Larva : Safer Monitoring of Real-Time Java Programs

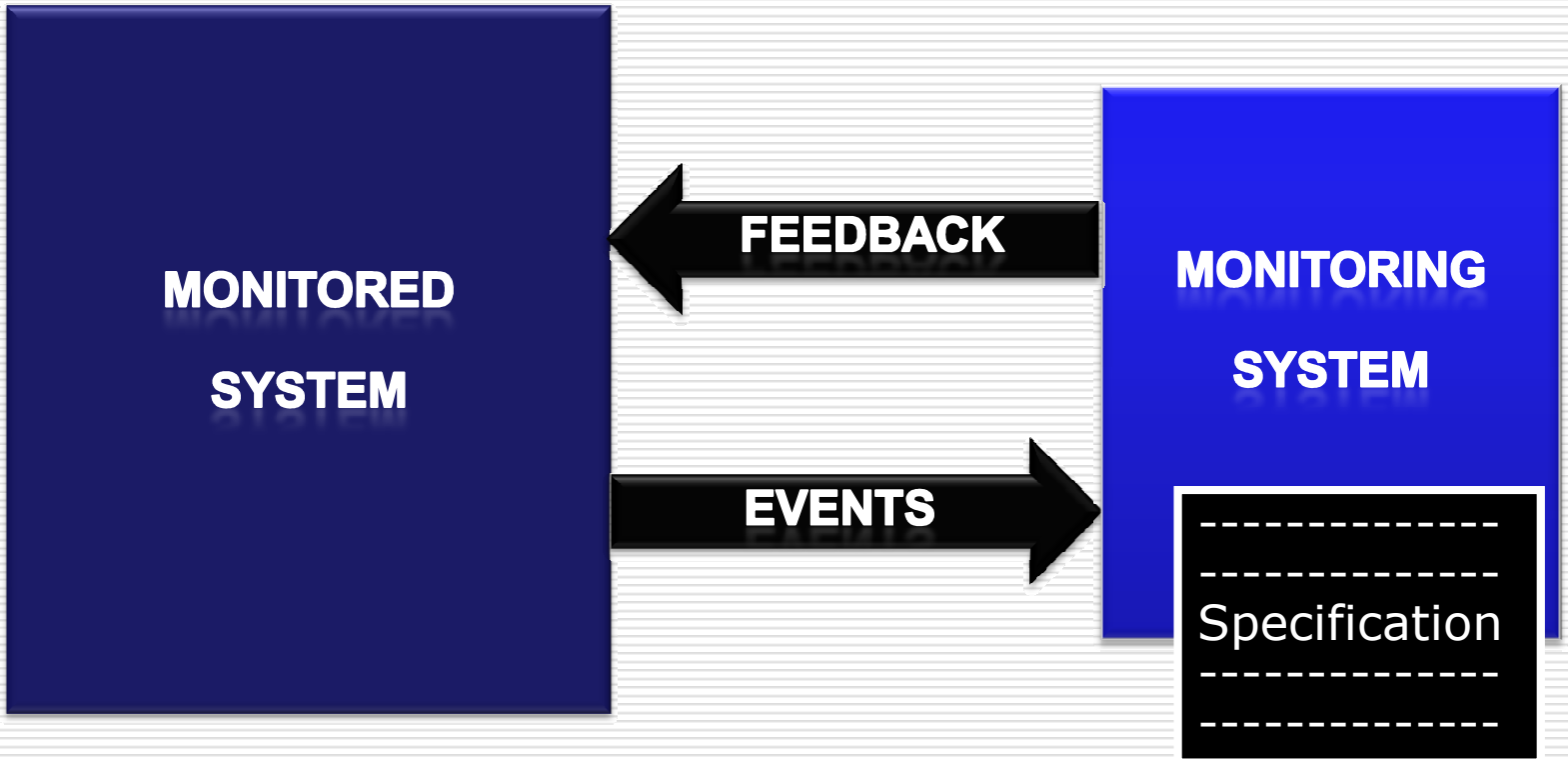
Christian Colombo
Dept. of Computer Science
University of Malta
Malta

Gordon Pace
Dept. of Computer Science
University of Malta
Malta

Gerardo Schneider
Dept. of Applied IT
Chalmers | Univ. of Gothenburg
Sweden
(Univ. of Oslo, Norway)

7th IEEE International Conference on Software Engineering and Formal Methods (SEFM'09)
Hanoi, Vietnam – 23-27 Nov 2009

Runtime Verification

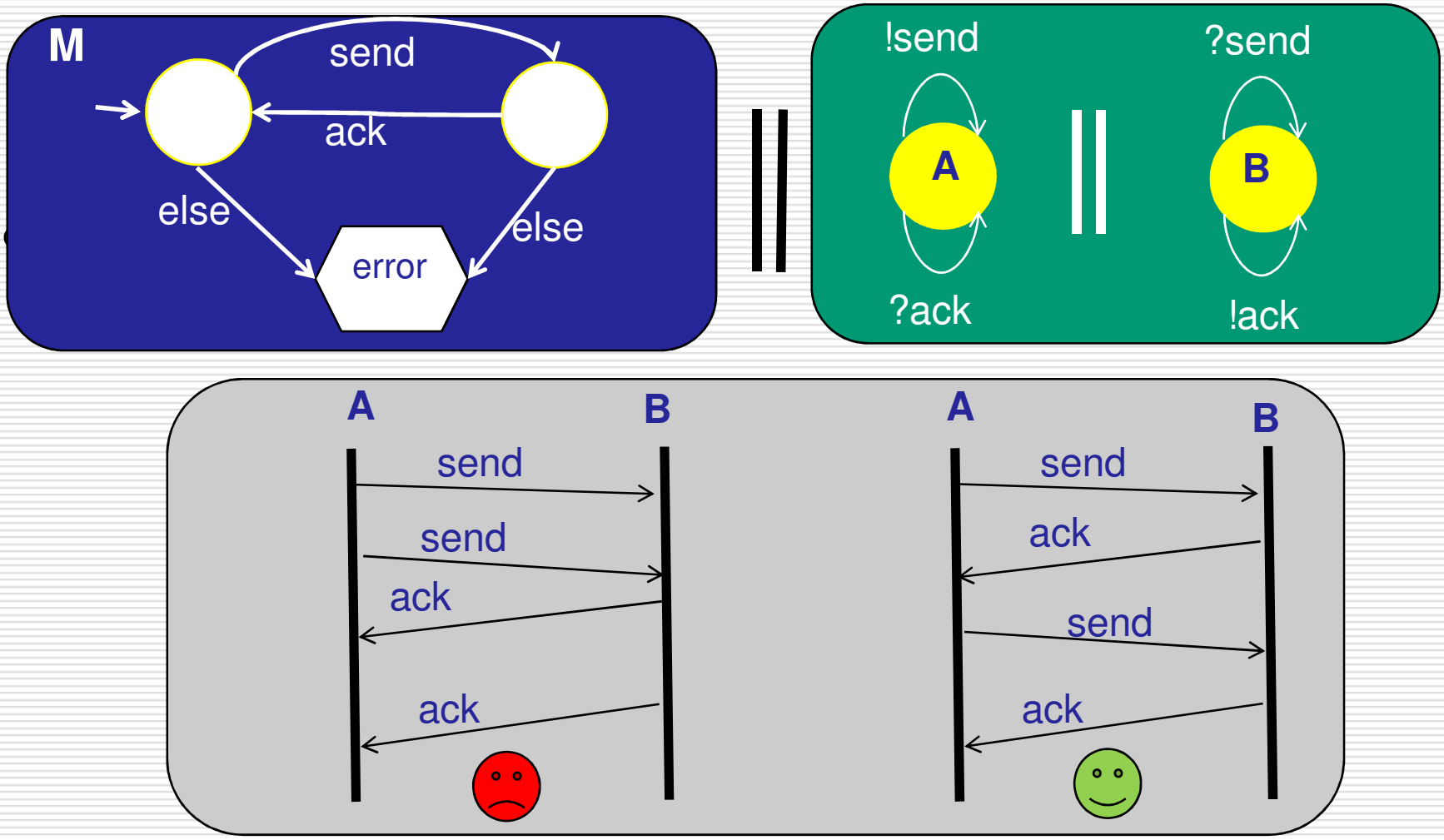


How to monitor a program

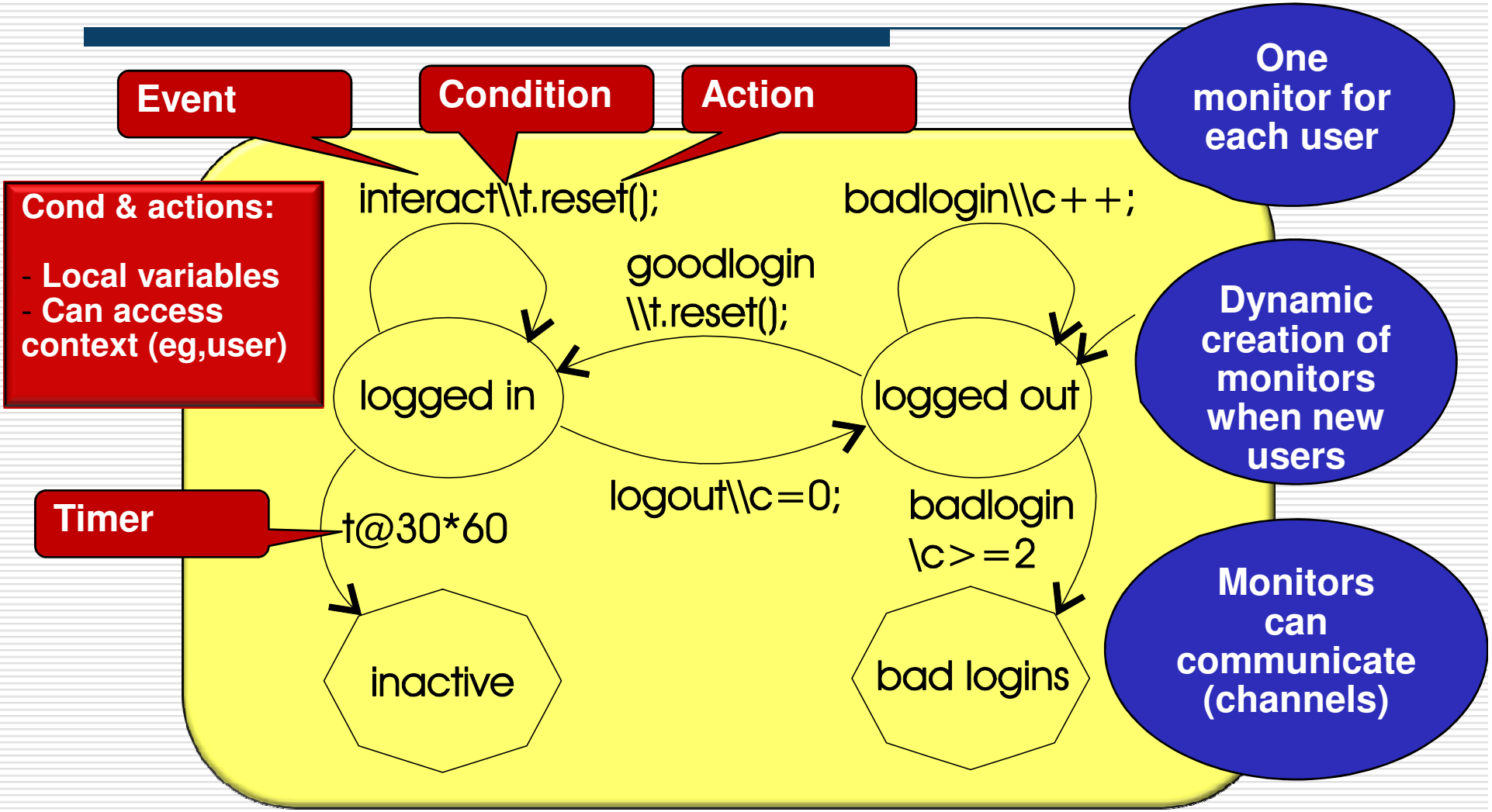
- Static phase:
  - Code instrumentation
- Dynamic phase:
  - Observing events from the program

- The monitor may be automatically extracted from the specification
  - Statically
  - Dynamically

Runtime Verification



A Scenario


Event
Condition
Action
One monitor for each user
Cond & actions:
- Local variables
- Can access context (eg,user)
interact\\t.reset();
badlogin\\c++;
goodlogin
\\t.reset();
logged in
logged out
Dynamic creation of monitors when new users
Timer
t@30*60
logout\\c=0;
badlogin
\c>=2
inactive
bad logins
Monitors can communicate (channels)

What Is the Message?

There is a need of a runtime verification tool that allows:

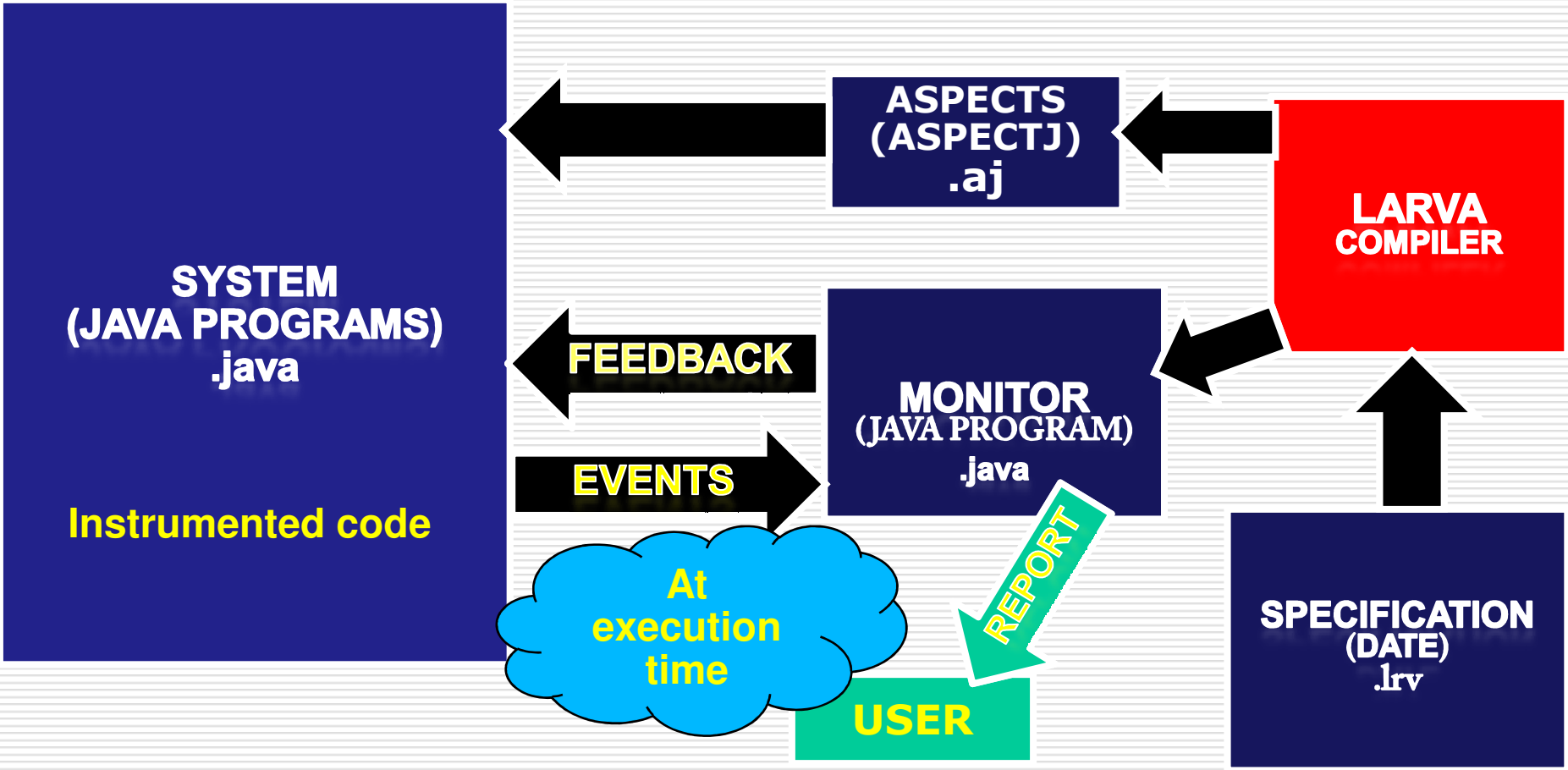
- The representation of contextual propeties
- The dynamic creation of monitors
- To monitor real-time properties
- The communication between different monitors

Runtime Verification of Real-Time Properties

Adding a monitor at runtime
slows down the system
and may invalidate certain properties
which would be valid otherwise

Eliminating a monitor at runtime
speeds up the system
and may invalidate certain properties
which would be valid otherwise

LARVA*



* **L**ogical **A**utomata for **R**untime **V**erification and **A**nalysis (http://www.cs.um.edu.mt/svrg/Tools/LARVA/)

DATE: Dynamic Automata with Timers & Events

Communicating symbolic automata enriched with **events** and **timers**

- Events: e.g. method calls and exception handling
- Timers can reset, pause and resume

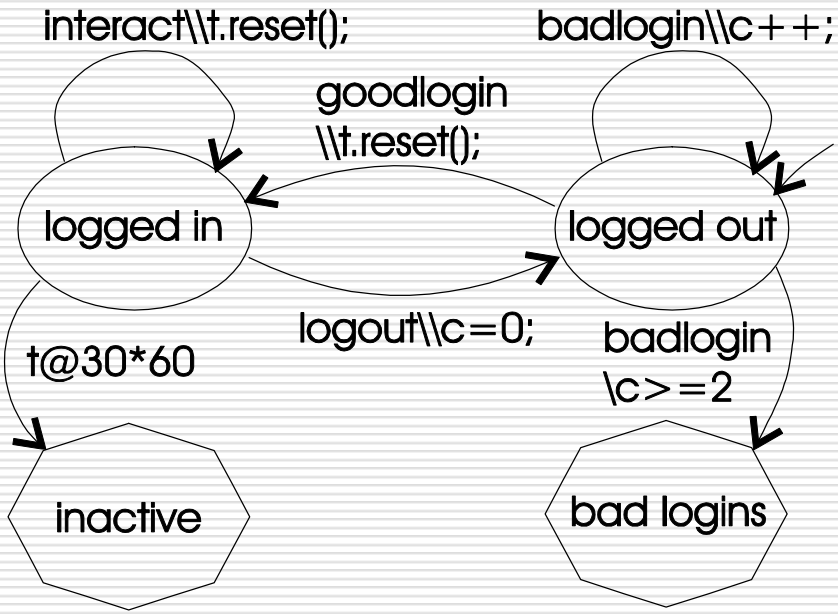
Automata are **dynamic:** automatically replicated according to **context**

- Context is taken into account

It supports: **Conditions** and **actions** on transitions
Communication between automata

- Actions can access code and data from the monitored program
- Communication is through channel synchronization

LARVA - Script

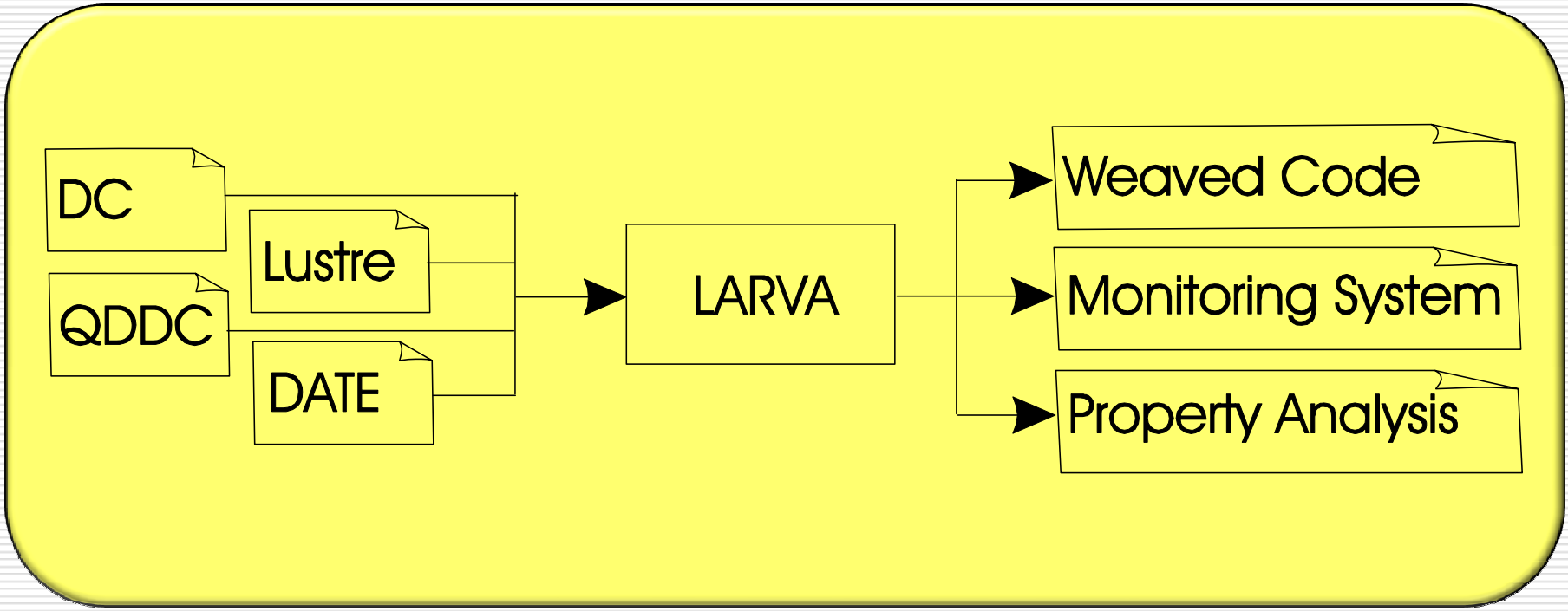



```
GLOBAL {
 VARIABL  ES {
  int c = 0;
  Clock t;     }

 EVENTS {
  badlogin() = {*.badlogin()}
  clk() = {t@30*60}
  ...        }

 PROPERTY users {
  STATES {
   BAD { badlogins inactive }
   NORMAL { loggedin }
   STARTING { loggedout }    }
  TRANSITIONS {
   loggedout -> badlogins [badlogin\c>=2\]
   loggedout -> loggedin  [goodlogin\\t.reset();]
   ...
   loggedout -> loggedout [badlogin\\c++;]
   loggedin -> inactive   [clk\\]     }  }
}
```

LARVA – Other input languages



Benchmark - Expressivity

Table 1. Expressivity features of various tools.

Tool	LARVA	ConSpec	Java-MOP	Java-MaC	Hawk	Lola
Scope	*Sess./Obj.*	✓ [a]	*Sess./Obj.*	*Sess.*	*Sess.*	*Sess.*
Exceptions	✓	✓	×	×	×	×
Temporal Logics	×	×	✓	×	✓	✓
Real-Time	✓	×	×	✓ [b]	✓ [c]	×
Mobile Application Policies	×	✓	×	×	×	×
Invariants	✓	×	✓	✓	×	×
Feedback	✓	*Stop.*	✓	✓	×	×
Conditions	✓	✓	✓ [d]	✓	×	×
Numerical Queries	×	×	×	×	×	✓

[a] in specification it supports all the mentioned scopes but currently only *session* is supported

[b] restricted (cannot trigger clock events)

[c] can be extended to support real-time

[d] restricted to implementing conditions in violation/validation handling method

What Is Behind the Stage?

Definition of suitable automata for RV with real-time (DATE)

A sound translation from Phase Automata into DATEs

- There exists a translation from DC into Phase Automata (characterize "implementable" DC) ([Bouajjani et al.95], [Hoenicke06])

-Characterization of slow-down and speed-up real-time formulae

- Formal definition and theoretical results on time transformation (instantiated into the Duration Calculus)

Translation of Lustre and QDDC into DATEs

Real Case Studies

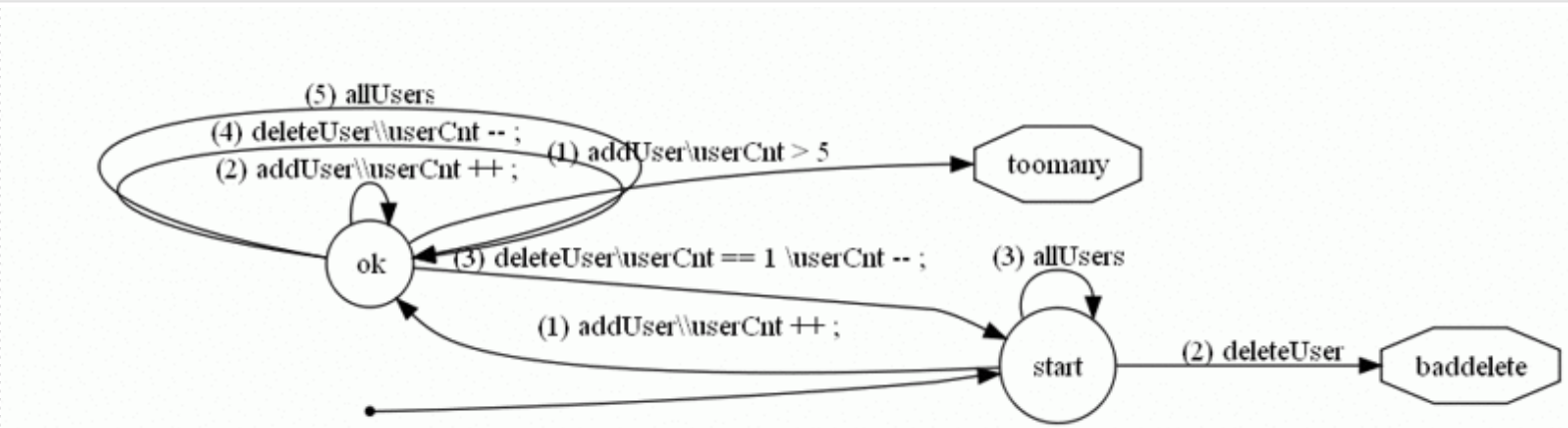
- Applications at Ixaris Systems Ltd.
  - Operates EntroPay, an online prepaid payment service
  - Services: E.g., virtual credit cards
- Runtime verification of
  - Contextual properties of objects (dynamically created)
  - Real-time: Expiration dates
  - Security: Rights, authorization, data integrity

Recent Developments:

- An extension to Larva called **LarvaStat** which is able to handle statistical properties
- An adapter connecting Larva to an SQL database, extracting events and allowing Larva to monitor offline

Demo

- A simple bank system
  - General bank system, users, accounts, transactions
- Property
  - No more than 5 users logged in at the same time
  - No more than 5 accounts per user
  - No more than 5 transactions at the same time



Conclusions

RV of real-time properties should be taken carefully

- Monitors affect behavior

We have a theory of slow-down and speed-up truth preserving properties (characterization in DC)

- We know which properties to monitor at runtime and which during testing

A tool for RV of Java – LARVA

- Mathematical framework – DATE
- Highly expressive (contextual and real-time properties)

Application to real case studies

Thank you very much for your atention!

Questions?

LARVA – Compilation into Java

- AOP to capture events
- A hierarchy of classes
  - One for each context
- Each class has a reference to its parent context (E.g. The account context, have access to the user context)
- A hashmap to keep track of the distinct objects which we are checking

How to monitor a program

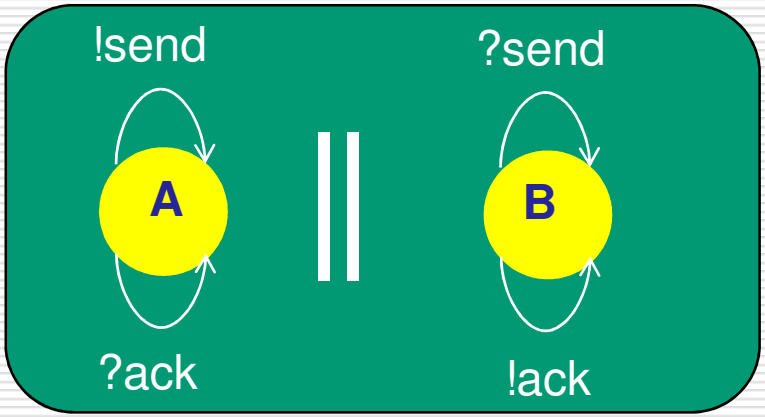
- Static phase:
  - Code instrumentation
- Dynamic phase:
  - Observing events from the program

- The monitor may be automatically extracted from the specification
  - Statically
  - Dynamically

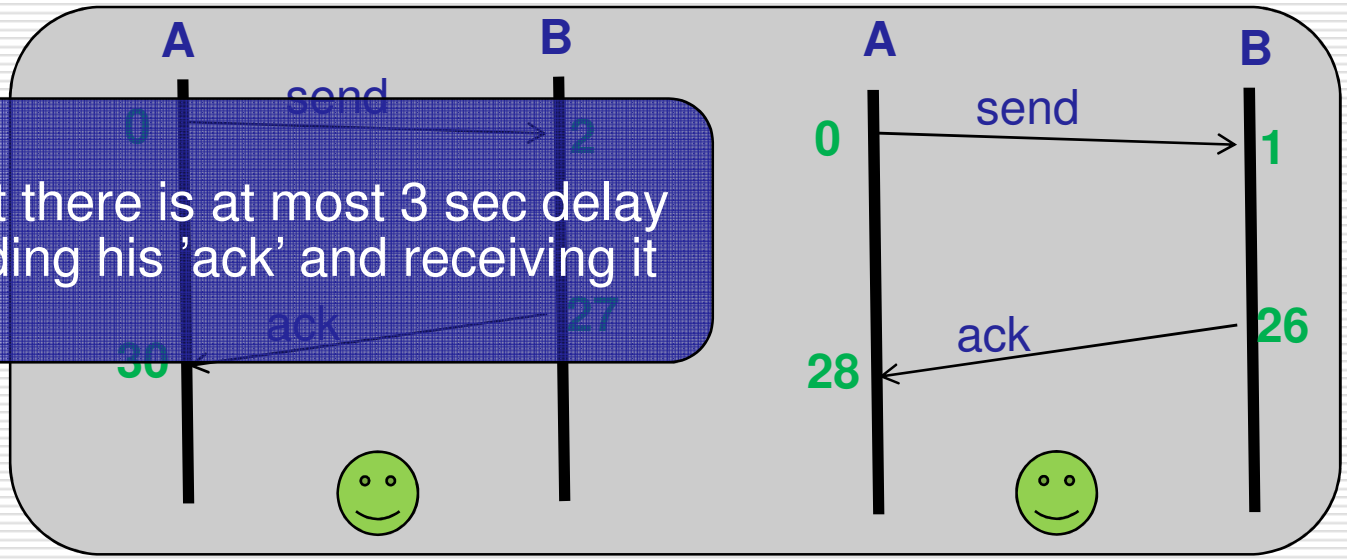
Runtime Verification of Real-Time Properties

A 'send' should only be followed by an 'ack'

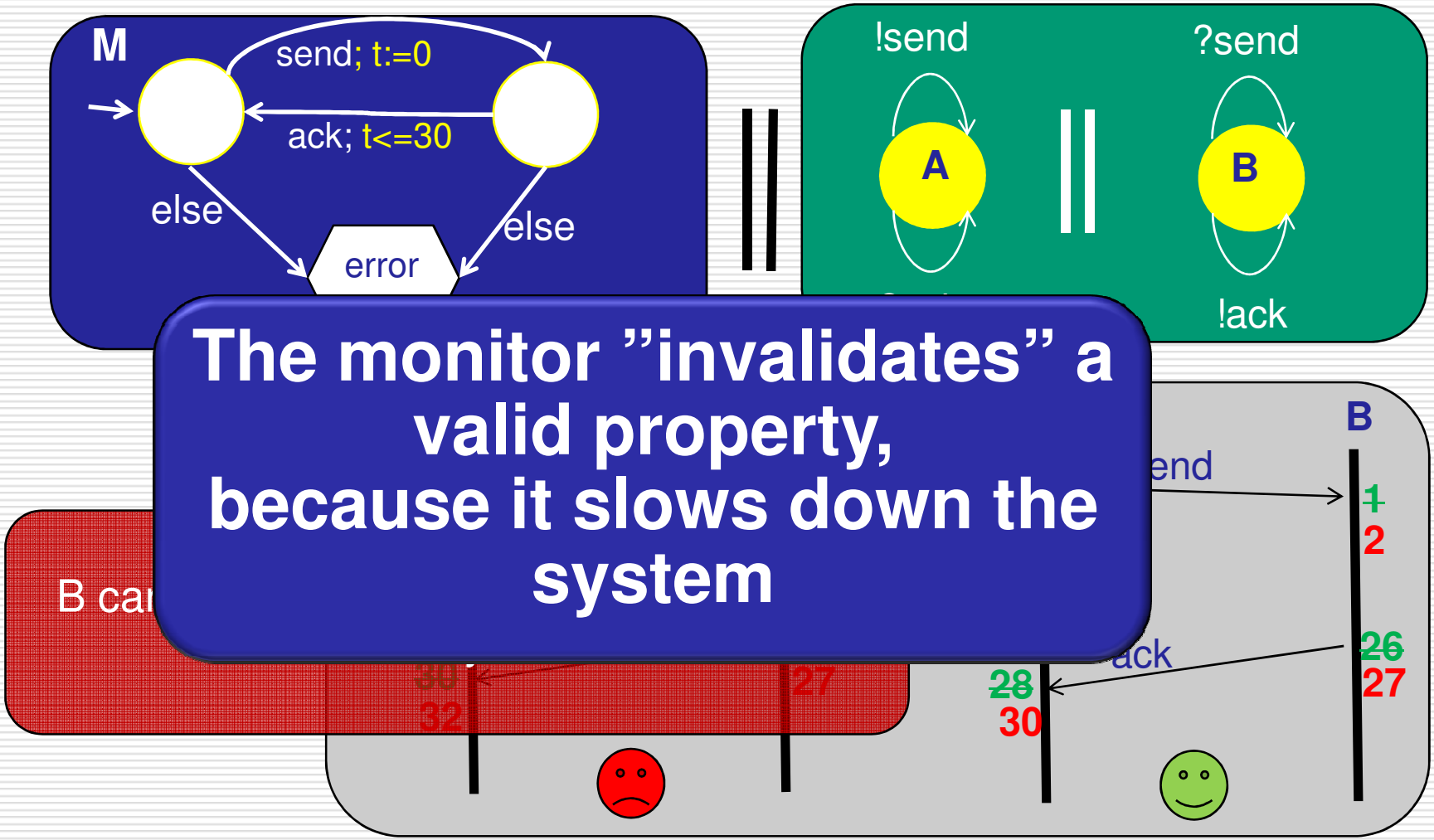
Any 'send' must be followed by an 'ack' within 30 sec



B "knows" that there is at most 3 sec delay between sending his 'ack' and receiving it



Runtime Verification of Real-Time Properties



The monitor "invalidates" a valid property, because it slows down the system

Duration Calculus

DC is an interval logic: It expresses properties which hold over time intervals

- It formalizes the concept of duration

A duration calculus formula is said to hold with respect to a particular interpretation

- An interpretation is the evaluation of a set of variables with respect to time

The calculus is based on two main concepts

- Integration: it measures the duration of a Boolean variable over an interval
- The chop operator: which splits an interval into two with different formulae which must hold on each subinterval

Lustre

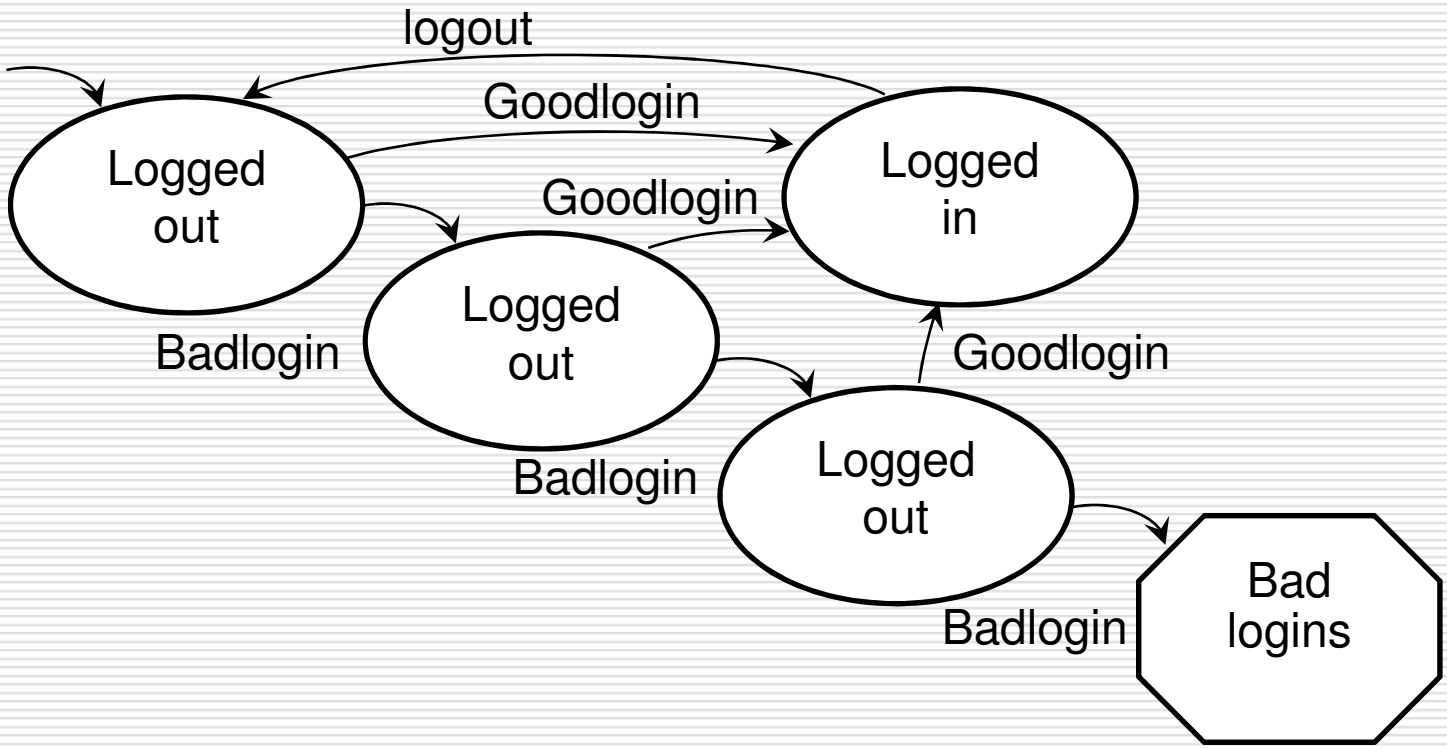
A slide on LUSTRE Here!!

QDDC

A slide on QDDC Here!!

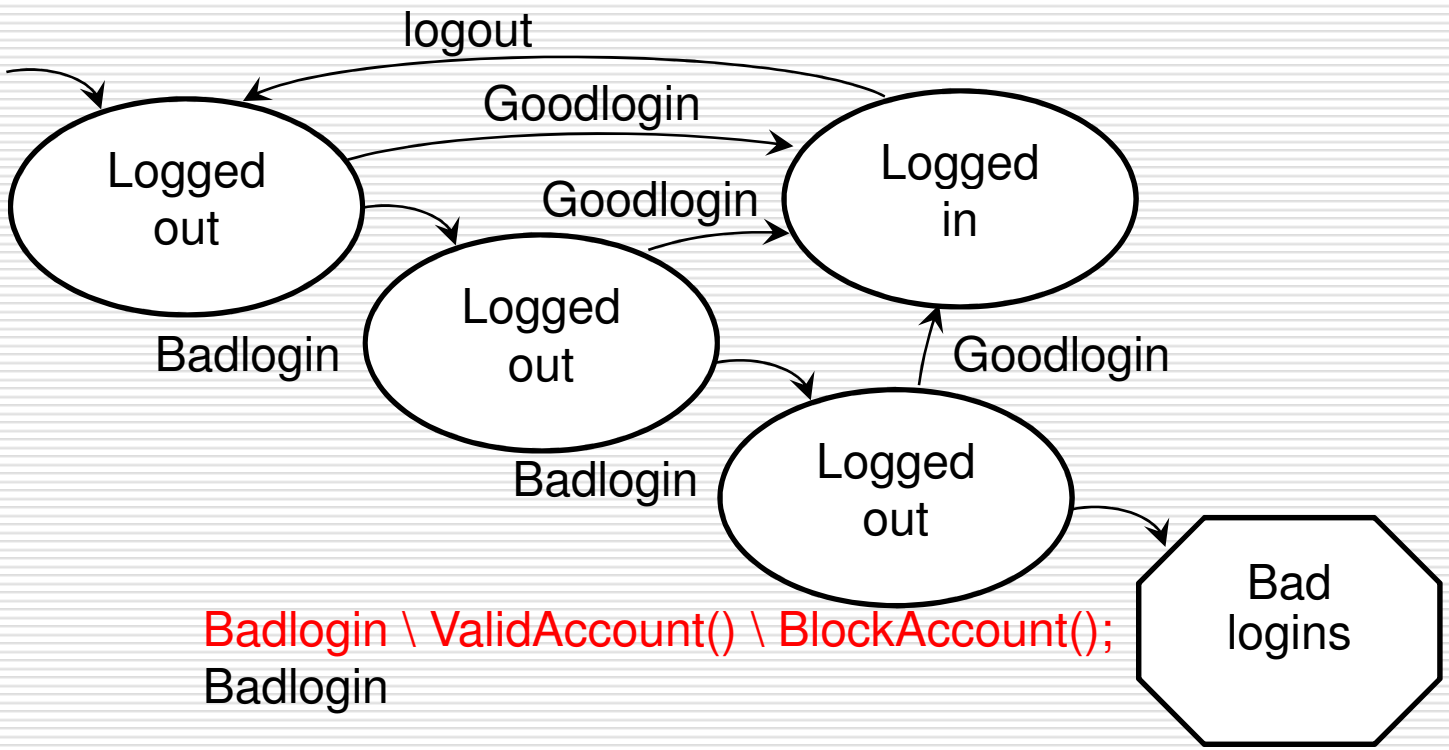
A Scenario - Events

No 3 successive bad logins



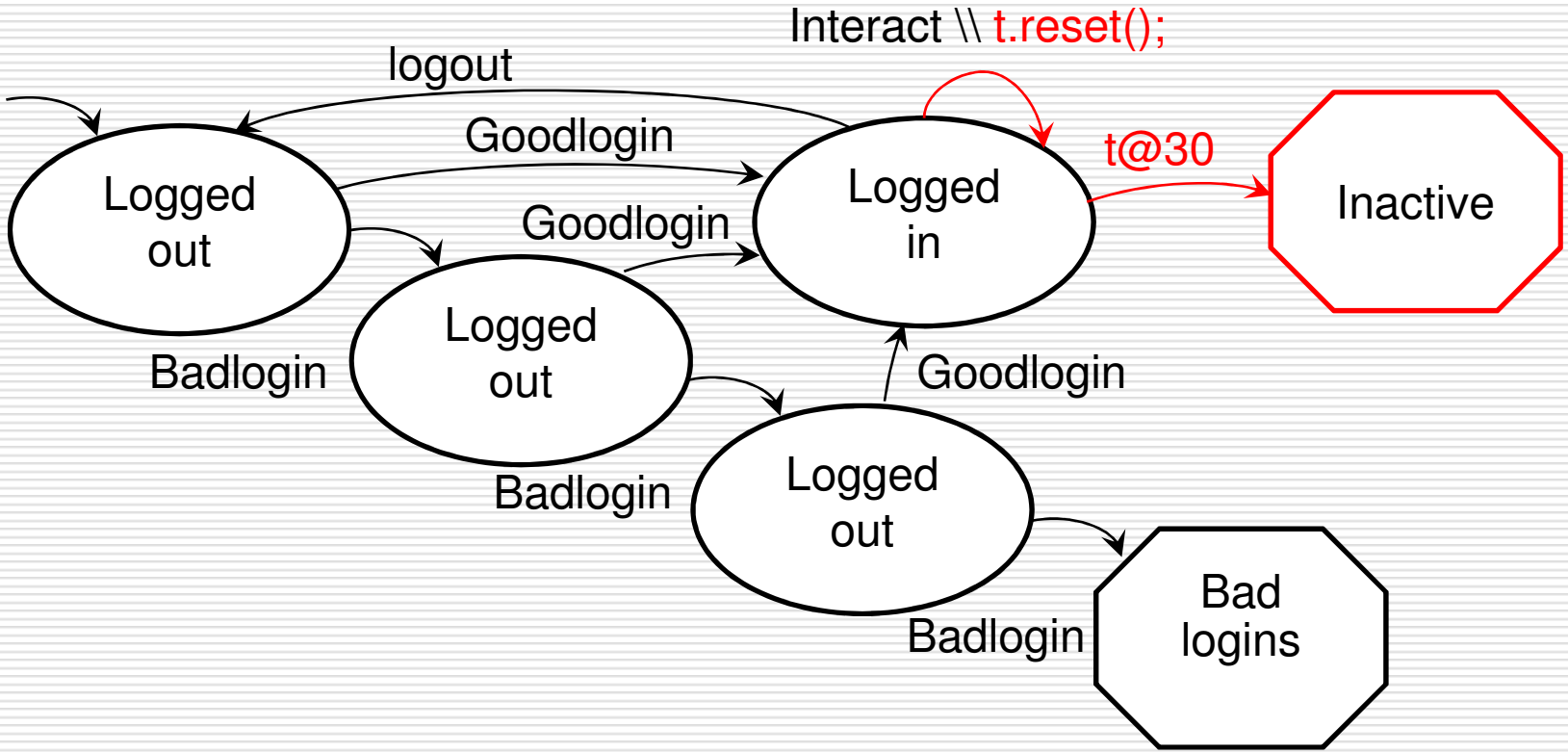
A Scenario – Conditions & Actions

No 3 successive bad logins

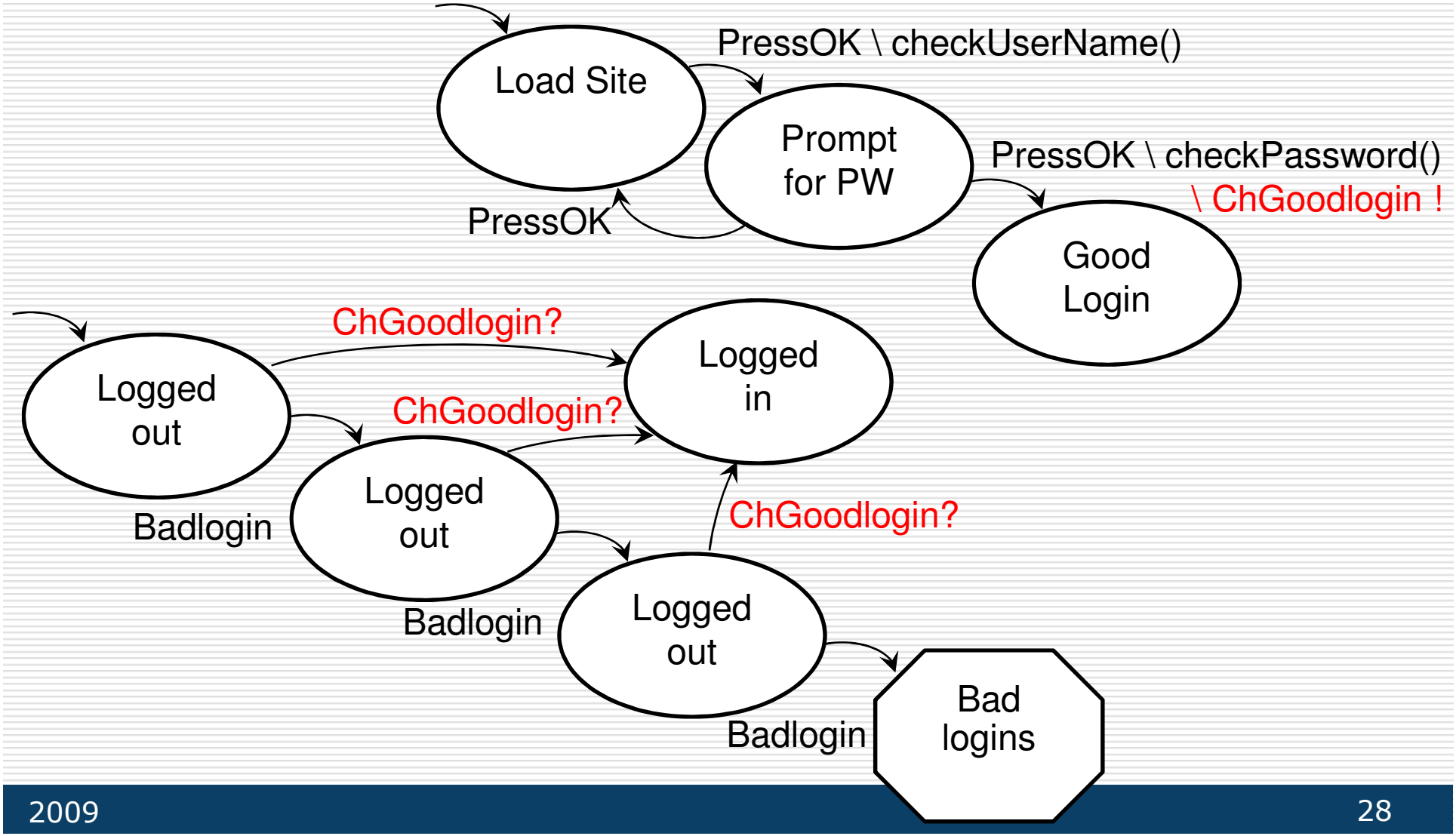


Badlogin \ ValidAccount() \ BlockAccount();
Badlogin

A Scenario - Clocks



A Scenario – Channels



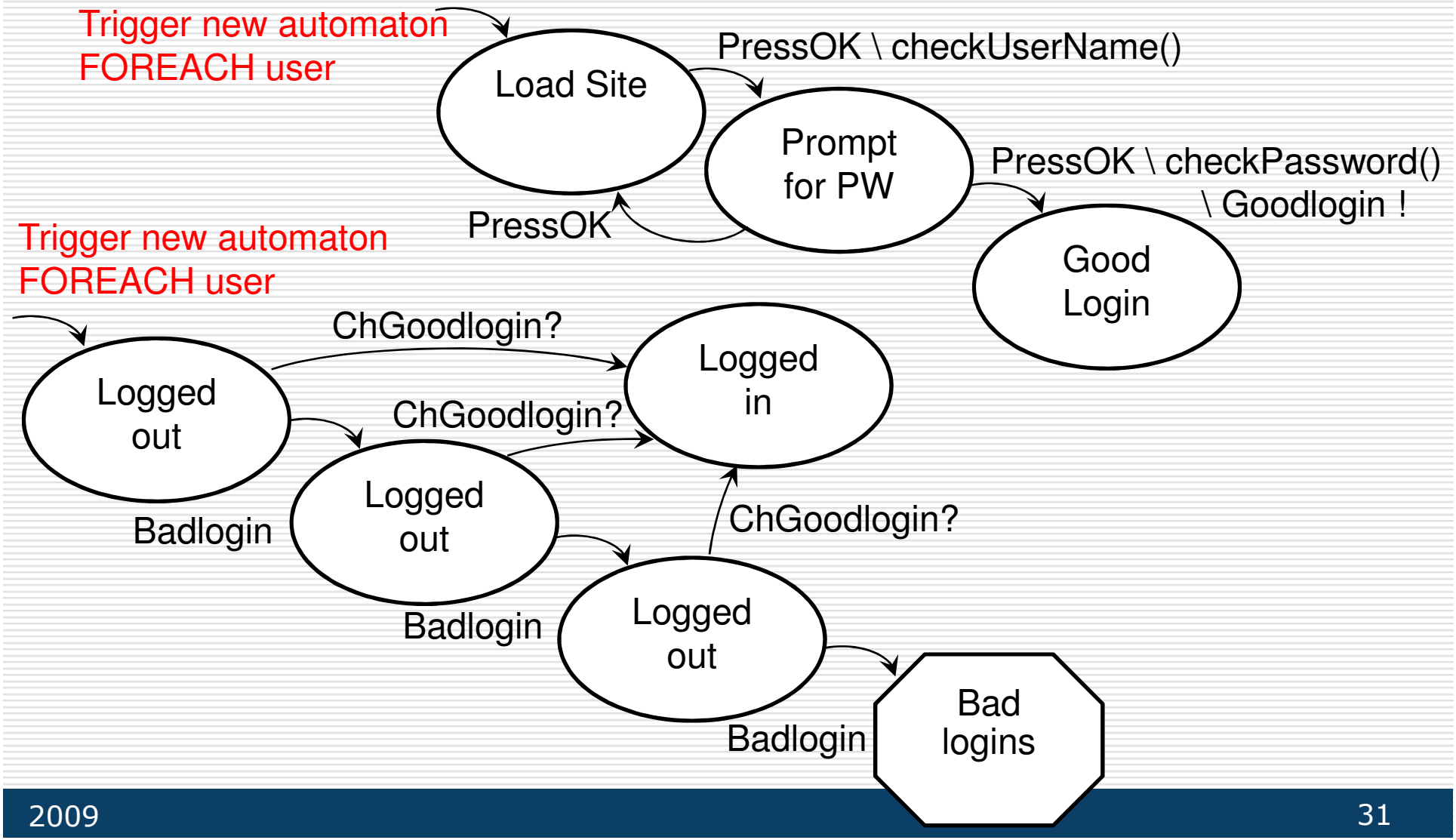
A Scenario – Dynamic Triggers

- Imagine we need to check login/logout for each user.
- We have to **trigger** an automaton for every user, to keep track whether each user is logged in or not.
- Use method parameters to get **context**.

Specifying Context

- Actions and conditions on transitions can access the context (User).
- A context can be nested to have a more specific context within it:
  - Eg: Check login for each **site** of each individual **user**.

A Scenario – Context



Benchmark – Performance

- Dummy transaction processing system (4 properties – 2 real-time)
- Memory and time required is considerable but linear to the number of objects being monitored (replication of automata).
- Compares well with Java-MOP which is the most similar work available for usage.