Nose Gear (NG) Velocity Challenge

A Sketch of an approach

Andy McCallum	Muretex
Colin O'Halloran	QinetiQ
Karen Stephenson	QinetiQ

www.QinetiQ.com

The nature of the challenge

- The property that has been set as a challenge to be proven is not about software.
- It is a relationship between the real physical world and estimating software that samples sensed information about the real physical world.
- To prove the property a:
  - continuous time mathematical model of the relevant part of the physical real world is needed;
  - discrete time mathematical model of the estimating software is needed;
  - and a relationship between discrete and continuous time.
- Suppose “SW_Spec(n)”, “Phys_Spec(t)” and “Sample(n,t)” denote predicates over the: discrete software; physical system; and the relationship between the discrete and continuous time, respectively.

- A proof of the property P(t) might take the form of a proof that

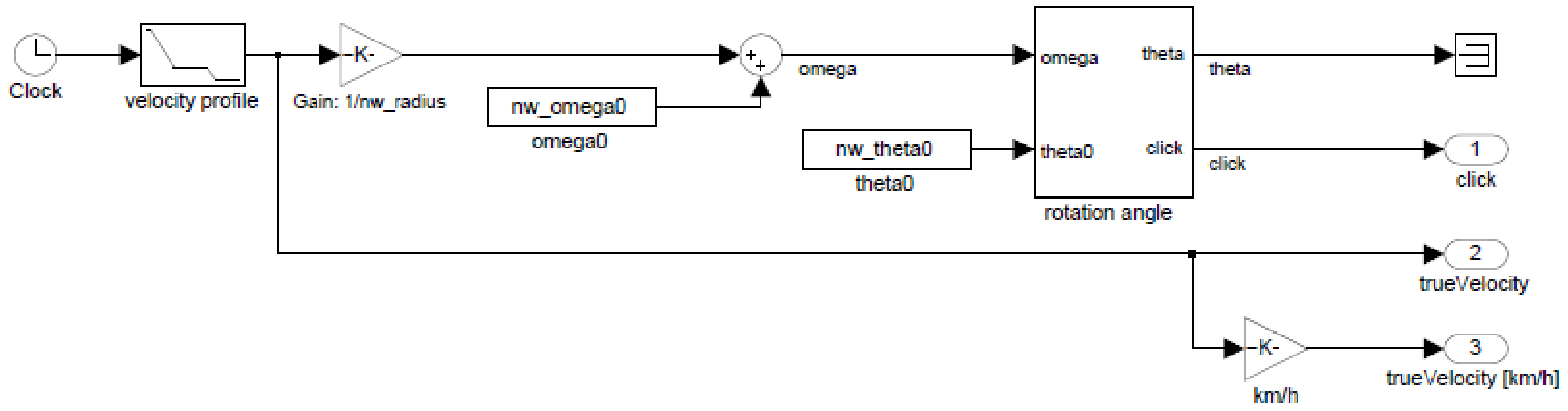
$$\forall\, t \,.\, SW\_Spec(n) \wedge Phys\_Spec(t) \wedge Sample(n,t) \Rightarrow P(t)$$

- From a certification point of view, where is the evidence that the predicates SW_Spec(n), Phys_Spec(t) and Sample(n,t) are valid?

- Even if a proof is done, so what?

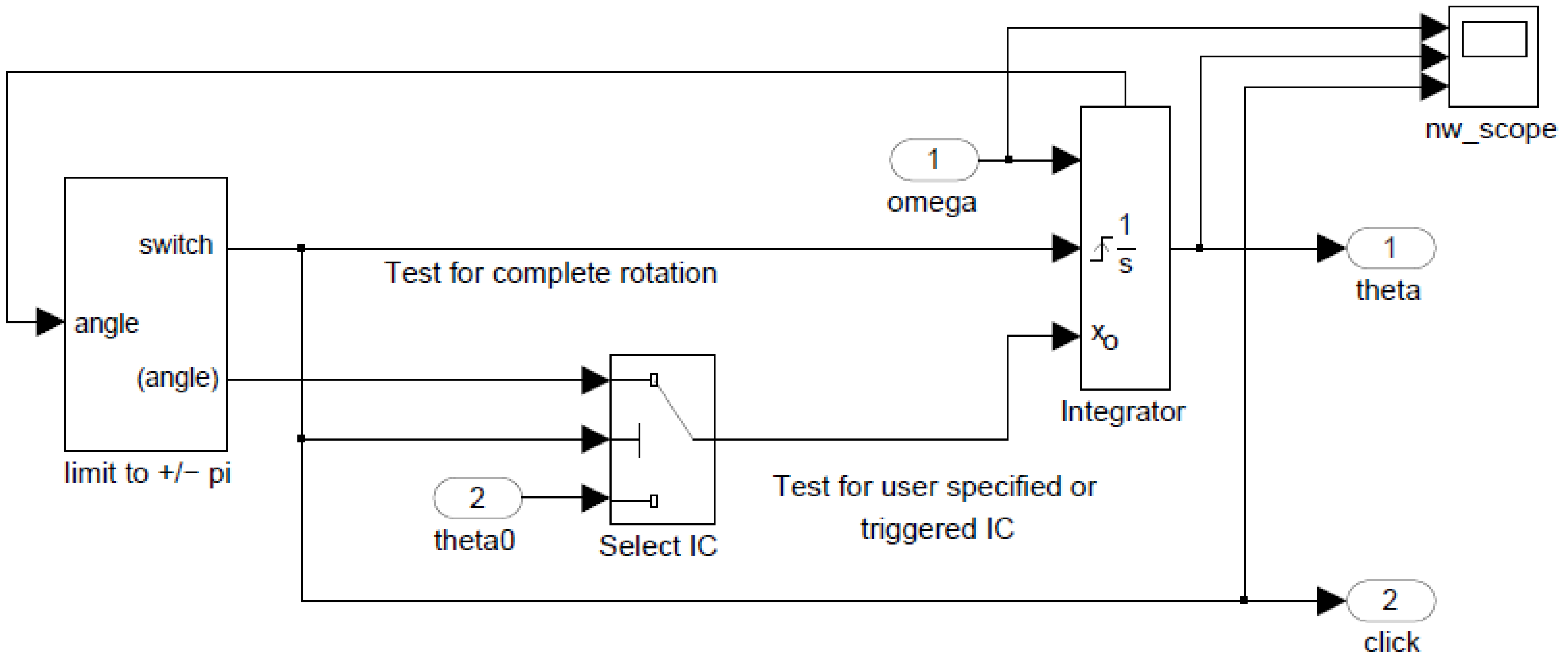
- Need to be able to define the predicates in a way that can be validated for certification.

Phys_Spec(t)



Clock
velocity profile
-K-
Gain: 1/nw_radius
nw_omega0
omega0
+
+
omega
nw_theta0
theta0
omega
theta0
theta
click
rotation angle
theta
click
1
click
2
trueVelocity
-K-
km/h
3
trueVelocity [km/h]

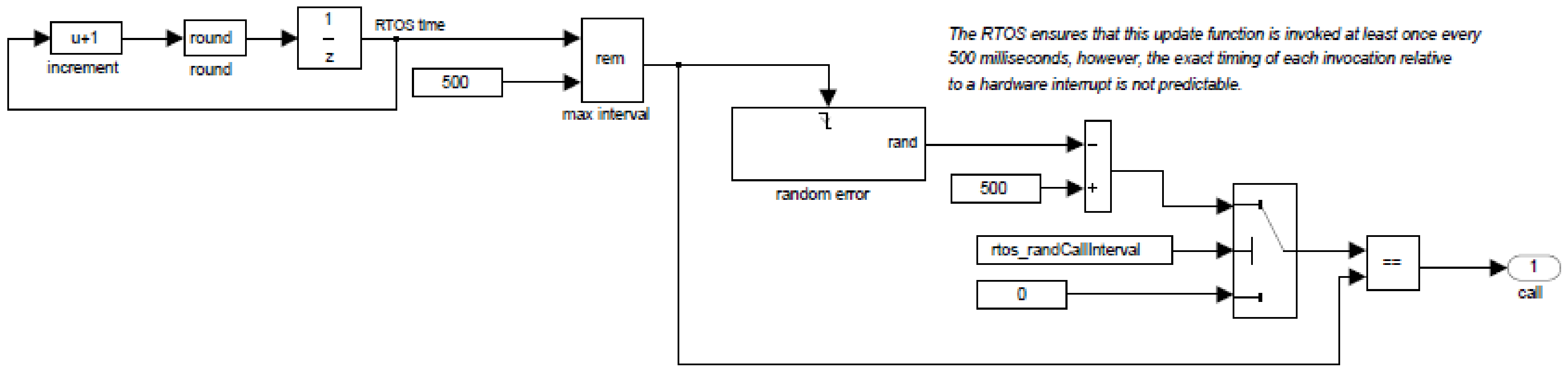
Rotation Angle Component



Comments on the Simulink Physical Model

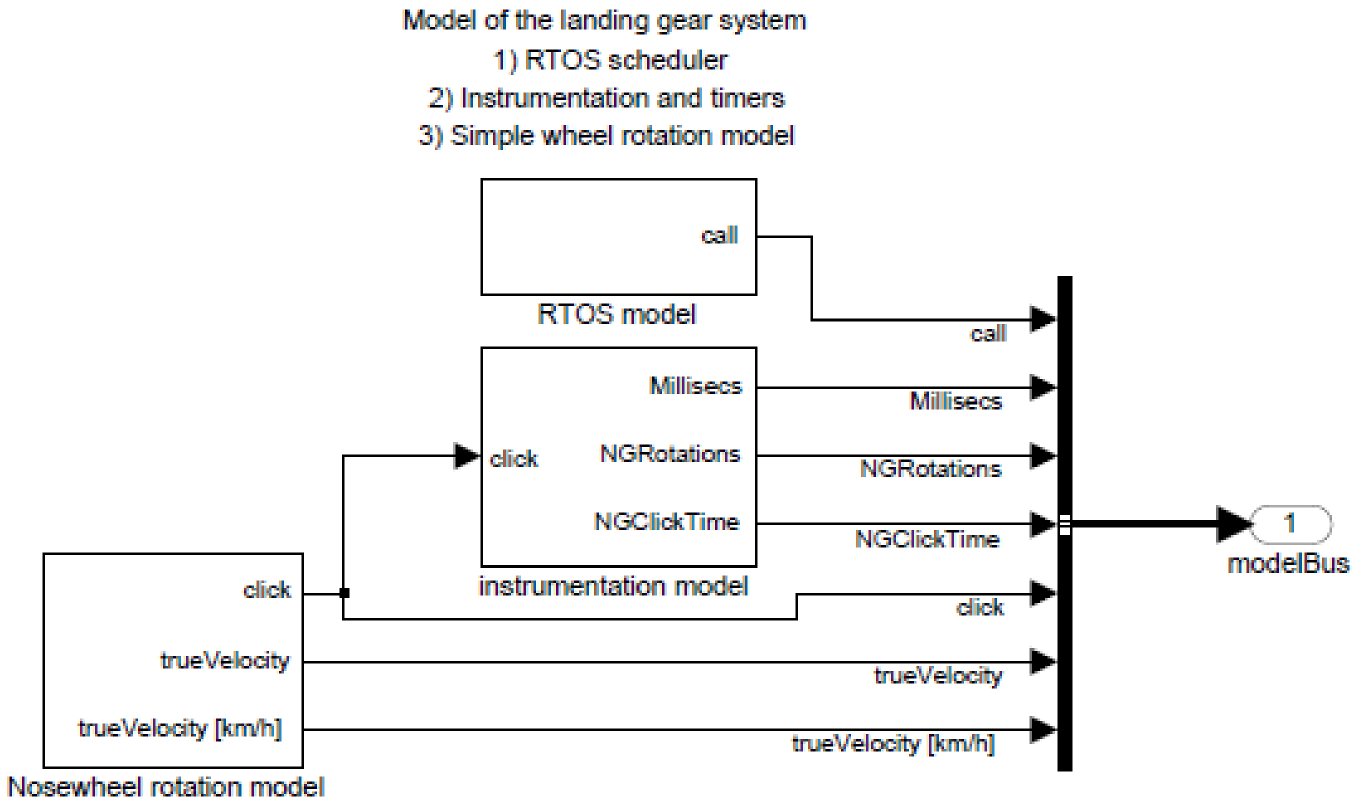
- The model is a first cut from the requirements and not the pseudo-code, although the pseudo-code has been looked at.
- The asynchronous relationship between invocations of the update function and updates to the two counters introduces some complications that would also make conventional verification very challenging.
- There are about a dozen assumptions that need to be made, e.g.,
  - the maximum acceleration of the aircraft on the ground,
  - the amount of variability in the timing of the pulse generated by the electro-mechanical sensor connected to the nose gear wheel.
- Discovering and formalizing these assumptions may be one of the most interesting aspects of the assurance cases.
- Hopefully first cut of Simulink model isn't overly simplistic.

Scheduling

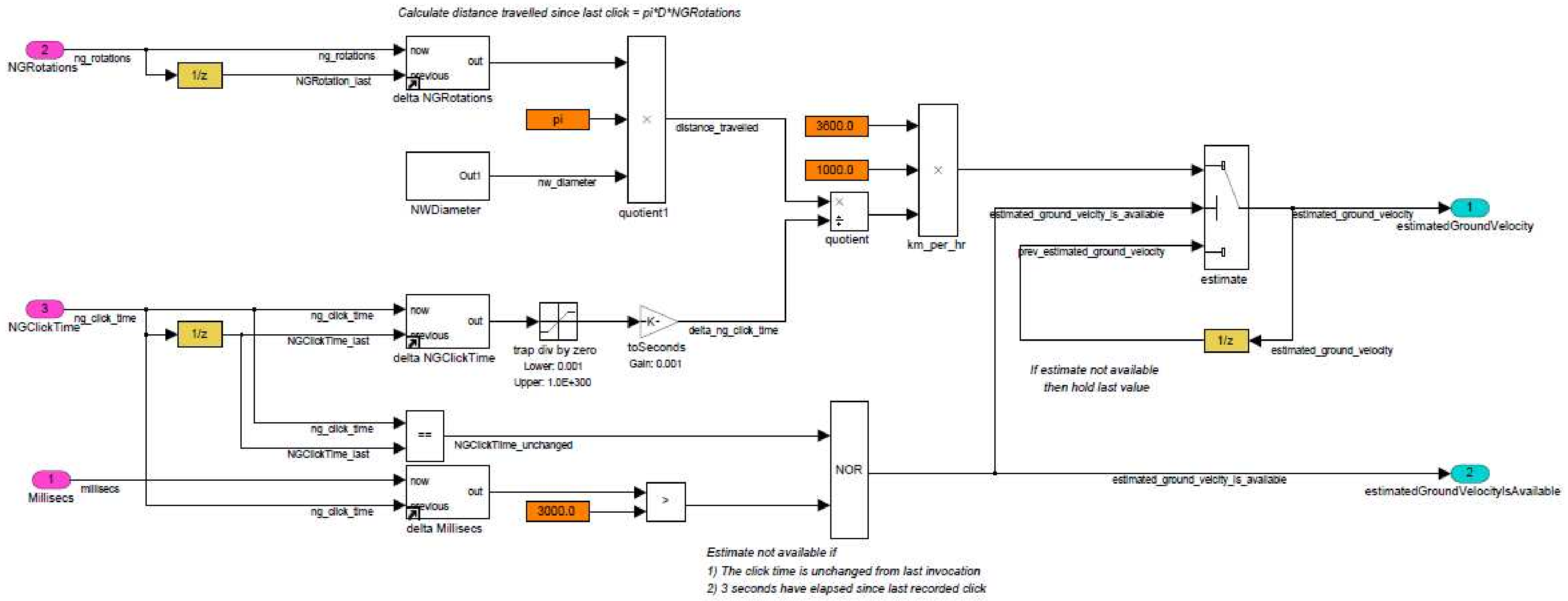


The RTOS ensures that this update function is invoked at least once every 500 milliseconds, however, the exact timing of each invocation relative to a hardware interrupt is not predictable.

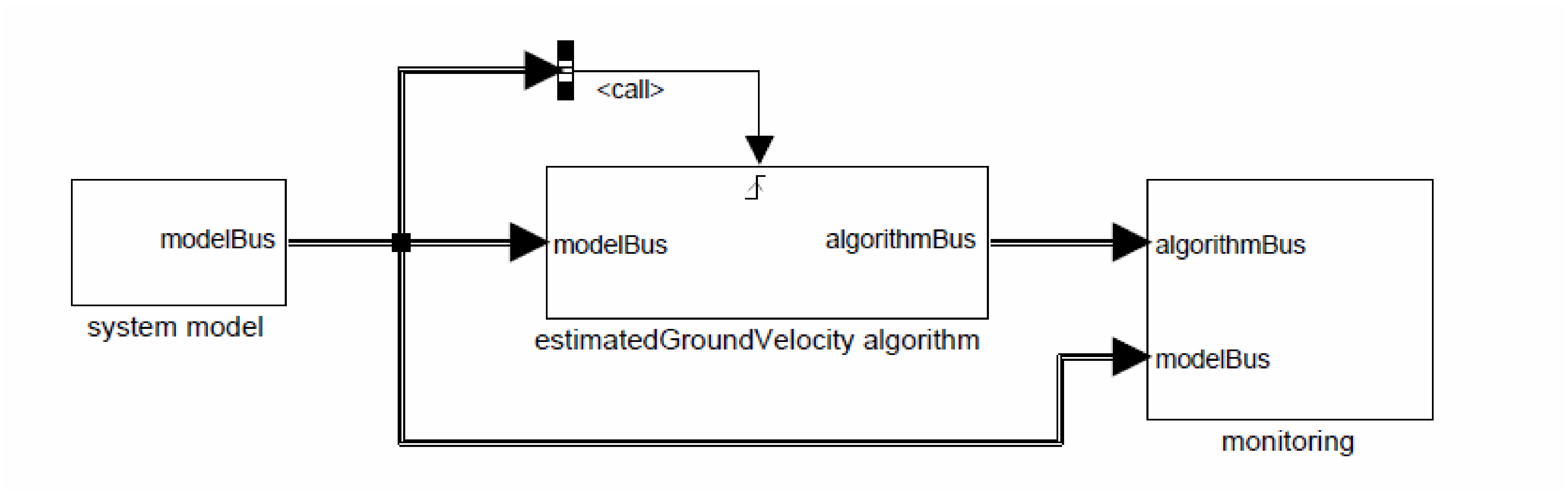
Scheduler and Physical Model (with instrumentation)



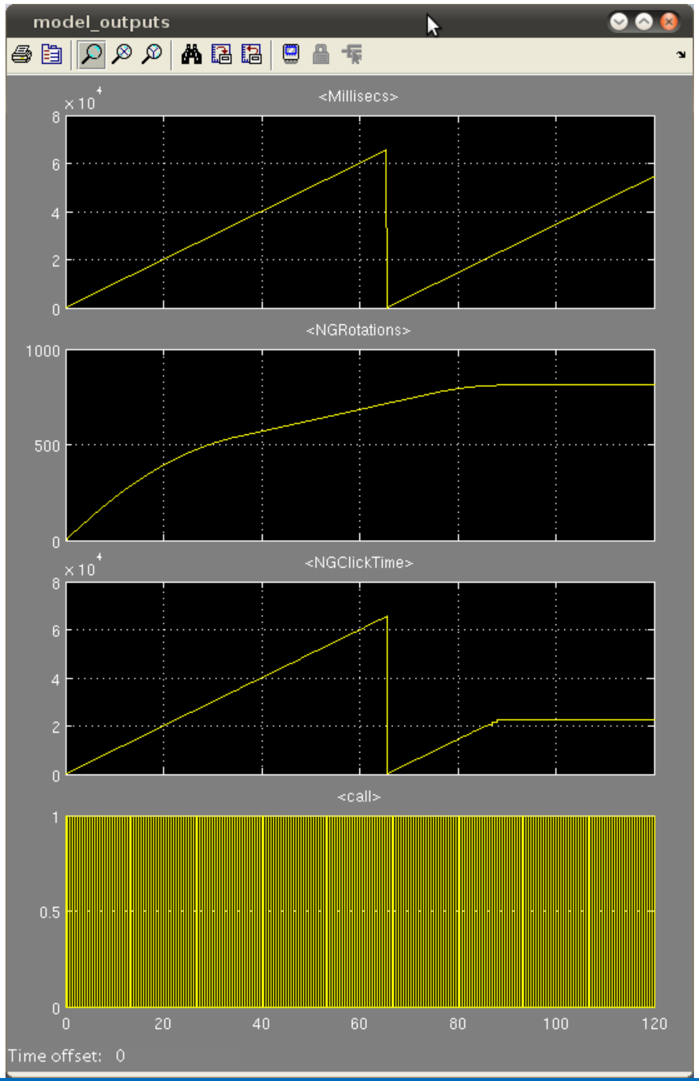
Discrete Simulink Model of Software



Overall Model



Quick example plots of some of the kinds of signals being generated.

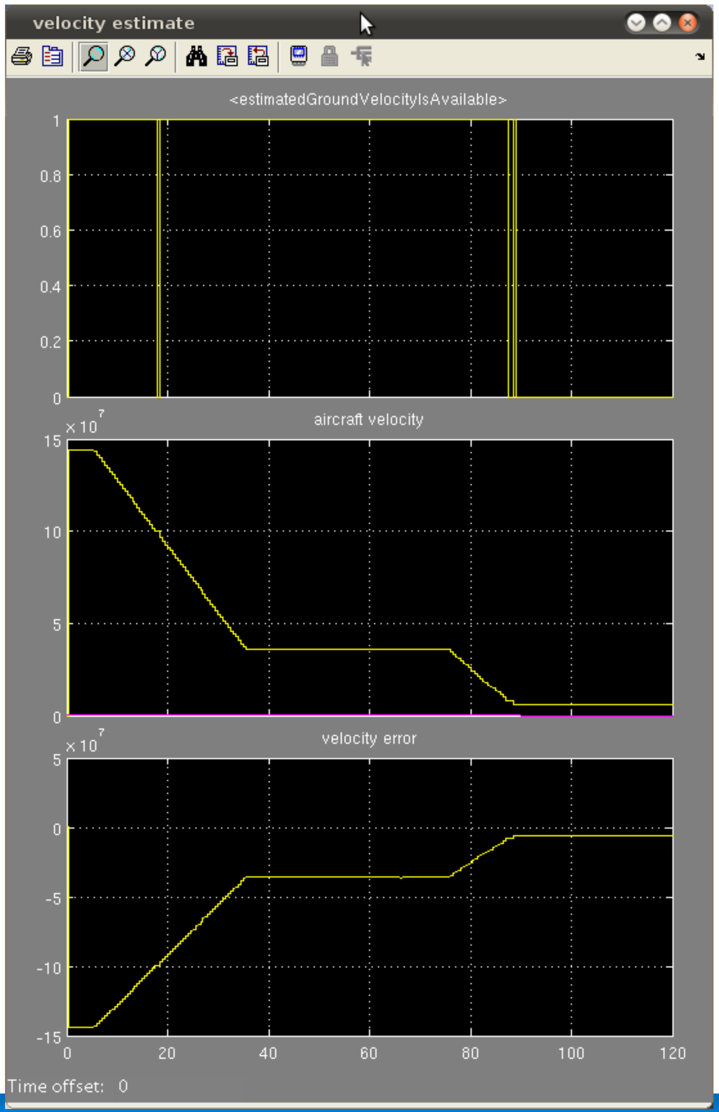


- All plots show a parameter plotted against time which runs for 120 seconds.
- <Millisecs> - clock time in millisecs, emulating int16 behaviour.
- <NGRotations> - counter, incrementing each time the wheel passes through 0. This also emulates 16 bit behaviour and if left to run long enough this would overflow/wrap
- NGClickTime - the time in millisecs at which the latest click was detected, emulating int16 behaviour
- call - Emulation of the interrupt generated by the RTOS to trigger a call to the estimation algorithm. By default this goes high at 500ms intervals but setting the workspace parameter rtos_randCallInterval to 1 will introduce some variation in this so that calls at less than 500ms intervals can be examined. Units are binary 0/1. The algorithm is called using a rising edge trigger, i.e. at the point when the signal goes from 0 to 1.

QinetiQ

www.QinetiQ.com

Quick example plots of some of the kinds of signals being generated.

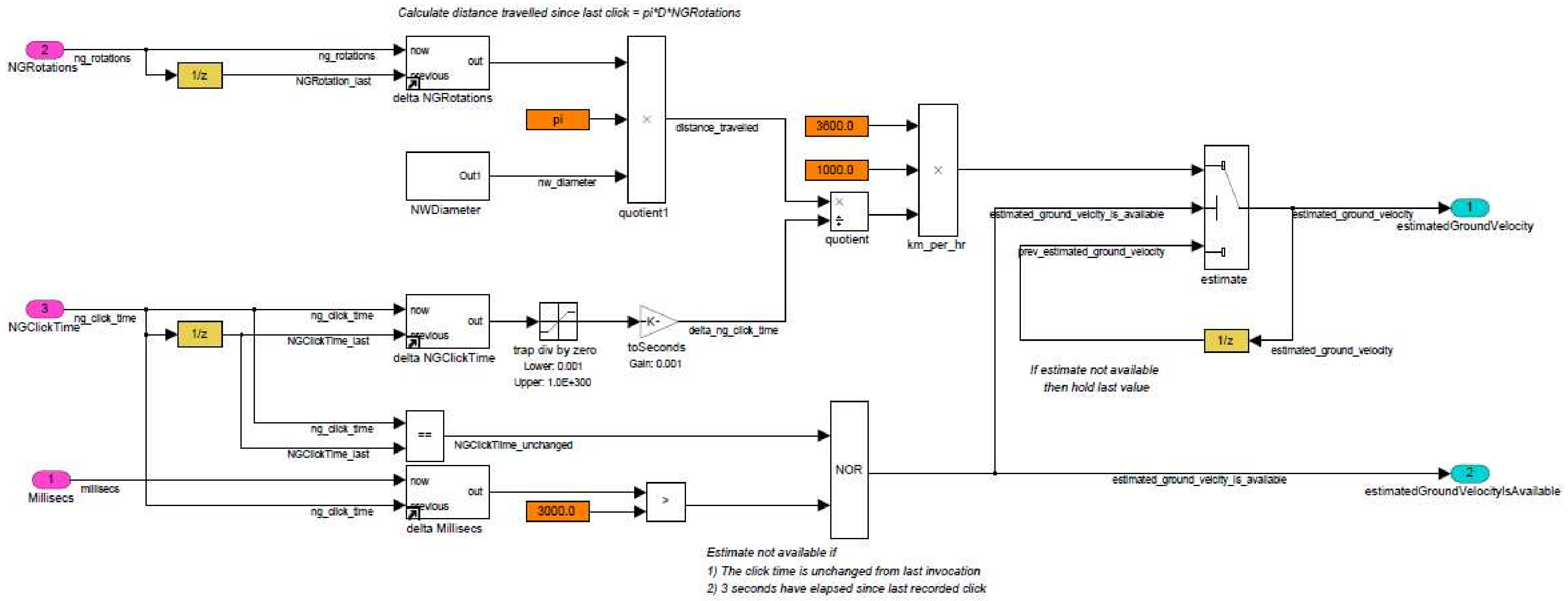


- <estimatedGroundVelocityIsAvailable> - the flag output by the algorithm to indicate validity of the estimated speed. Units are binary 0/1
- "aircraft velocity" - a composite plot showing the true velocity (magenta) and the estimate (yellow). The units are km/h.
- "velocity error" - the error between the estimate and the true value. This starts very high because within the 500ms of simulated time, the algorithm has yet to be called and the initial estimated speed is zero but the truth is 144 km/h. Perhaps should have set this chart up to take into account the validity flag because zooming in on the graph (magnifying glass and rubber banding with the mouse) shows that the flag is low during that time.

QinetiQ

www.QinetiQ.com

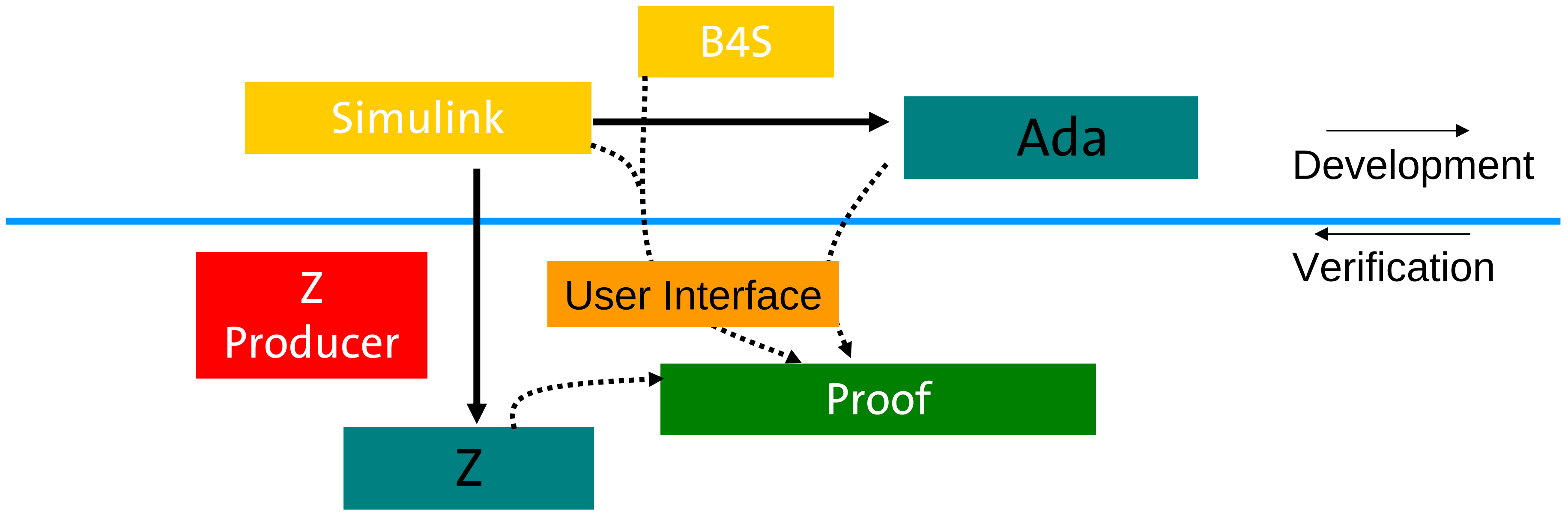
Discrete Simulink Model of Software repeated



QinetiQ

www.QinetiQ.com

Overview of the CLawZ Process



Ada Package Specification generated by B4S from the Simulink model of the Software System

```
-- BEACON Auto-Generated Code
with BEACON_Standard;
package shell is
   estimated_ground_velcity_is_available : BEACON_Standard.Extended_Float ;
   estimated_ground_velocity : BEACON_Standard.Extended_Float ;
   millisecs : BEACON_Standard.Extended_Float ;
   ng_click_time : BEACON_Standard.Extended_Float ;
   ng_rotations : BEACON_Standard.Extended_Float ;
   pi : constant BEACON_Standard.Extended_Float := 3.141592654 ;
procedure shell ( estimated_ground_velocity_is_available : out
BEACON_Standard.Extended_Float );
end shell;
```

Ada Package Body generated by B4S from the Simulink model of the Software System

```
-- BEACON Auto-Generated Code
with BEACON_Standard;
package body shell is
    shell_estimatedGroundVelocity_estimate_state :
BEACON_Standard.Extended_Float ;
    shell_estimatedGroundVelocity_NGClickTime_state :
BEACON_Standard.Extended_Float ;
    shell_estimatedGroundVelocity_NGRotation_state :
BEACON_Standard.Extended_Float ;
procedure NWDiameter ( nw_diameter : out    BEACON_Standard.Extended_Float )
is
    ft2m : constant BEACON_Standard.Extended_Float := 0.3048 ;
    wheel_diameter : constant BEACON_Standard.Extended_Float := 22.0 ;
  begin
    nw_diameter := (wheel_diameter / 12.0) * ft2m;

  end NWDiameter ;
```

QinetiQ

Ada Package Body (continued)

```
function calc_delta ( now : in      BEACON_Standard.Extended_Float ;
              previous : in     BEACON_Standard.Extended_Float )
                   return   BEACON_Standard.Extended_Float is
   tmp1 : BEACON_Standard.Extended_Float ;
   tmp2 : BEACON_Standard.Discrete_Flag ;
   delta_val : BEACON_Standard.Extended_Float ;
  begin
   tmp1 := now - (previous);
   tmp2 := (now > previous);
   if tmp2 then
     delta_val := tmp1;
   else
     tmp1 := tmp1 + 65534.0;
     delta_val := tmp1;
   end if;
   return delta_val;
  end calc_delta ;
```

QinetiQ

www.QinetiQ.com

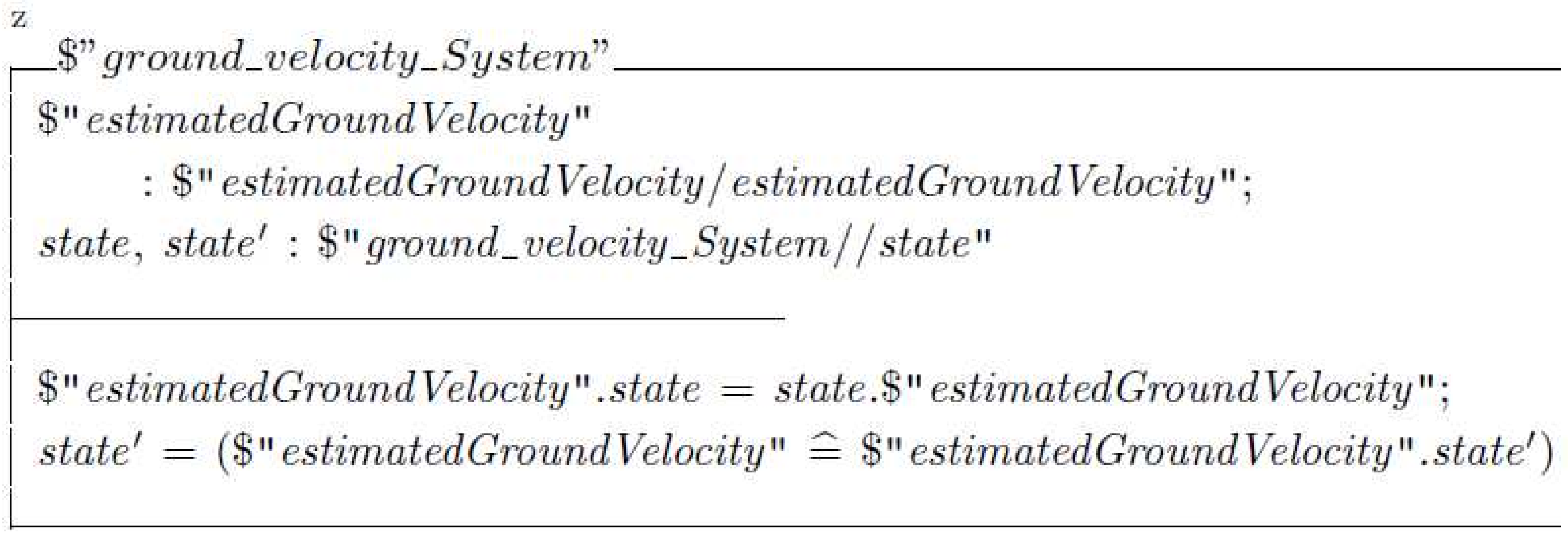
Ada Package Body (continued)

```
procedure ground_velocity ( estimate_state : in out BEACON_Standard.Extended_Float ;
                            NGClickTime_state : in out BEACON_Standard.Extended_Float ;
                           NGRotation_state : in out BEACON_Standard.Extended_Float )
is
   delta_ng_click_time : BEACON_Standard.Extended_Float ;
   distance_travelled : BEACON_Standard.Extended_Float ;
   NGClickTiime_unchanged : BEACON_Standard.Discrete_Flag ;
   NGClickTime_last : BEACON_Standard.Extended_Float ;
   NGRotation_last : BEACON_Standard.Extended_Float ;
   nw_diameter : BEACON_Standard.Extended_Float ;
   prev_estimated_ground_velocity : BEACON_Standard.Extended_Float ;
   tmp1 : BEACON_Standard.Discrete_Flag ;
   tmp2 : BEACON_Standard.Extended_Float ;
   tmp3 : BEACON_Standard.Extended_Float ;
   tmp4 : BEACON_Standard.Extended_Float ;
  begin
   NWDiameter ( nw_diameter => nw_diameter );
   prev_estimated_ground_velocity := estimate_state;
.
.
.
```

QinetiQ

www.QinetiQ.com

Fragment of Z specification automatically generated from Simulink model of Software Specification



Z

$\$\text{"}ground\_velocity\_System\text{"}$

$\$\text{"}estimatedGroundVelocity\text{"}$

$\quad : \$\text{"}estimatedGroundVelocity/estimatedGroundVelocity\text{"};$

$state,\ state' : \$\text{"}ground\_velocity\_System//state\text{"}$

$\$\text{"}estimatedGroundVelocity\text{"}.state = state.\$\text{"}estimatedGroundVelocity\text{"};$

$state' = (\$\text{"}estimatedGroundVelocity\text{"} \mathrel{\widehat{=}} \$\text{"}estimatedGroundVelocity\text{"}.state')$

QinetiQ

www.QinetiQ.com

Fragment of Z specification automatically generated from Simulink model of Software Specification (continued)

z

$\mathit{SHELLoGROUND\_VELOCITY\_Interface}_B$

$\$"\mathit{ground\_velocity\_System}";$
$\mathit{NGROTATION\_STATE} : \mathbb{R};$
$\mathit{NGCLICKTIME\_STATE} : \mathbb{R};$
$\mathit{ESTIMATE\_STATE} : \mathbb{R};$
$\mathit{NGROTATION\_STATE}_0 : \mathbb{R};$
$\mathit{NGCLICKTIME\_STATE}_0 : \mathbb{R};$
$\mathit{ESTIMATE\_STATE}_0 : \mathbb{R};$
$\mathit{SHELLoESTIMATED\_GROUND\_VELCITY\_IS\_AVAILABLE} : \mathbb{R};$
$\mathit{SHELLoESTIMATED\_GROUND\_VELOCITY} : \mathbb{R};$
$\mathit{SHELLoNG\_CLICK\_TIME} : \mathbb{R};$
$\mathit{SHELLoNG\_ROTATIONS} : \mathbb{R};$
$\mathit{SHELLoMILLISECS} : \mathbb{R}$

$\$"\mathit{estimatedGroundVelocity}".\mathit{In?}\ 1$
$\quad = \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{SHELLoMILLISECS} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle);$
$\$"\mathit{estimatedGroundVelocity}".\mathit{In?}\ 2$
$\quad = \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{SHELLoNG\_ROTATIONS} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle);$
$\$"\mathit{estimatedGroundVelocity}".\mathit{In?}\ 3$
$\quad = \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{SHELLoNG\_CLICK\_TIME} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle);$
$\$"\mathit{estimatedGroundVelocity}".\mathit{Out!}\ 1$
$\quad = \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{SHELLoESTIMATED\_GROUND\_VELOCITY} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle);$
$\$"\mathit{estimatedGroundVelocity}".\mathit{Out!}\ 2$
$\quad = \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{SHELLoESTIMATED\_GROUND\_VELCITY\_IS\_AVAILABLE} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle);$
state
$\quad = (\$"\mathit{estimatedGroundVelocity}" \mathrel{\widehat=}$
$\qquad (\$"\mathit{Unit\ Delay1}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{ESTIMATE\_STATE}_0 \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle),$
$\qquad\quad \$"\mathit{delay}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{NGCLICKTIME\_STATE}_0 \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle),$
$\qquad\quad \$"\mathit{delay2}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{NGROTATION\_STATE}_0 \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle)));$
state'
$\quad = (\$"\mathit{estimatedGroundVelocity}" \mathrel{\widehat=}$
$\qquad (\$"\mathit{Unit\ Delay1}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{ESTIMATE\_STATE} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle),$
$\qquad\quad \$"\mathit{delay}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{NGCLICKTIME\_STATE} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle),$
$\qquad\quad \$"\mathit{delay2}" \mathrel{\widehat=} \mathit{Array}_S\ (\mathit{data} \mathrel{\widehat=} \langle \mathit{NGROTATION\_STATE} \rangle, \mathit{size} \mathrel{\widehat=} \langle 1 \rangle)))$

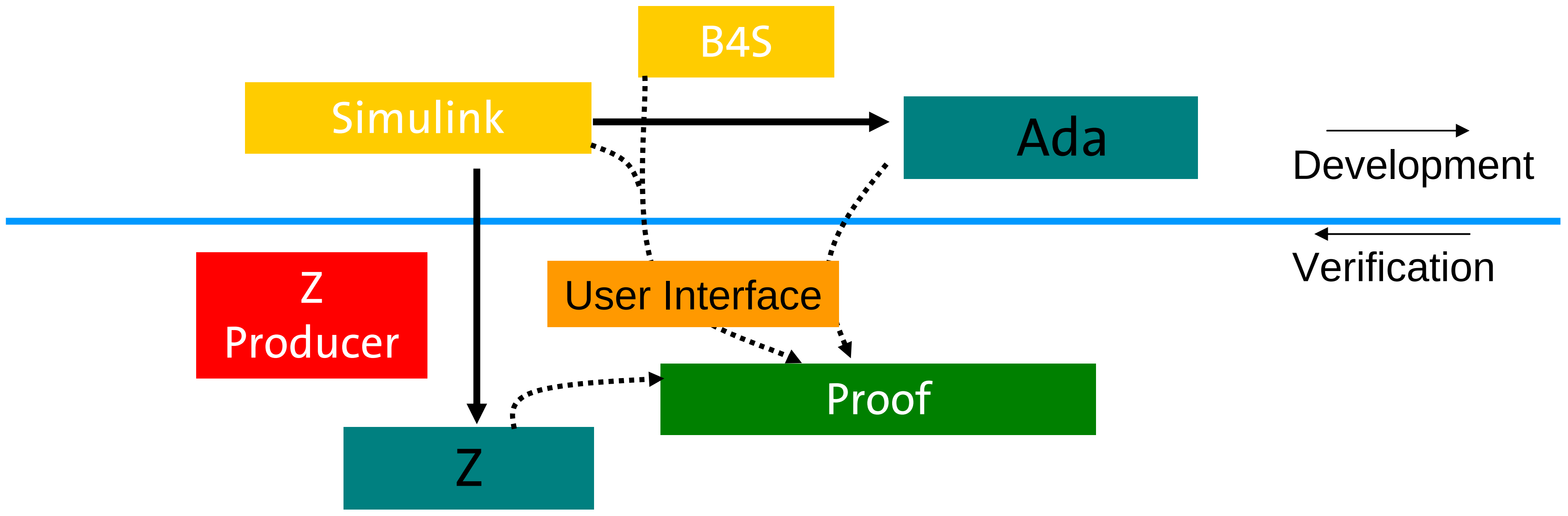
QinetiQ

Final Z abbreviation that is automatically incorporated into Specification Statement (analogous to specification statement in Morgan’s refinement calculus)

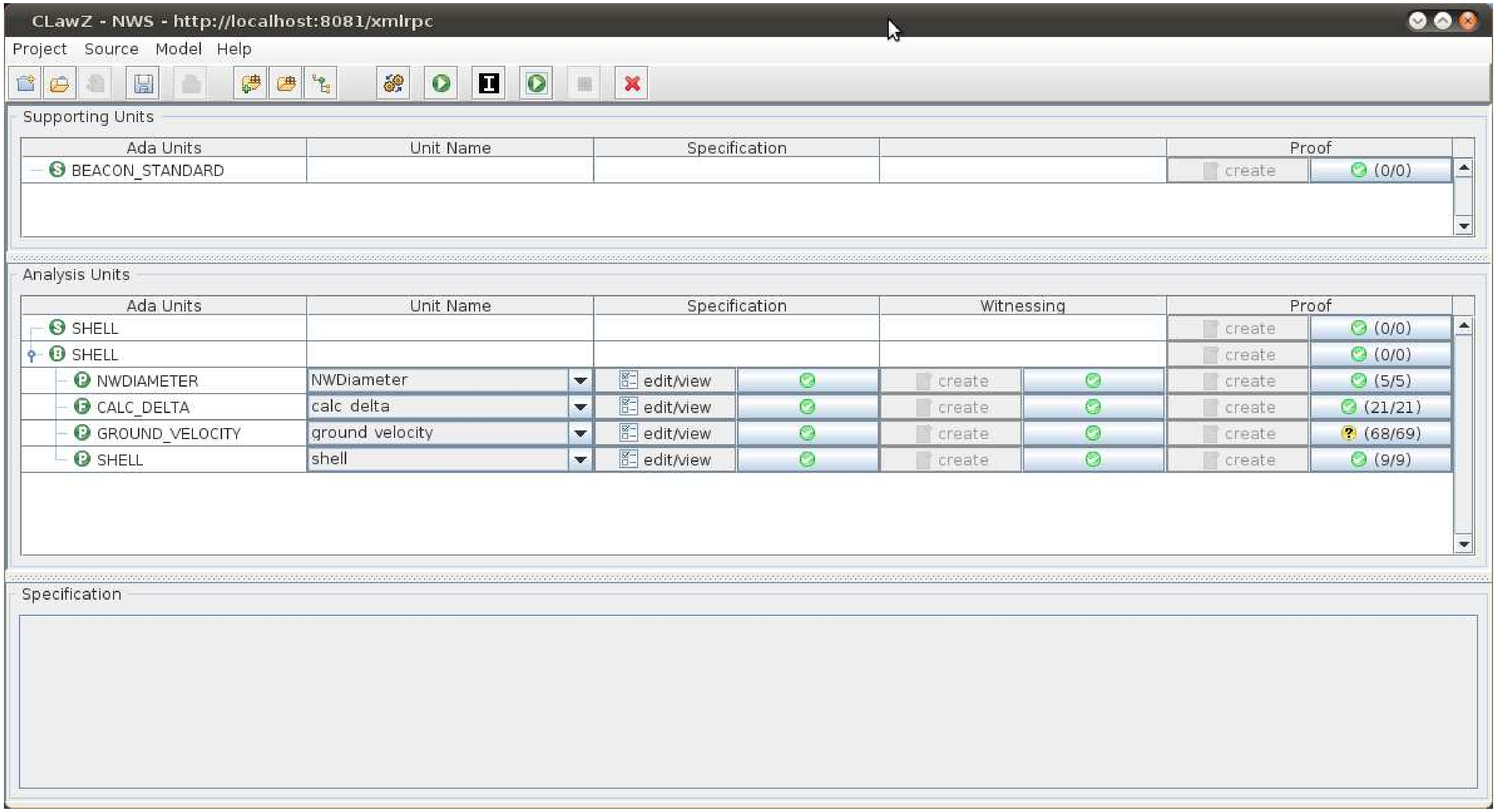
z

$$SHELLoGROUND\_VELOCITY\_Spec_B \mathrel{\widehat=}$$
$$\exists_s \$"ground\_velocity\_System" \bullet SHELLoGROUND\_VELOCITY\_Interface_B$$

Overview of the CLawZ Process again



Screenshot of CLawZ GUI


CLawZ - NWS - http://localhost:8081/xmlrpc
Project Source Model Help

Supporting Units

Ada Units	Unit Name	Specification		Proof	
BEACON_STANDARD				create	✓ (0/0)

Analysis Units

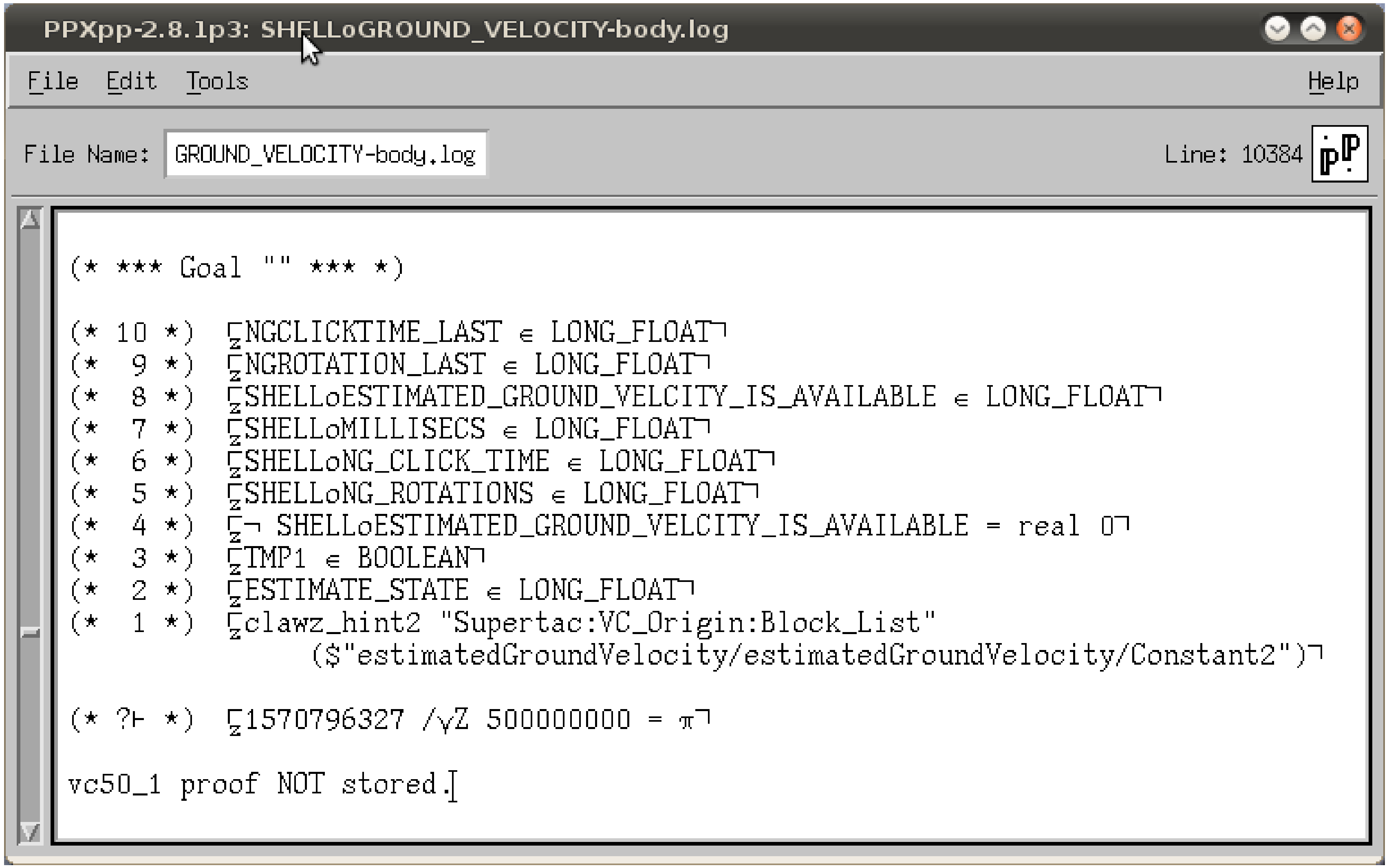
Ada Units	Unit Name	Specification		Witnessing		Proof	
SHELL						create	✓ (0/0)
SHELL						create	✓ (0/0)
NWDIAMETER	NWDiameter	edit/view	✓	create	✓	create	✓ (5/5)
CALC_DELTA	calc delta	edit/view	✓	create	✓	create	✓ (21/21)
GROUND_VELOCITY	ground velocity	edit/view	✓	create	✓	create	? (68/69)
SHELL	shell	edit/view	✓	create	✓	create	✓ (9/9)

Specification

QinetiQ

www.QinetiQ.com

Single Unproven Verification Condition


PPXpp-2.8.1p3: SHELLoGROUND_VELOCITY-body.log
File Edit Tools Help
File Name: GROUND_VELOCITY-body.log
Line: 10384

```
(* *** Goal "" *** *)

(* 10 *)  ⌜NGCLICKTIME_LAST ∈ LONG_FLOAT⌝
(*  9 *)  ⌜NGROTATION_LAST ∈ LONG_FLOAT⌝
(*  8 *)  ⌜SHELLoESTIMATED_GROUND_VELCITY_IS_AVAILABLE ∈ LONG_FLOAT⌝
(*  7 *)  ⌜SHELLoMILLISECS ∈ LONG_FLOAT⌝
(*  6 *)  ⌜SHELLoNG_CLICK_TIME ∈ LONG_FLOAT⌝
(*  5 *)  ⌜SHELLoNG_ROTATIONS ∈ LONG_FLOAT⌝
(*  4 *)  ⌜¬ SHELLoESTIMATED_GROUND_VELCITY_IS_AVAILABLE = real 0⌝
(*  3 *)  ⌜TMP1 ∈ BOOLEAN⌝
(*  2 *)  ⌜ESTIMATE_STATE ∈ LONG_FLOAT⌝
(*  1 *)  ⌜clawz_hint2 "Supertac:VC_Origin:Block_List"
              ($"estimatedGroundVelocity/estimatedGroundVelocity/Constant2")⌝

(* ?⊢ *)  ⌜1570796327 /⍼Z 500000000 = π⌝

vc50_1 proof NOT stored.
```

Conclusions

- The continuous model and scheduler allow us to define and evolve a discrete model of the software system until we are confident that it is good enough to deploy Formal Proof.
- From the Simulation models we can define
  - a predicate over a continuous domain including time
  - a scheduling predicate
  - and from these two models and the requirements a predicate over the discrete domain
- The CLawZ “Z producer” automatically generates the formal representation of the discrete Simulink model in Z.
- ADI’s B4S auto-coder independently generates an Ada implementation of the same Simulink model.
- CLawZ automatically verifies the functional behaviour Ada against the Z representation by proof.
- Exception analysis (e.g. QinetiQ Malporte tool) and WCET analysis is required to discharge underlying assumptions.

Conclusions

- Metrics
  - 1 day to define first cut of Simulink models.
  - 1 day to modify discrete model to:
    - comply with CLawZ and B4S supported subsets;
    - automatically generate the predicate representing the discrete Simulink model;
    - automatically generate the Ada using B4S;
    - automatically verify the code.
- In a position to collaborate with other organisations with the capability to reason about predicates over continuous time to prove the property
  - We bring a capability to quickly evolve a validated predicate describing the software and to quickly prove that the software is correct in a manner commensurate with DO-178B.

Comments on Challenge Problem

- *The requirements specification is not precise, e.g., what does "accurate" mean?*
  - The pseudo-code reflects some thinking on this by setting maximum allowable errors on acceleration and "jerk" (rate of change of acceleration).
- *What about when the aircraft is being pushed backwards – is this a negative velocity?*
  - From the description of the system it is not clear that we would know. The sensor raises a click when the wheel passes a particular point. As far as we know there is no encoder counting up or down in a particular direction. The click simply indicates a single position but in itself doesn't allow direction to be deduced.
- *The finite size of the data values (i.e., 16 bits) introduces some complications that would make the conventional verification by means of testing more challenging.*
  - The pseudo-code reflects consideration of these aspects. The current area of most concern in the simple first cut of the Simulink model is the test for "staleness" (i.e. has more than 3 seconds elapsed since the last click) and how to tackle the case where the clock resets (overflow/wraps whatever the right term is) during that window.

Comments on Challenge Problem

- The initial simplicity of the problem belies the potential for some considered thinking.
  - Tyre slip and skidding. Not sure if nose wheels ever have a braking system (usually part of the main gear) but it is possible for the aircraft to still be moving but the rotational information to suggest otherwise because of the wheel becoming locked or skidding. Introduces possible need for integration with other on-board systems to cross check (e.g. Inertial Navigation System)
  - Also the wheel may be rotating before it is in contact with the ground (the model allows for this with the variable nw_omega0) so, as the pseudo-code mentions, some form of weight on wheels signal may be needed. In fact, we believe that in large aircraft the wheels are spun up to reduce skidding and flat spotting of the tyre on contact with the ground.
  - Type compression. The effective radius of the tyre will reduce as the load it carries increases. This will impact on estimated speed. Again, is a type pressure signal needed to estimate compression? Or, turning it around, do we use a range of possible radii to bracket the estimates and provide a range or measurement of accuracy.
  - Slight pedantry. The example refers to estimated <u>velocity</u> which to most engineers implies a vector quantity. In fact the current scheme is measuring a scalar <u>speed</u>.
  - With this in mind, the question is raised of what is actually being measured? The nose wheel on a large aircraft sits well ahead of the centre of gravity. During ground taxiing and especially when turning the local translational velocity/speed of the wheel can be quite different to that of the Centre of Gravity . (This can be an issue for Inertial Navigation System systems where placement in the airframe may require corrections to be made when calculating Centre of Gravity velocity due to moment-arm issues). Again, introduction of an additional sensor (nose wheel steering angle) or cross correlation with similar systems on the main gear may be needed to resolve the Centre of Gravity's translational velocity.