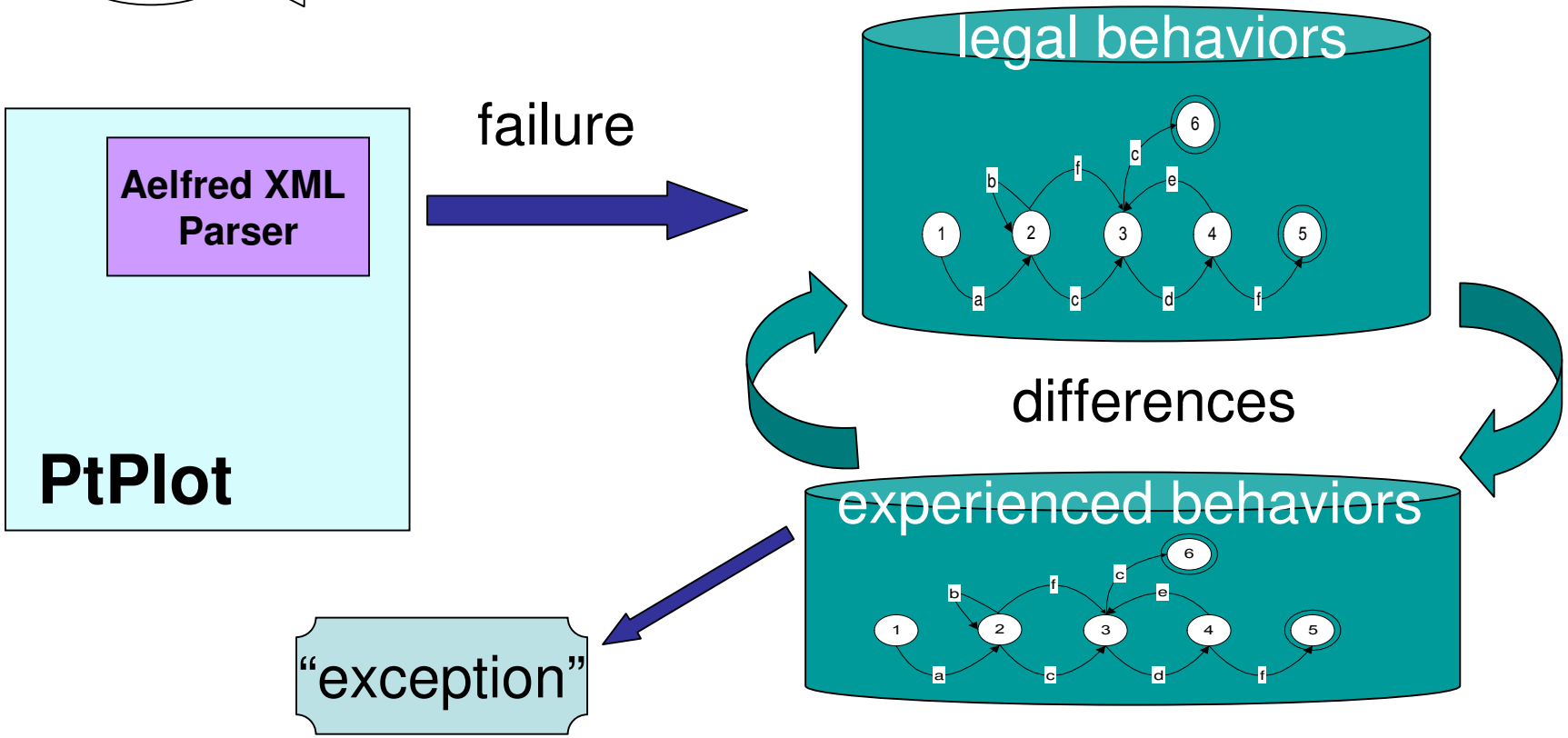
PtPlot

Record behaviors

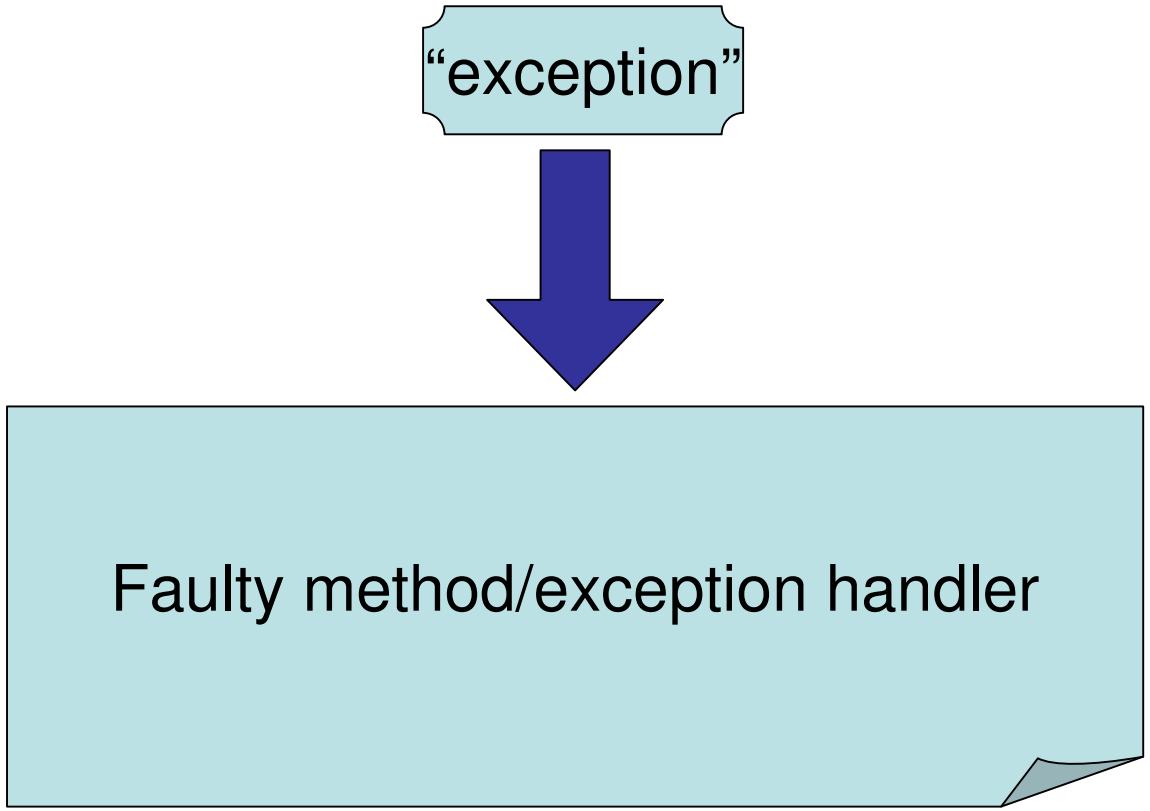


Aelfred XML Parser





Aelfred XML Parser



What's next

- Failure detection
  - What are the right models?
- Dynamic analysis support fault diagnosis
  - How can we generalize?
- fault taxonomies to identify fixing strategies
  - What are interesting fault taxonomies

Your input is greatly appreciated!