

## Applying TLA+ in a Safety-Critical Railway Project

Stefan Resch  
Dependable Systems Architect, Thales



This project has received funding from the ECSEL Joint Undertaking under grant agreement No 692455. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.



# Overview TAS Control Platform

■ **Safety approval according to CENELEC 50129 SIL 4**

■ **Safety layer**

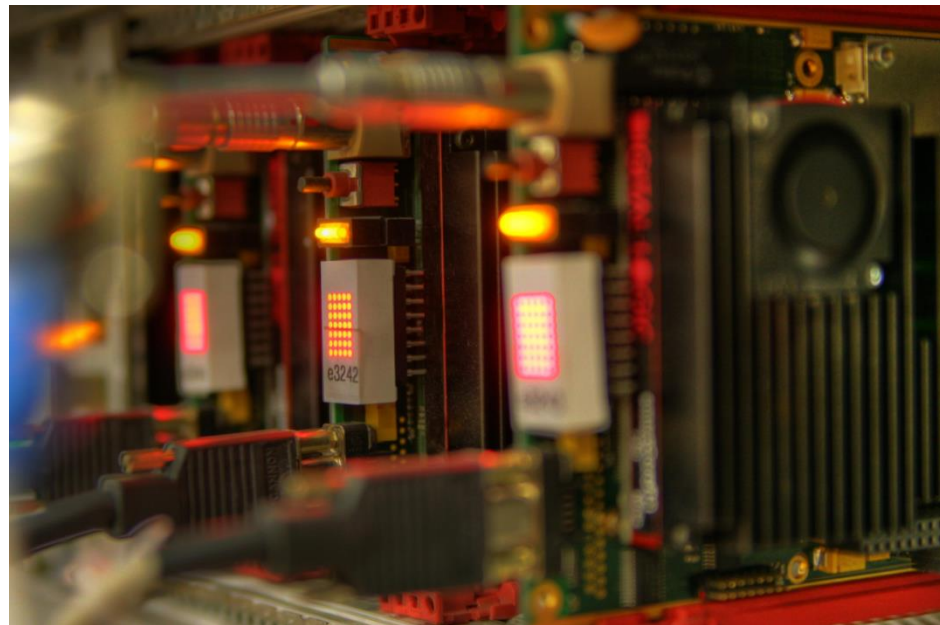
- Fault tolerance
- Health monitoring
- Various redundancy architectures

■ **Board support package**

- Communications interfaces / drivers
- Some are very specific

■ **Based on COTS hardware / operating system**

■ **Support 25 years of application business logic (with changing underlying hardware and software)**



■ **TAS Control Platform in about 70% of new safety-critical products in Thales Ground Transportation**

# Formal Modelling Objectives

## ■ New Redundancy Architecture

## ■ Gain confidence in design

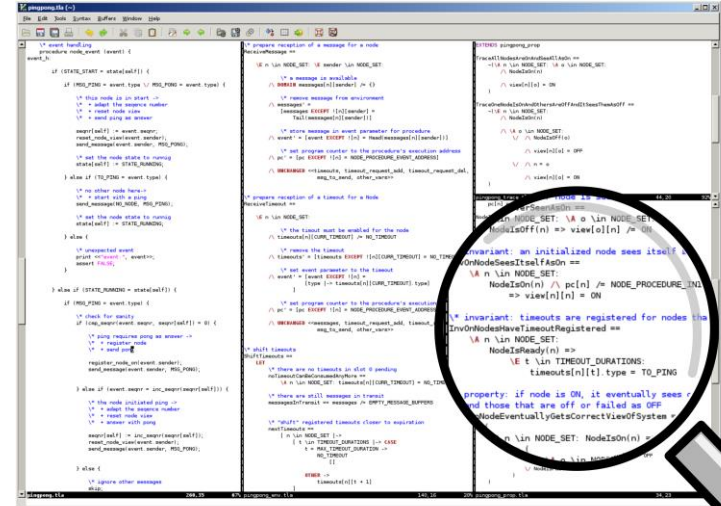
- Relevant results/insights

## ■ Feasible

- Time and effort

## ■ Close representation to implementation

- Reduce bugs introduced between model and coding
- Get feedback from experienced experts in team



# Some time later ...

# The Chosen Approach

## Formal model in TLA+ and PlusCal

- Algorithm specification in C-like language
- Linear temporal logic formulas
- Model checking with TLC

## Efficient model checking through a good level of abstraction

- Prerequisite for design process

## Property-driven design

- Process that provides confidence in model

## C code generation from model

- Close gap between model and implementation

# Formal Model in TLA+ and PlusCal

## Algorithm defined in PlusCal

## Environment in TLA+

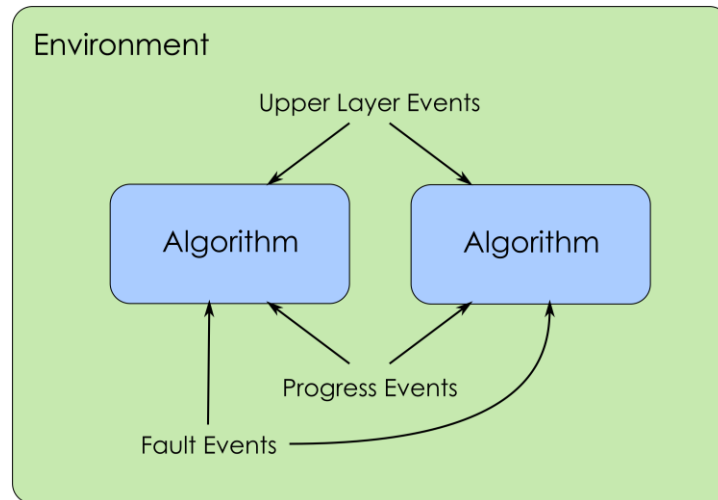
### > 3 classes of events

- Progress
- Upper Layer
- Fault

## Properties

- > Invariants: e.g. no two masters
- > Liveness: e.g. all nodes eventually get a correct view of the system

## TLC model checker



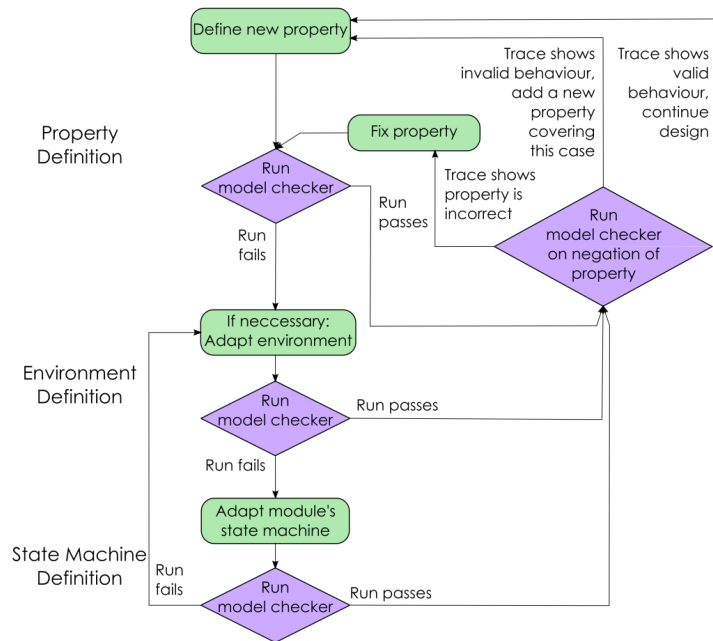
# So we found a model checker ...

OPEN

## Analogous to test driven development

## Gain confidence in model

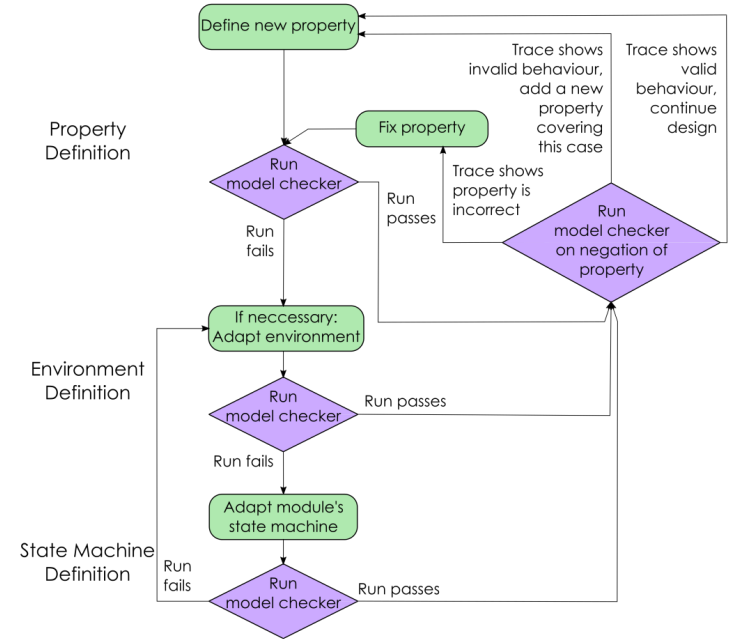
- Observe many traces of model
- Check that properties capture intended behaviour
- Fast feedback to design decisions
- Feedback from increasing size of state space





## Recap test driven development

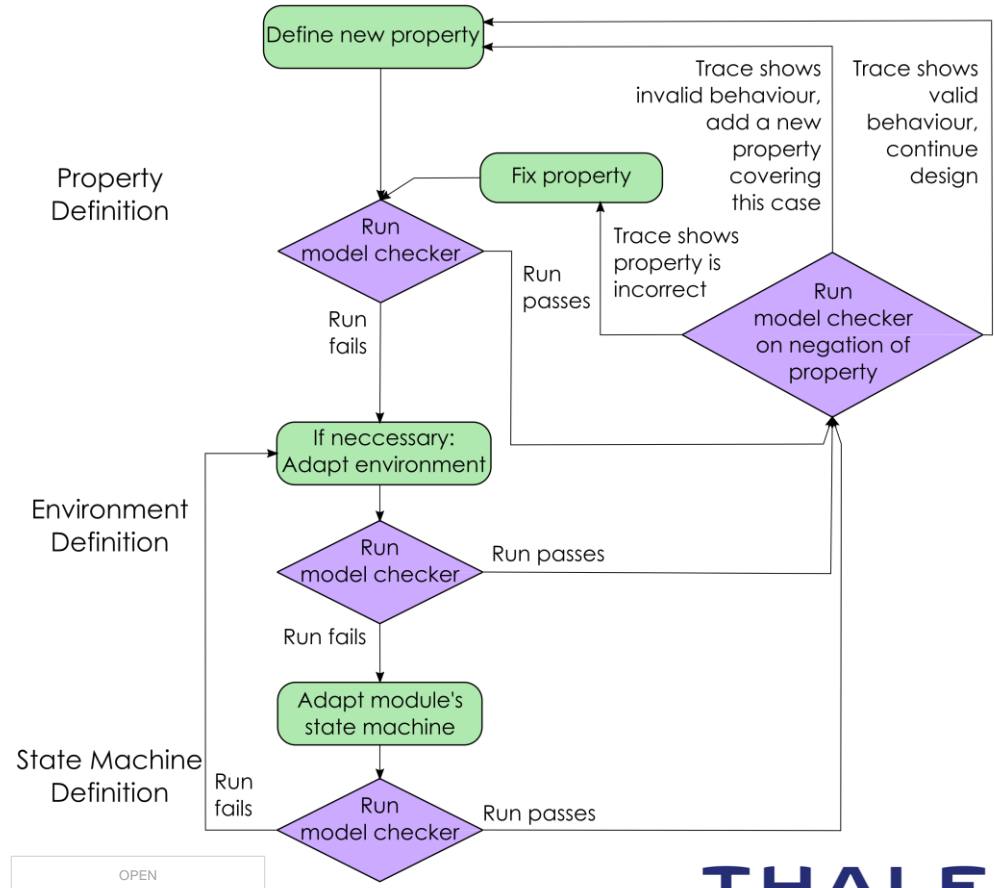
- Write new test
- Execute test → fail
- Write implementation
- Execute all tests → pass
- Refactor



# Property-Driven Design

## Steps

- Define property
- Enhance environment
- Adapt algorithm
- Cross check with negation of property

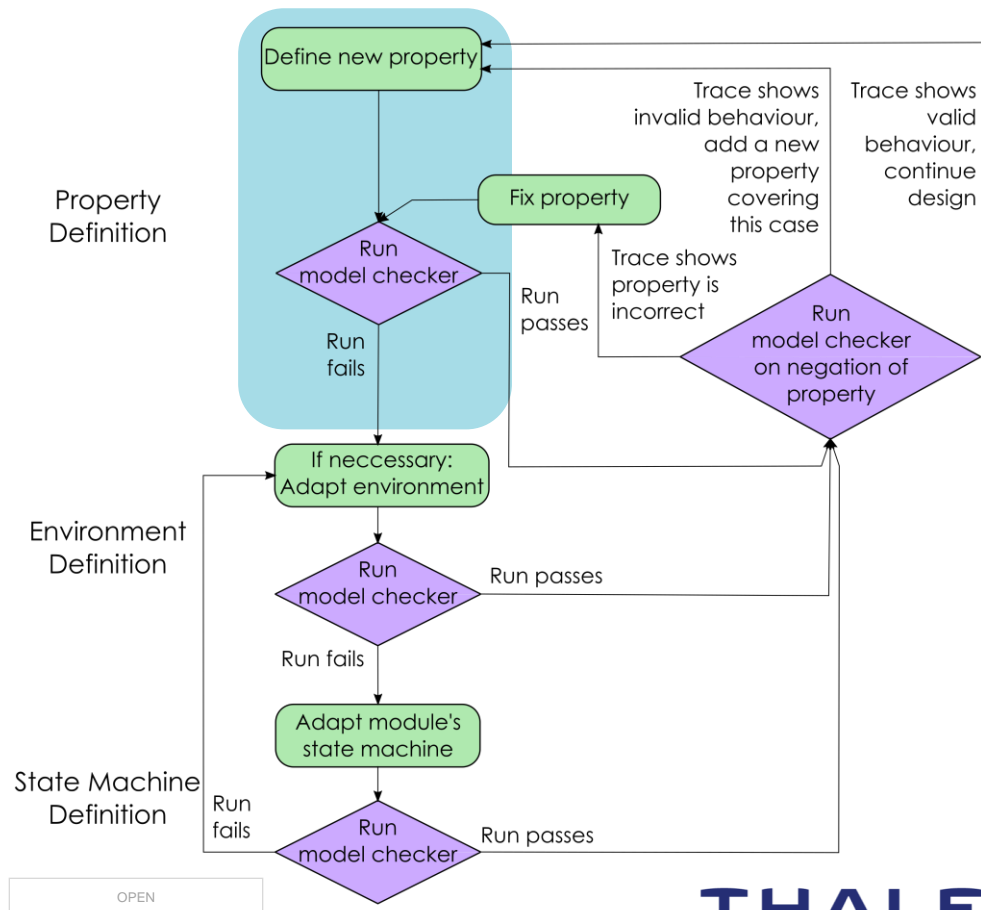


OPEN

# Property-Driven Design

## Steps

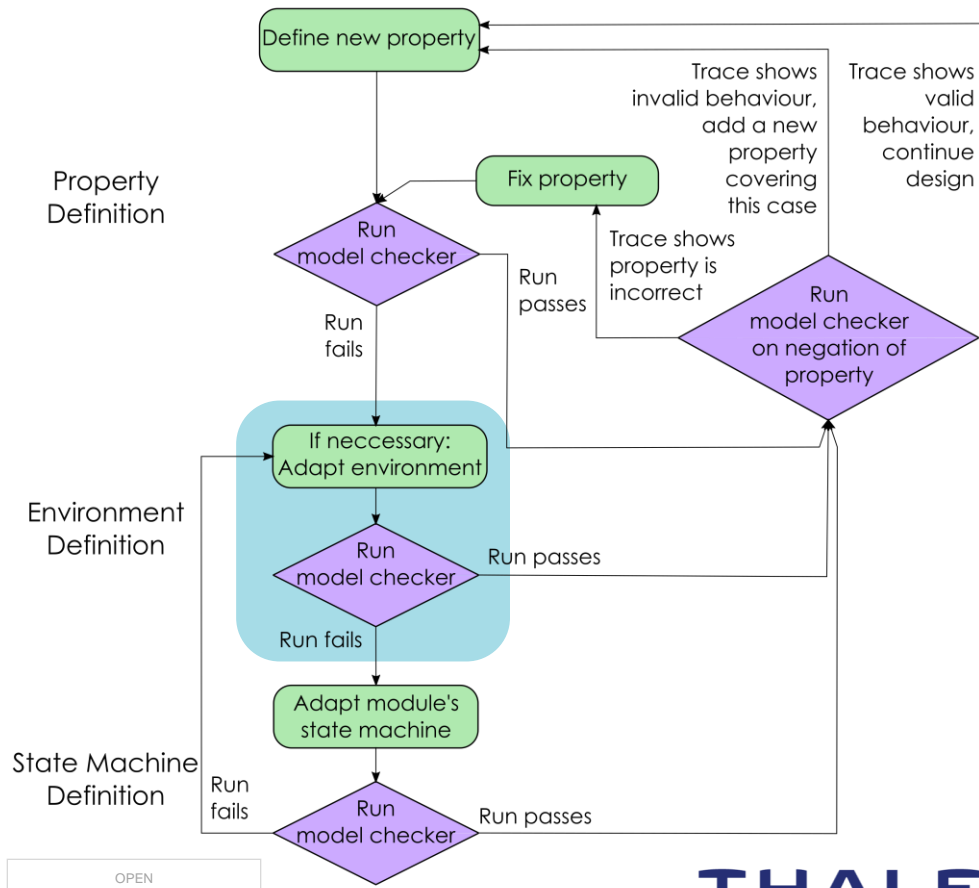
- Define property
- Enhance environment
- Adapt algorithm
- Cross check with negation of property



# Property-Driven Design

## Steps

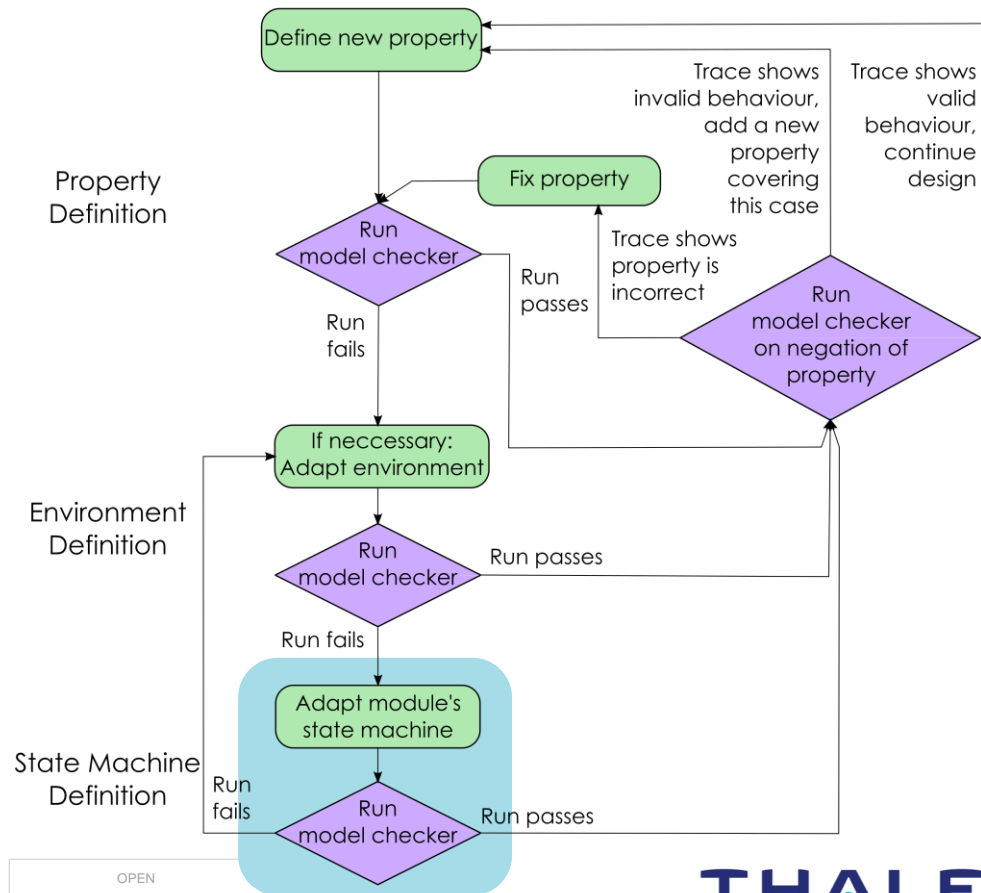
- Define property
- Enhance environment
- Adapt algorithm
- Cross check with negation of property



# Property-Driven Design

## Steps

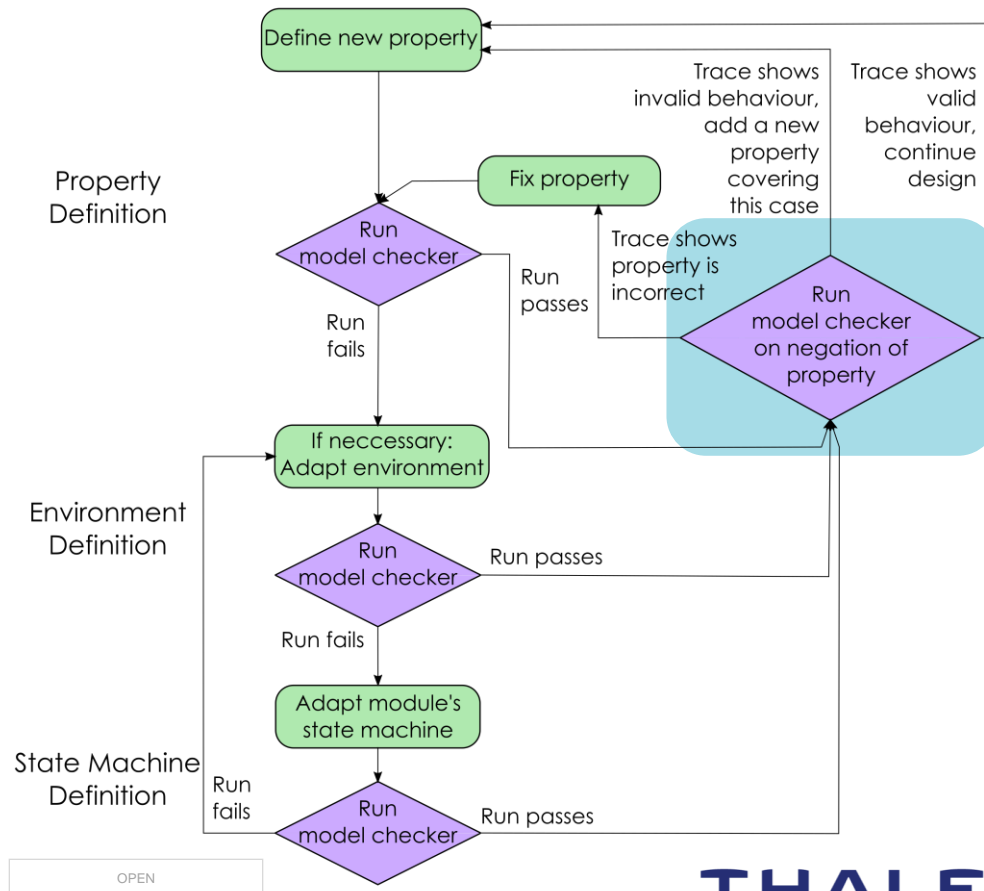
- Define property
- Enhance environment
- Adapt algorithm
- Cross check with negation of property



# Property-Driven Design

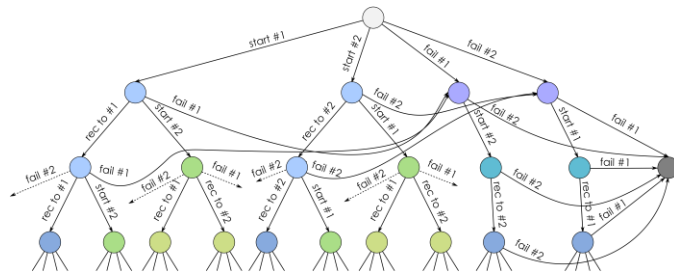
## Steps

- Define property
- Enhance environment
- Adapt algorithm
- Cross check with negation of property



## Track size of state space between model checker runs

- Constant size but algorithm got more complex
  - Most likely added dead code
- Sudden increase of size
  - Most likely a bug in specification



## How properties drive the process

### > Invariants

- Help detecting “bugs”

### > Liveness

- Force enhancement of algorithm



# Property-Driven Design

■ Instant feedback supports design decisions

■ Many traces build confidence in model

■ Unanticipated scenarios were found

➤ Longest trace consisted of more than 30 steps

■ Major drawback is time necessary for modelling

OPEN

THALES

# PlusCal to C Transformation

## PlusCal

```
\* initialization
procedure node_init () {
init:
    state[self] := STATE_START;

\* set all but own state to off
    reset_node_view(NO_NODE);

\* start a timeout to send
\* ping for the first time
    start_timeout(TO_PING,
        TO_PING_STARTUP_TIME);
}
```

## C

```
// initialization
void node_init (void) {

    state = STATE_START;

// set all but own state to off
    reset_node_view(NO_NODE);

// start a timeout to send ping
// for the first time
    start_timeout(TO_PING,
        TO_PING_STARTUP_TIME);
}
```

OPEN

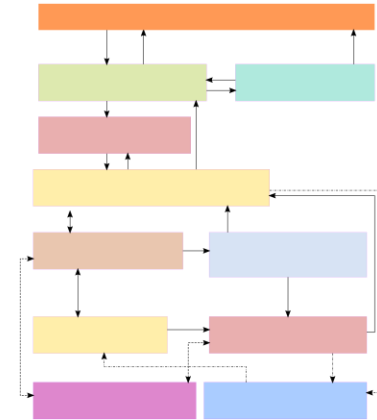
# Defining Different Modules

## Specific environment definition necessary

- Avoid state space explosion
- Substantial effort

## Strategy for interfaces is essential

## Layering seems to be the simplest approach

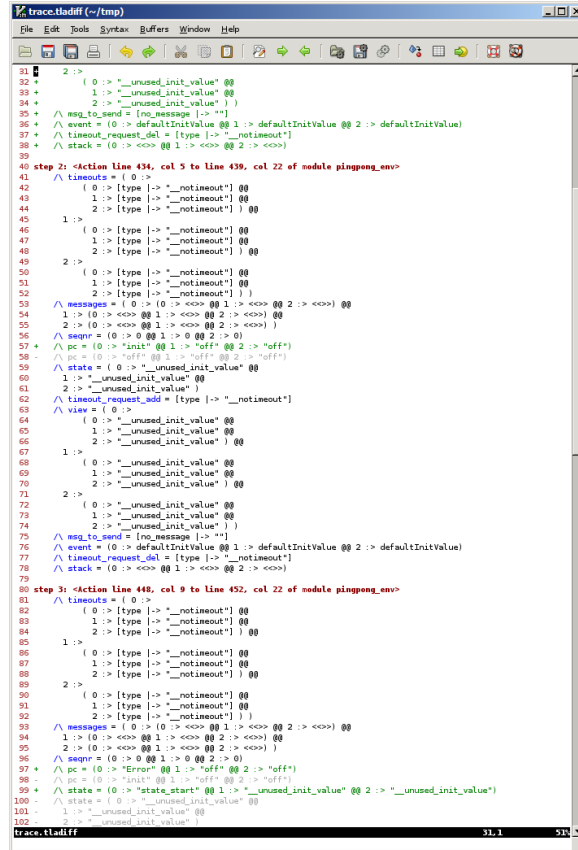


# Potential Improvements

## Readable Traces

## Animation

## Faster check of liveness properties



```
31 | 2 :>
32 | ( 0 :> "_unused_init_value" @0
33 | 1 :> "_unused_init_value" @0
34 | 2 :> "_unused_init_value" ) @0
35 | /\ msg_to_send = [no_message |-> ""]
36 | /\ event = ( 0 :> defaultInitValue @0 1 :> defaultInitValue @0 2 :> defaultInitValue )
37 | /\ timeout_request_del = [type |-> "_notimeout"]
38 | /\ stack = ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> )
39 |
40 step 2: <Action line 434, col 5 to line 439, col 22 of module pingpong_env>
41 | /\ timeouts = ( 0 :>
42 | ( 0 :> [type |-> "_notimeout"] @0
43 | 1 :> [type |-> "_notimeout"] @0
44 | 2 :> [type |-> "_notimeout"] ) @0
45 | 1 :>
46 | ( 0 :> [type |-> "_notimeout"] @0
47 | 1 :> [type |-> "_notimeout"] @0
48 | 2 :> [type |-> "_notimeout"] ) @0
49 | 2 :>
50 | ( 0 :> [type |-> "_notimeout"] @0
51 | 1 :> [type |-> "_notimeout"] @0
52 | 2 :> [type |-> "_notimeout"] ) @0
53 | /\ messages = ( 0 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
54 | 1 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
55 | 2 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
56 | /\ seqnr = ( 0 :> 0 @0 1 :> 0 @0 2 :> 0 )
57 | /\ pc = ( 0 :> "init" @0 1 :> "off" @0 2 :> "off" )
58 | /\ pc = ( 0 :> "off" @0 1 :> "off" @0 2 :> "off" )
59 | /\ state = ( 0 :> "_unused_init_value" @0
60 | 1 :> "_unused_init_value" @0
61 | 2 :> "_unused_init_value" ) @0
62 | /\ timeout_request_add = [type |-> "_notimeout"]
63 | /\ view = ( 0 :>
64 | ( 0 :> "_unused_init_value" @0
65 | 1 :> "_unused_init_value" @0
66 | 2 :> "_unused_init_value" ) @0
67 | 1 :>
68 | ( 0 :> "_unused_init_value" @0
69 | 1 :> "_unused_init_value" @0
70 | 2 :> "_unused_init_value" ) @0
71 | 2 :>
72 | ( 0 :> "_unused_init_value" @0
73 | 1 :> "_unused_init_value" @0
74 | 2 :> "_unused_init_value" ) @0
75 | /\ msg_to_send = [no_message |-> ""]
76 | /\ event = ( 0 :> defaultInitValue @0 1 :> defaultInitValue @0 2 :> defaultInitValue )
77 | /\ timeout_request_del = [type |-> "_notimeout"]
78 | /\ stack = ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> )
79 |
80 step 3: <Action line 448, col 9 to line 452, col 22 of module pingpong_env>
81 | /\ timeouts = ( 0 :>
82 | ( 0 :> [type |-> "_notimeout"] @0
83 | 1 :> [type |-> "_notimeout"] @0
84 | 2 :> [type |-> "_notimeout"] ) @0
85 | 1 :>
86 | ( 0 :> [type |-> "_notimeout"] @0
87 | 1 :> [type |-> "_notimeout"] @0
88 | 2 :> [type |-> "_notimeout"] ) @0
89 | 2 :>
90 | ( 0 :> [type |-> "_notimeout"] @0
91 | 1 :> [type |-> "_notimeout"] @0
92 | 2 :> [type |-> "_notimeout"] ) @0
93 | /\ messages = ( 0 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
94 | 1 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
95 | 2 :> ( 0 :> <<<> @0 1 :> <<<> @0 2 :> <<<> ) @0
96 | /\ seqnr = ( 0 :> 0 @0 1 :> 0 @0 2 :> 0 )
97 | /\ pc = ( 0 :> "Error" @0 1 :> "off" @0 2 :> "off" )
98 | /\ pc = ( 0 :> "init" @0 1 :> "off" @0 2 :> "off" )
99 | /\ state = ( 0 :> "state_start" @0 1 :> "_unused_init_value" @0 2 :> "_unused_init_value" )
100 | /\ state = ( 0 :> "_unused_init_value" @0
101 | 1 :> "_unused_init_value" @0
102 | 2 :> "_unused_init_value" ) @0
103 |
104 |
105 |
106 |
107 |
108 |
109 |
110 |
111 |
112 |
113 |
114 |
115 |
116 |
117 |
118 |
119 |
120 |
121 |
122 |
123 |
124 |
125 |
126 |
127 |
128 |
129 |
130 |
131 |
132 |
133 |
134 |
135 |
136 |
137 |
138 |
139 |
140 |
141 |
142 |
143 |
144 |
145 |
146 |
147 |
148 |
149 |
150 |
151 |
152 |
153 |
154 |
155 |
156 |
157 |
158 |
159 |
160 |
161 |
162 |
163 |
164 |
165 |
166 |
167 |
168 |
169 |
170 |
171 |
172 |
173 |
174 |
175 |
176 |
177 |
178 |
179 |
180 |
181 |
182 |
183 |
184 |
185 |
186 |
187 |
188 |
189 |
190 |
191 |
192 |
193 |
194 |
195 |
196 |
197 |
198 |
199 |
200 |
201 |
202 |
203 |
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |
224 |
225 |
226 |
227 |
228 |
229 |
230 |
231 |
232 |
233 |
234 |
235 |
236 |
237 |
238 |
239 |
240 |
241 |
242 |
243 |
244 |
245 |
246 |
247 |
248 |
249 |
250 |
251 |
252 |
253 |
254 |
255 |
256 |
257 |
258 |
259 |
260 |
261 |
262 |
263 |
264 |
265 |
266 |
267 |
268 |
269 |
270 |
271 |
272 |
273 |
274 |
275 |
276 |
277 |
278 |
279 |
280 |
281 |
282 |
283 |
284 |
285 |
286 |
287 |
288 |
289 |
290 |
291 |
292 |
293 |
294 |
295 |
296 |
297 |
298 |
299 |
300 |
301 |
302 |
303 |
304 |
305 |
306 |
307 |
308 |
309 |
310 |
311 |
312 |
313 |
314 |
315 |
316 |
317 |
318 |
319 |
320 |
321 |
322 |
323 |
324 |
325 |
326 |
327 |
328 |
329 |
330 |
331 |
332 |
333 |
334 |
335 |
336 |
337 |
338 |
339 |
340 |
341 |
342 |
343 |
344 |
345 |
346 |
347 |
348 |
349 |
350 |
351 |
352 |
353 |
354 |
355 |
356 |
357 |
358 |
359 |
360 |
361 |
362 |
363 |
364 |
365 |
366 |
367 |
368 |
369 |
370 |
371 |
372 |
373 |
374 |
375 |
376 |
377 |
378 |
379 |
380 |
381 |
382 |
383 |
384 |
385 |
386 |
387 |
388 |
389 |
390 |
391 |
392 |
393 |
394 |
395 |
396 |
397 |
398 |
399 |
400 |
401 |
402 |
403 |
404 |
405 |
406 |
407 |
408 |
409 |
410 |
411 |
412 |
413 |
414 |
415 |
416 |
417 |
418 |
419 |
420 |
421 |
422 |
423 |
424 |
425 |
426 |
427 |
428 |
429 |
430 |
431 |
432 |
433 |
434 |
435 |
436 |
437 |
438 |
439 |
440 |
441 |
442 |
443 |
444 |
445 |
446 |
447 |
448 |
449 |
450 |
451 |
452 |
453 |
454 |
455 |
456 |
457 |
458 |
459 |
460 |
461 |
462 |
463 |
464 |
465 |
466 |
467 |
468 |
469 |
470 |
471 |
472 |
473 |
474 |
475 |
476 |
477 |
478 |
479 |
480 |
481 |
482 |
483 |
484 |
485 |
486 |
487 |
488 |
489 |
490 |
491 |
492 |
493 |
494 |
495 |
496 |
497 |
498 |
499 |
500 |
501 |
502 |
503 |
504 |
505 |
506 |
507 |
508 |
509 |
510 |
511 |
512 |
513 |
514 |
515 |
516 |
517 |
518 |
519 |
520 |
521 |
522 |
523 |
524 |
525 |
526 |
527 |
528 |
529 |
530 |
531 |
532 |
533 |
534 |
535 |
536 |
537 |
538 |
539 |
540 |
541 |
542 |
543 |
544 |
545 |
546 |
547 |
548 |
549 |
550 |
551 |
552 |
553 |
554 |
555 |
556 |
557 |
558 |
559 |
560 |
561 |
562 |
563 |
564 |
565 |
566 |
567 |
568 |
569 |
570 |
571 |
572 |
573 |
574 |
575 |
576 |
577 |
578 |
579 |
580 |
581 |
582 |
583 |
584 |
585 |
586 |
587 |
588 |
589 |
590 |
591 |
592 |
593 |
594 |
595 |
596 |
597 |
598 |
599 |
600 |
601 |
602 |
603 |
604 |
605 |
606 |
607 |
608 |
609 |
610 |
611 |
612 |
613 |
614 |
615 |
616 |
617 |
618 |
619 |
620 |
621 |
622 |
623 |
624 |
625 |
626 |
627 |
628 |
629 |
630 |
631 |
632 |
633 |
634 |
635 |
636 |
637 |
638 |
639 |
640 |
641 |
642 |
643 |
644 |
645 |
646 |
647 |
648 |
649 |
650 |
651 |
652 |
653 |
654 |
655 |
656 |
657 |
658 |
659 |
660 |
661 |
662 |
663 |
664 |
665 |
666 |
667 |
668 |
669 |
670 |
671 |
672 |
673 |
674 |
675 |
676 |
677 |
678 |
679 |
680 |
681 |
682 |
683 |
684 |
685 |
686 |
687 |
688 |
689 |
690 |
691 |
692 |
693 |
694 |
695 |
696 |
697 |
698 |
699 |
700 |
701 |
702 |
703 |
704 |
705 |
706 |
707 |
708 |
709 |
710 |
711 |
712 |
713 |
714 |
715 |
716 |
717 |
718 |
719 |
720 |
721 |
722 |
723 |
724 |
725 |
726 |
727 |
728 |
729 |
730 |
731 |
732 |
733 |
734 |
735 |
736 |
737 |
738 |
739 |
740 |
741 |
742 |
743 |
744 |
745 |
746 |
747 |
748 |
749 |
750 |
751 |
752 |
753 |
754 |
755 |
756 |
757 |
758 |
759 |
760 |
761 |
762 |
763 |
764 |
765 |
766 |
767 |
768 |
769 |
770 |
771 |
772 |
773 |
774 |
775 |
776 |
777 |
778 |
779 |
780 |
781 |
782 |
783 |
784 |
785 |
786 |
787 |
788 |
789 |
790 |
791 |
792 |
793 |
794 |
795 |
796 |
797 |
798 |
799 |
800 |
801 |
802 |
803 |
804 |
805 |
806 |
807 |
808 |
809 |
810 |
811 |
812 |
813 |
814 |
815 |
816 |
817 |
818 |
819 |
820 |
821 |
822 |
823 |
824 |
825 |
826 |
827 |
828 |
829 |
830 |
831 |
832 |
833 |
834 |
835 |
836 |
837 |
838 |
839 |
840 |
841 |
842 |
843 |
844 |
845 |
846 |
847 |
848 |
849 |
850 |
851 |
852 |
853 |
854 |
855 |
856 |
857 |
858 |
859 |
860 |
861 |
862 |
863 |
864 |
865 |
866 |
867 |
868 |
869 |
870 |
871 |
872 |
873 |
874 |
875 |
876 |
877 |
878 |
879 |
880 |
881 |
882 |
883 |
884 |
885 |
886 |
887 |
888 |
889 |
890 |
891 |
892 |
893 |
894 |
895 |
896 |
897 |
898 |
899 |
900 |
901 |
902 |
903 |
904 |
905 |
906 |
907 |
908 |
909 |
910 |
911 |
912 |
913 |
914 |
915 |
916 |
917 |
918 |
919 |
920 |
921 |
922 |
923 |
924 |
925 |
926 |
927 |
928 |
929 |
930 |
931 |
932 |
933 |
934 |
935 |
936 |
937 |
938 |
939 |
940 |
941 |
942 |
943 |
944 |
945 |
946 |
947 |
948 |
949 |
950 |
951 |
952 |
953 |
954 |
955 |
956 |
957 |
958 |
959 |
960 |
961 |
962 |
963 |
964 |
965 |
966 |
967 |
968 |
969 |
970 |
971 |
972 |
973 |
974 |
975 |
976 |
977 |
978 |
979 |
980 |
981 |
982 |
983 |
984 |
985 |
986 |
987 |
988 |
989 |
990 |
991 |
992 |
993 |
994 |
995 |
996 |
997 |
998 |
999 |
1000 |
```

OPEN

■ Valuable insights to system design

■ Justifiable overhead

■ Very close to implementation

## Environment with example algorithm

➤ <https://github.com/stresch/pingpong>

OPEN