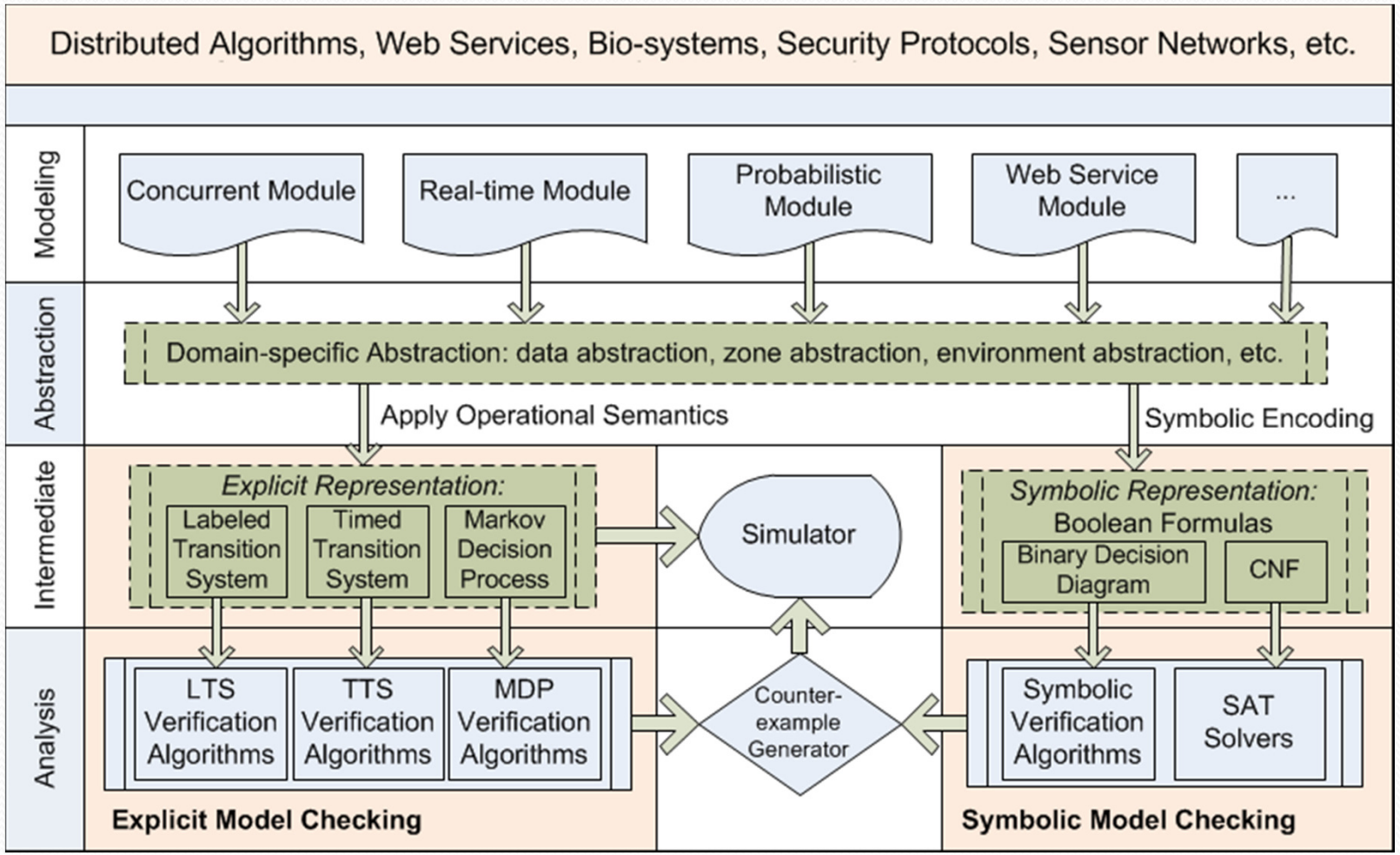
Event Analytics and Verification: PAT Approach

Jin-Song Dong (first met Ed at Marktoberdorf 1994)
National University of Singapore (NUS)
PAT team: two former PhD students: prof. **J. Sun** (SUTD), prof. **Y. Liu** (NTU) and 20+ PhD/Postdocs

PAT System - based on Hoare's ***event*** based formalism CSP

(CAV'09'12'13'14, FM'11'12'14, TOSEM'13'14, TSE'13'14, FMSD'14)



Support event based formalisms:

CSP# with shared variables, Timed-CSP, Probabilistic-CSP …

Model checking as planning/scheduling [FMSD'14]

```
//@@Sliding Game@@
//The following models the sliding game with the extra 'costs' complexity

var board[9]:{0..8} = [3,5,6,  // 0,1,2 :index
                       0,2,7,  // 3 4 5 :index
                       8,4,1]; // 6,7,8 :index
hvar empty:{0..8} = 3; //empty position is a secondary variable, no need to put it in the state space

var c = 0; // cost utility, e.g. costs 1 for left and right move, 2 for up, 0 for down

Game() = Left() [] Right() [] Up() [] Down();

Left() = [empty!=2 && empty!=5 && empty!=8] left                          // c=c+1
         {board[empty]=board[empty+1]; board[empty+1]=0; empty=empty+1; c++} -> Game();

Right() = [empty!=0 && empty!=3 && empty!=6] right
          {board[empty]=board[empty-1]; board[empty-1]=0; empty=empty-1; c++} -> Game();

Up() = [empty!=6&&empty!=7&&empty!=8] up
       {board[empty]=board[empty+3]; board[empty+3]=0; empty=empty+3; c=c+2} -> Game();

Down() = [empty!=0&&empty!=1&&empty!=2] down
         {board[empty]=board[empty-3]; board[empty-3]=0; empty=empty-3} -> Game();


#define goal board[0] == 1 && board[1] == 2 && board[2] == 3 &&
             board[3] == 4 && board[4] == 5 && board[5] == 6 &&
             board[6] == 7 && board[7] == 8 && board[8] == 0;

#assert Game() reaches goal with min(c);
```

3	5	6
	2	7
8	4	1

1	2	3
4	5	6
7	8	

Strategy Analytics (MDP-based)
How can Federer beat Nadal ?

- Federer vs Nadal is one of the greatest rivalries in tennis history.
- While Federer is widely regarded the best tennis player in history with 17 G.Slam (13 for Nadal) and 300+ weeks #1 (100+ for Nadal) and 6 ATP Year End Final Titles (0 for Nadal).
- However Nadal's record against Federer is 23-10 in Nadal's favour.
- Why? How can Federer beat Nadal

Year	Australian Open	French Open	Wimbledon	US Open
2003	Andre Agassi	Juan Carlos Ferrero	Roger Federer	Andy Roddick
2004	Roger Federer	Gastón Gaudio	Roger Federer	Roger Federer
2005	Marat Safin	Rafael Nadal	Roger Federer	Roger Federer
2006	Roger Federer	Rafael Nadal	Roger Federer	Roger Federer
2007	Roger Federer	Rafael Nadal	Roger Federer	Roger Federer
2008	Novak Djokovic	Rafael Nadal	Rafael Nadal	Roger Federer
2009	Rafael Nadal	Roger Federer	Roger Federer	Juan Martín del Potro
2010	Roger Federer	Rafael Nadal	Rafael Nadal	Rafael Nadal
2011	Novak Djokovic	Rafael Nadal	Novak Djokovic	Novak Djokovic
2012	Novak Djokovic	Rafael Nadal	Roger Federer	Andy Murray
2013	Novak Djokovic	Rafael Nadal	Andy Murray	Rafael Nadal

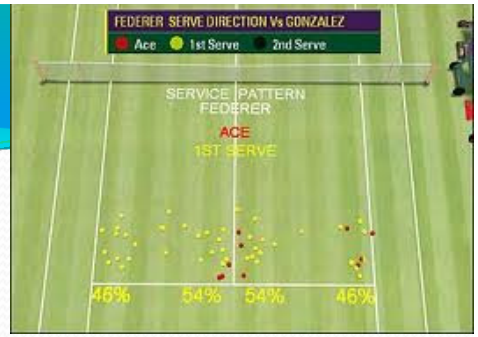
Building a PAT model

```
    Federer
 de_ct   ad_ct
 ------+------  baseline
 |  1  |  2  |
 |-----|-----|  service line
 |  3  |  4  |
 |===========|  net
 |  5  |  6  |
 |-----|-----|  service line
 |  7  |  8  |
 ------+------  baseline
 ad_ct   de_ct
```

```
    Nadal
"9" represents net error or
hit outside
```

```
enum{f_ad_ct, n_ad_ct, f_de_ct, n_de_ct};
//serve position: ad court or deuce court

enum{federer, nadal, na};

var turn = na; //serve turn;
var fscore = 0;
var nscore = 0;
var won = na;
var ball = 9;

WhoServe1st = []i:{f_de_ct,n_de_ct}@
        TossCoin{turn = i} -> Skip;

TieBreakGame = WhoServe1st;
        (FedererServe [] NadalServe);
```

Probability distribution on Federer Serve




```
De_Fed1stServe = pcase { // all probability is based on percent %, 30 means 30%
        23: ServeT_in{ball= 6} -> NadForehandR  // Nadal is lefty.
        15: ServeT_err{ball=9} -> De_Fed2ndServe
        30: ServeWide_in{ball =6} -> NadBackhandR
        11: ServeWide_err{ball=9} -> De_Fed2ndServe
        14: ServeBody_in{ball=6} -> (NadBackhandR [] NadForehandR)
         7: ServeBody_err{ball=9} -> De_Fed2ndServe};
De_Fed2ndServe = pcase {                    //1st serve is out
        15: ServeT_in{ball= 6} -> NadForehandR
         3: ServeT_err{ball=9} ->
             Fdoublefault{nscore++; if (nscore == 7) {won = nadal}
                                  else {if (turn == f_ad_ct){turn = f_de_ct}
                                        else {turn = n_ad_ct}}} -> NextPt
        33: ServeWide_in{ball =6} -> NadBackhandR
         2: ServeWide_err{ball=9} -> …
        …
NextPt = FedererServe [] NadalServe [] ([won != na] GameOver -> Skip);
```

```
        Federer
   de_ct   ad_ct
  ------+------
  |  1  |  2  |
  |-----|-----|
  |  3  |  4  |
  |===========|
  |  5  |  6  |
  |-----|-----|
  |  7  |  8  |
  ------+------
   ad_ct   de_ct
        Nadal
"9" represents
hit outside
```

Combine Real-Time and Probability [TOSEM'13]

(model checking C# program interface properties)

Passing me without stopping!

Given the C# Program of a lift algorithm

C# Library Editor and Compiler

New Library | New DataType | Open C# Code | Save | Release

```
16  public class LiftControl : ExpressionValue
17  {
18      //-1; for not assigned; i for assigned to i-lift;
19      int[] ExternalRequestsUp;
20      int[] ExternalRequestsDown;
21      //0; for not pressed, 1 for pressed
22      int[][] InternalRequests;
23      //0 for stopped at ground level; ready to go up.
24      int[] LiftStatus;
25
26      public LiftControl()
27      {
28          ExternalRequestsUp = new int[2];
29          ExternalRequestsDown = new int[2];
30          InternalRequests = new int[2][];
31          InternalRequests[0] = new int[2];
32          InternalRequests[1] = new int[2];
33          LiftStatus = new int[2];
34      }
35
36      public LiftControl(int levels, int lifts)
37      {
38          ExternalRequestsUp = new int[levels];
39          ExternalRequestsDown = new int[levels]; ;
40
41          for (int i = 0; i < levels; i++)
42          {
43              ExternalRequestsUp[i] = -1;
44              ExternalRequestsDown[i] = -1;
45          }
46          ;
47          InternalRequests = new int[lifts][];
48          LiftStatus = new int[lifts];
```

No need to understand the details of the code

C# Library Editor and Compiler

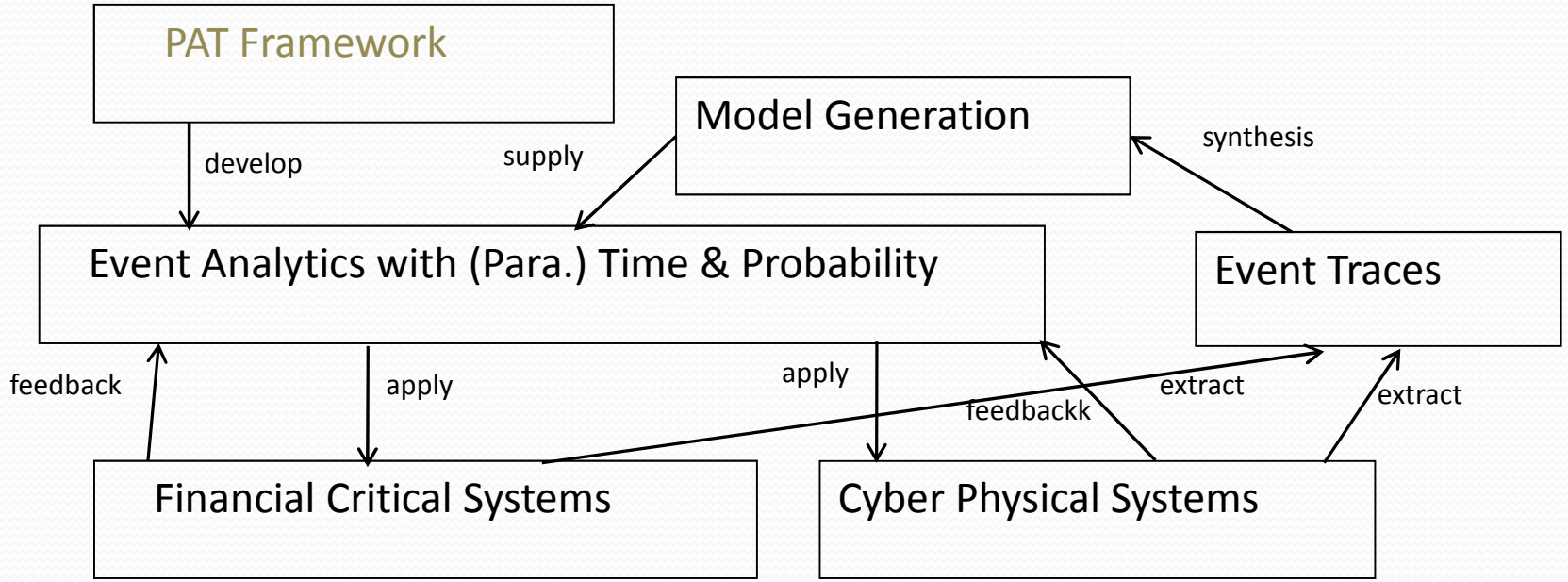
New Library | New DataType | Open C# Code | Save | Release | Build DLL

```
274 public int PassBy (int lift, int level, int up)
275 {
276     //if (isToOpenDoor(lift, level) == 0)
277     //{
278         if (up > 0)
279         {
280             if (ExternalRequestsUp[level] != lift && ExternalRequestsUp[level] >= 0)
281             {
282                 return 1;
283             }
284         }
285         else
286         {
287             if (ExternalRequestsDown[level] >= 0 && ExternalRequestsDown[level] == lift)
288             {
289                 return 1;
290             }
291         }
292     //}
293
294     return 0;
295 }
296
297 public void AddInternalRequest(int lift, int level)
298 {
299     InternalRequests[lift][level] = 1;
300 }
301
302 public int UpdateLiftStatus(int lift, int level, int direction)
303 {
304     LiftStatus[lift] = LiftStatus[lift] + 1;
305
306     return PassBy(lift, level, direction);
307 }
```

PAT checking the C# program with time+probability

```
#import "PAT.Lib.Lift";
#define NoOfFloors 3;
#define NoOfLifts 2;
var<LiftControl> ctrl = new LiftControl(NoOfFloors,NoOfLifts);
var passby = 0;

aSystem = (||| x:{0..NoOfLifts-1} @ Lift(x, 0, 1)) ||| Requests();
Requests() = Request();Request();
Request() = pcase {
            1 : extreq.0.1{ctrl.AssignExternalRequest(0,1)} -> Skip
            1 : intreq.0.0.1{ctrl.AddInternalRequest(0,0)} -> Skip
            1 : intreq.1.0.1{ctrl.AddInternalRequest(1,0)} -> Skip
            1 : extreq.1.0{ctrl.AssignExternalRequest(1,0)} -> Skip
            1 : extreq.1.1{ctrl.AssignExternalRequest(1,1)} -> Skip
            …
       } within[1];
Lift(i, level, direction) = case {
            ctrl.isToOpenDoor(i, level)==1: (serve.level.direction{ctrl.ClearRequests(i, level,
direction)} -> Lift(i, level, direction))
            ctrl.KeepMoving(i, level, direction)==1: (reach.level+direction.direction{passby =
ctrl.UpdateLiftStatus(i, level, direction)} -> Lift(i, level+direction, direction))
            ctrl.HasAssignment(i)==1: changedirection.i{ctrl.ChangeDirection(i)} -> Lift(i, level,
-1*direction)
            default : idle.i -> Lift(i, level, direction)
       } within[2];
#define goal passby == 1;
#assert aSystem reaches goal with prob;
```

Event Analytics (a collaborating proposal with Clarke, Thiagu, Sun, Liu …)



To auto calculate:

- the maximum time delay of a critical event beyond which the overall system reliability will be compromised.
- the minimum probability shift of a specific event that will significantly tip the balance toward the winning strategy.
- …

Also look into:

- Linking to Operational Research and Machine Learning techniques/tools?

Some PAT info

- PAT is available at http://pat.comp.nus.edu.sg
- 1Million lines of C# code, 15 verification systems with 200+ build in examples, 100+ publications (CAV, FM, ICSE, ASE, TSE, TOSEM …).
- Used as an educational tool in many universities.
- Attracted 3000+ registered users in the last 7 years from 800+ organizations in 72 countries, e.g. Microsoft, HP, Sony, Hitachi, Canon, Mitsubishi, NTT, Toyota …
- Japanese PAT User group formed in Sep 2009:

Many Thanks to Ed!

- should have listened to you more seriously back in 1994
- Look forward to seeing you again soon in Singapore!

Founding Members:
Hiroshi Fujimoto
Nobukazu Yoshioka
Toshiyuki Fujikura
Kenji Taguchi
Masaru Nagaku
Kazuto MATSUI

Commercialized in multiple countries, esp. in Japan, thanks to CATS!