Formal Development of Multi-Purpose Interactive Application (MPIA) for ARINC 661

Dominique Méry[2]
Joint work with
Neeraj Kumar Singh[1], Yamine Aït Ameur[1], David Navarre[3], Philippe Palanque[3] and Marc Pantel[1]

[1]INPT-ENSEEIHT / IRIT University of Toulouse, France
[2]LORIA, Telecom Nancy Université de Lorraine, France
[3]IRIT, Université de Toulouse, Toulouse, France

January 14, 2020

IFIP WG 1.3

Outline

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Current Summary

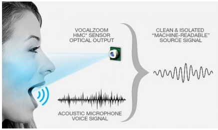
1 Context and Problems
2 Correctness by Construction
3 Event-B
4 First steps in HMI using Event-B
Using refinement-based methodology
MPIA in Event-B
5 Second step in HMI using Pivot Modelling Language
6 FLUID
7 Modelling Framework
ICO and PetShop CASE Tools
8 MPIA Case Study
9 Development of MPIA
MPIA in FLUID
MPIA FLUID Model in Event-B
MPIA FLUID Model in PetShop
10 Discussion
11 Conclusion and Future Work

Human Machine Interface (HMI)

What is Human Machine Interface?

A human-machine interface (HMI) is a device or software that allows its user to communicate with systems.

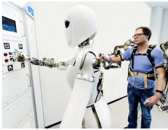
Types of HMI



Acoustic & Optics


Bionics


Tactile

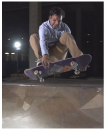
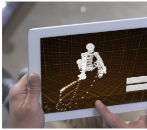

Motion

Applications of HMI


Flight Cockpit


Medical


Industry



Games and Robotics

Context and Problems

Context

Development of a rigorous formal framework for modelling and designing Human Machine Interface (HMI) complying with ARINC 661 specification standard using a correct-by-construction approach.

Problems

- Increasing complexity in HMIs;
- Lack of abstraction or of formal design patterns for handling different aspects interactive systems;
- Needs better techniques & tools to manage interactions, time, task analysis, domain properties, scenarios and concurrency in HMIs;
- Needs some sound techniques to meet certification standards related to HMIs.

Objectives (1)

- Development of a new pivot language, FLUID (Formal Language of User Interface Design), for modelling and designing a complex HMI;
- Formal development of a complex HMI using a correct by construction approach;
- Formal verification and validation of requirements, scenarios, tasks, interactions and safety properties of HMI;
- Integration of several techniques and tools in a single modelling framework for developing HMIs;
- To demonstrate the use of proposed framework to an industrial case study;
- Use of formal proofs and animations as an evidence in HMI certification.

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Correctness by Construction

- Correctness by Construction is a method of building software -based systems with **demonstrable correctness** for security- and safety-critical applications.
- Correctness by Construction advocates a **step-wise refinement** process from specification to code using tools for checking and transforming models.
- Correctness by Construction is an approach to software/system construction
  - starting with an abstract model of the problem.
  - progressively adding details in a step-wise and checked fashion.
  - each step guarantees and proves the correctness of the new concrete model with respect to requirements

The Cleanroom Method as CbC

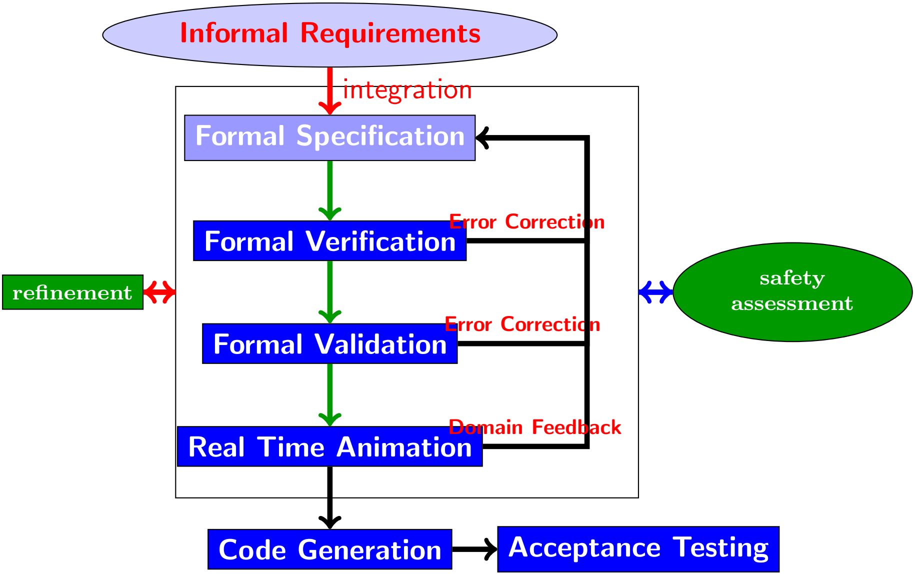
- The **Cleanroom** method, developed by Harlan Mills and his colleagues at IBM and elsewhere, attempts to do for software what cleanroom fabrication does for semiconductors: to achieve quality by keeping defects out during fabrication.
- In semiconductors, **dirt** or **dust** that is allowed to **contaminate** a chip as it is being made cannot possibly be removed later.
- But we try to do the equivalent when we write programs that are full of bugs, and then attempt to remove them all using debugging.

The Cleanroom Method as CbC

The Cleanroom method, then, uses a number of techniques to develop software carefully, in a well-controlled way, so as to avoid or eliminate as many defects as possible before the software is ever executed. Elements of the method are:

- specification of all components of the software at all levels;
- stepwise refinement using constructs called "box structures";
- verification of all components by the development team;
- statistical quality control by independent certification testing;
- no unit testing, no execution at all prior to certification testing.

Critical System Development Life-Cycle Methodology



Overview of Methodology

- Informal Requirements: Restricted form of natural language.
- Formal Specification: Modeling language like Event-B , Z, ASM, VDM, TLA+. . .
- Formal Verification: Theorem Prover Tools like PVS, Z3, SAT, SMT Solver. . .
- Formal Validation: Model Checker Tools like ProB, UPPAAL , SPIN, SMV . . .
- Real-time Animation: **Real-Time Animator . . .**
- Code Generation: **EB2ALL: EB2C, EB2C++, EB2J, EB2C# . . .**
- Acceptance Testing: Failure Mode, Effects and Critically analysis(FMEA and FMEA), System Hazard Analyses(SHA)

Previous Topics

- *Colin Boyd and Anish Mathuria. Protocols Authentication and Key Establisment. Springer 2003.*
- *C. C. Marquezan and L. Z. Granville. Self-* and P2P for Network Management - Design Principles and Case Studies. Springer Briefs in Computer Science. Springer, 2012.*
- *Pacemaker Challenge Contribution*

Case Studies

Previous Topics

- *Colin Boyd and Anish Mathuria. Protocols Authentication and Key Establisment. Springer 2003.*
- *C. C. Marquezan and L. Z. Granville. Self-* and P2P for Network Management - Design Principles and Case Studies. Springer Briefs in Computer Science. Springer, 2012.*
- *Pacemaker Challenge Contribution*

Human-Machine Interface

- *Cruise Controller*
- *Multi-Purpose Interactive Application*

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Event-B

- **Contexts** describe static properties of underlying logical and mathematical structures (i.e. graphs)

- **Machines** describe invariants for variables modified by events

$$
\begin{tikzcd}
M_0 \arrow[r, "SEES"] & C_0 \\
M_1 \arrow[u, "REFINES"] \arrow[r, "SEES"] & C_1 \arrow[u, "EXTENDS"'] \\
\ldots \arrow[u, "REFINES"] \arrow[r, "SEES"] & \ldots \arrow[u, "EXTENDS"'] \\
M_n \arrow[u, "REFINES"] \arrow[r, "SEES"] & C_n \arrow[u, "EXTENDS"']
\end{tikzcd}
$$

Event B Structure and Proofs

CONTEXT	MACHINE
ctxt_id_2	*machine_id_2*
EXTENDS	**REFINES**
ctxt_id_1	*machine_id_1*
SETS	**SEES**
s	*ctxt_id_2*
CONSTANTS	**VARIABLES**
c	v
AXIOMS	**INVARIANTS**
$A(s,c)$	$I(s,c,v)$
THEOREMS	**THEOREMS**
$T_c(s,c)$	$T_m(s,c,v)$
END	**VARIANT**
	$V(s,c,v)$
	EVENTS
	Event $evt?$
	any x
	where $G(s,c,v,x)$
	then
	$v :\mid BA(s,c,v,x,v')$
	end
	END

Table: Model structure

Invariant preservation	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\wedge BA(s,c,v,x,v')$ $\Rightarrow I(s,c,v')$
Event feasibility	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\Rightarrow \exists v'.BA(s,c,v,x,v')$
Variant modelling progress	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\wedge BA(s,c,v,x,v')$ $\Rightarrow V(s,c,v') < V(s,c,v)$
Theorems	$A(s,c) \Rightarrow T_c(s,c)$ $A(s,c) \wedge I(s,c,v)$ $\Rightarrow T_m(s,c,v)$

Table: Selected proof obligations

Refinement of states and/or events

- Adding/refinement of state variables
- Adding new events by refining the skip event (stuttering)
- Refinement of events (reduction of underspecification) by guard strengthening and event simulation

Rodin Tools

- Safety and type invariants, and deadlock freeness
- Availability of powerful provers (i.e. SMTs) in the Rodin platform
- Availability of model checkers and model animators (i.e. ProB)
- Availability of code generators (i.e. EB2ALL)

MACHINE
  m
SEES
  c
VARIABLES
  x
INVARIANT
  $I(x)$
THEOREMS
  $Q(x)$
INITIALISATION
  $Init(x)$
EVENTS
  $\ldots e$
END

Modelling systems in Event-B

MACHINE
  m
SEES
  c
VARIABLES
  x
INVARIANT
  $I(x)$
THEOREMS
  $Q(x)$
INITIALISATION
  $Init(x)$
EVENTS
  $\ldots e$
END

c defines the static environment for the proofs related to m: sets, constants, axioms, theorems $\Gamma(m)$.

MACHINE
    m
SEES
    c
VARIABLES
    x
INVARIANT
    $I(x)$
THEOREMS
    $Q(x)$
INITIALISATION
    $Init(x)$
EVENTS
    $\ldots e$
END

c defines the static environment for the proofs related to m: sets, constants, axioms, theorems $\Gamma(m)$.

$\Gamma(m) \vdash \forall x \in Values : \textsc{Init}(x) \Rightarrow \mathrm{I}(x)$

MACHINE
m
SEES
c
VARIABLES
x
INVARIANT
$I(x)$
THEOREMS
$Q(x)$
INITIALISATION
$Init(x)$
EVENTS
$\ldots e$
END

c defines the static environment for the proofs related to m: sets, constants, axioms, theorems $\Gamma(m)$.

$$\Gamma(m) \vdash \forall x \in Values : \textsc{Init}(x) \Rightarrow \mathrm{I}(x)$$

$\forall e$:

$$\Gamma(m) \vdash \forall x, x', u \in Values : \mathrm{I}(x) \wedge R(u, x, x') \Rightarrow \mathrm{I}(x')$$

MACHINE
m
SEES
c
VARIABLES
x
INVARIANT
$I(x)$
THEOREMS
$Q(x)$
INITIALISATION
$Init(x)$
EVENTS
$\ldots e$
END

c defines the static environment for the proofs related to m: sets, constants, axioms, theorems $\Gamma(m)$.

$\Gamma(m) \vdash \forall x \in Values : \textsc{Init}(x) \Rightarrow \mathrm{I}(x)$

$\forall e :$
$\Gamma(m) \vdash \forall x, x', u \in Values : \mathrm{I}(x) \wedge R(u, x, x') \Rightarrow \mathrm{I}(x')$

$\Gamma(m) \vdash \forall x \in Values : \mathrm{I}(x) \Rightarrow \mathrm{Q}(x)$

MACHINE
m
SEES
c
VARIABLES
x
INVARIANT
$I(x)$
THEOREMS
$Q(x)$
INITIALISATION
$Init(x)$
EVENTS
$\ldots e$
END

c defines the static environment for the proofs related to m: sets, constants, axioms, theorems $\Gamma(m)$.

$\Gamma(m) \vdash \forall x \in Values : \textsc{Init}(x) \Rightarrow \mathrm{I}(x)$

$\forall e :$
$\Gamma(m) \vdash \forall x, x', u \in Values : \mathrm{I}(x) \wedge R(u, x, x') \Rightarrow \mathrm{I}(x')$

$\Gamma(m) \vdash \forall x \in Values : \mathrm{I}(x) \Rightarrow \mathrm{Q}(x)$

e
ANY
u
WHERE
$G(x, u)$
THEN
$x : \vert (R(u, x, x'))$
END

or e is observed $x \xrightarrow{e} x'$

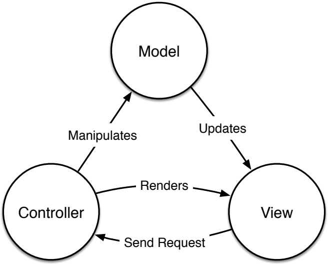
Current Summary

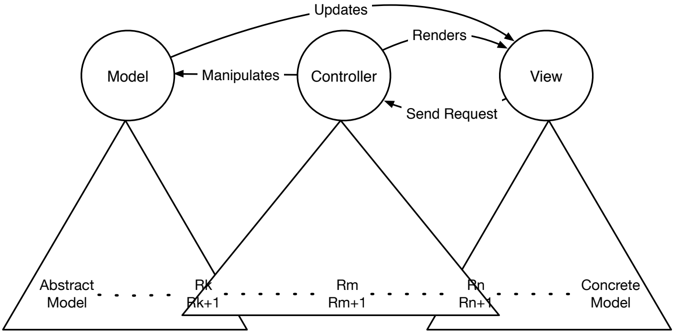
1 Context and Problems
2 Correctness by Construction
3 Event-B
4 First steps in HMI using Event-B
 Using refinement-based methodology
 MPIA in Event-B
5 Second step in HMI using Pivot Modelling Language
6 FLUID
7 Modelling Framework
 ICO and PetShop CASE Tools
8 MPIA Case Study
9 Development of MPIA
 MPIA in FLUID
 MPIA FLUID Model in Event-B
 MPIA FLUID Model in PetShop
10 Discussion
11 Conclusion and Future Work

Current Summary

1 Context and Problems
2 Correctness by Construction
3 Event-B
4 First steps in HMI using Event-B
  - Using refinement-based methodology
  - MPIA in Event-B
5 Second step in HMI using Pivot Modelling Language
6 FLUID
7 Modelling Framework
  - ICO and PetShop CASE Tools
8 MPIA Case Study
9 Development of MPIA
  - MPIA in FLUID
  - MPIA FLUID Model in Event-B
  - MPIA FLUID Model in PetShop
10 Discussion
11 Conclusion and Future Work

MVC-based Methodology





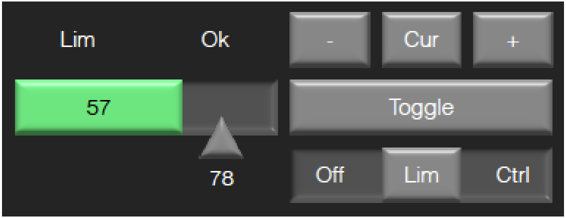
Summary on the MVC-based Methodology

- each component of the MVC is defined progressively using refinement.
- the classical scheme of MVC with possible interaction protocol and refinements for each MVC component: in the figure, each triangle represents possible refinements corresponding to the MVC components.
- Such refinement strategy allows us to analyse and reasoning a complex behaviour of an interactive system under the given constraints.
- Initially, an interactive system can be defined abstractly and then it can be refined by introducing more concrete behaviour using new state variables, events and properties.

Cruise Controller Interface

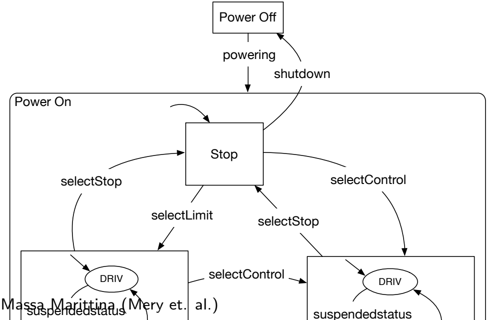
A set of informal requirements of HMI is defined as:

- **REQ1:** *the selected speed is bounded;*
- **REQ2:** *the current speed is bounded;*
- **REQ3:** *only one button can be pressed at a time;*
- **REQ4:** *the slider can be moved only if no button is pressed;*
- **REQ5:** *the default mode of HMI is stopped;*
- **REQ6:** *the limit mode and control mode can be active;*



Automaton

When the system is in the *stop* mode then it can switch either in the *limit* mode or in the *control* mode. There are several possible interactions defined in this abstract automaton to describe the model of HMI. In the context of the initial model, we define three enumerated sets: *MODES* - a set of different controller modes; *POWERED* - on and off power states; and *STATUS* - driving status and suspended status.



Steps

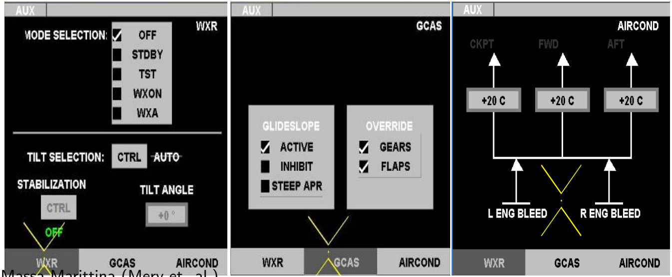
- first, formalize the *model* component, which describes a very high level of abstraction of an interactive system in form of system modality.
- Each refinement step introduces system level modality related to subsystem for analysing the required safety properties and for guaranteeing the correctness of modes transitions of an interactive system.
- second, introduce the *controller* component and the required controller behaviour.
- third, after introducing the model and controller components in the developing interactive system, introduce the view component: all visual and graphical elements, such as buttons, radio buttons, labels, of an interactive system.

Current Summary

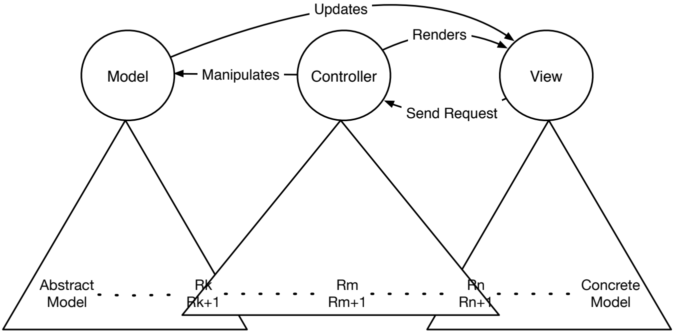
1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

MPIA (from left to right: WXR, GCAS and AIRCOND)

- the Multi-Purpose Interactive Application (MPIA) complies with ARINC 661 standard
- MPIA is a real User Application (UA) for handling several flight parameters.
- This application contains a tabbed panel with three tabs:
  - WXR for managing weather radar information
  - GCAS for Ground Collision Avoidance System parameters
  - AIRCOND for dealing with air conditioning settings.
- A crew member is allowed to switch in any mode.



Applying MVC-based Methodology



M as Model

- M describes possible interactions.
- M declares three enumerated sets, WXR_MODE_SELC_SET - a set of mode selection from a radio option widget, WXR_MODE_SELC_SET - auto or manual mode for tilt selection, and WXR_STAB_SELC_SET - on or off mode for stabilisation, to specify the MPIA components in axioms ($axm1$-$axm3$).

$$
\boxed{\begin{array}{l}
axm1 : partition(WXR\_MODE\_SELC\_SET, \{M\_OFF\}, \\
\qquad \{STDBY\}, \{TST\}, \{WXON\}, \{WXA\}) \\
axm2 : partition(WXR\_TILT\_SELC\_SET, \{AUTO\}, \\
\qquad \{MANUAL\}) \\
axm3 : partition(WXR\_STAB\_SELC\_SET, \{ON\}, \{OFF\}))
\end{array}}
$$

M as Model

- The machine model specifies the dynamic properties of MPIA.
- Three variables *ModeSelection*, *TiltSelection* and *Stabilization*:
    - the variable $ModeSelection$ represents the current selected mode from a widget (radio) given in the workspace area of WXR,
    - the variable $TiltSelection$ presents the current tilt selection mode (AUTO or MANUAL)
    - the variable $Stabilization$ indicates the current stabilization mode (ON or OFF).
- Introduce a safety property $saf1$ to state that if the tilt selection is in MANUAL mode then the stabilization can turn *on* or turn *off* and the AUTO mode of the tilt selection does not alllow to change the stabilization mode.

M as Model

$$
\begin{array}{l}
inv1 : ModeSelection \in WXR\_MODE\_SELC\_SET \\
inv2 : TiltSelection \in WXR\_TILT\_SELC\_SET \\
inv3 : Stabilization \in WXR\_STAB\_SELC\_SET \\
saf1 : TiltSelection = MANUAL \Rightarrow \\
\qquad Stabilization = ON \vee Stabilization = OFF
\end{array}
$$

Five events: $WXR\_modeSelection$ - to select the current mode from a widget (radio) of WXR; $TiltControl\_Manual$ - to switch in manual mode; $TiltControl\_Auto$ - to switch in auto mode; $Stabilization\_On$ - to select the stabilization mode (On); and $Stabilization\_Off$ - to select the stabilization mode (Off).

C as Controller

- to introduce controller components of MPIA.
- to define a constant WXR_ANGL_RANG to represent a range of tilt angles.

$$axm1 : WXR\_ANGL\_RANG = -15 \,..\, 15$$

- to introduce a new variable $TAngle$ to modify/update the tilt angle in MANUAL mode.
- A new safety property ($saf1$) is added to guarantee that the tilt angle $TAngle$ is always within the range of -15 to +15 in the MANUAL mode whenever it is modified.

$$inv : TAngle \in WXR\_ANGL\_RANG$$
$$saf1 : TiltSelection = MANUAL \Rightarrow TAngle \geq -15 \land TAngle \leq 15$$

EVENT LowTiltAngle
ANY $angl$
WHERE
grd1 : $TiltSelection = MANUAL$
grd2 : $angl \in \mathbb{Z} \land angl < -15$
THEN
act1 : $TAngle := -15$
END

V as View (1)

- to define an enumerated set WXR_BUTTONS in $axm1$ for tilt control and stabilization control buttons.

$$\begin{array}{l} axm1 : partition(WXR\_BUTTONS, \{TILT\_CTRL\}, \\ \qquad \{STAB\_CTRL\}) \end{array}$$

- To introduce two new variables $RadioBox$ and $BAction$ to describe the functional behaviour of option (radio) button, and tilt and stabilization buttons, respectively.
- The variable $RadioBox$ is defined as a total function that maps the set of options of radio widget to boolean to specify different (selected / not selected) states of the option widget.

$$\begin{array}{l} inv1 : RadioBox \in WXR\_MODE\_SELC\_SET \rightarrow BOOL \\ inv2 : BAction \in WXR\_BUTTONS \rightarrow BOOL \end{array}$$

V as View (2)

- Safety

$$
\begin{aligned}
saf1 : \forall m1, m2 \cdot m1 \in WXR\_MODE\_SELC\_SET \wedge \\
& m2 \in WXR\_MODE\_SELC\_SET \wedge \\
& m1 \mapsto TRUE \in RadioBox \wedge \\
& m2 \mapsto TRUE \in RadioBox \\
& \Rightarrow \\
& m1 = m2 \\
saf2 : TiltSelection = MANUAL \\
& \Leftrightarrow \\
& BAction(STAB\_CTRL) = TRUE \\
saf3 : TiltSelection = AUTO \\
& \Leftrightarrow \\
& BAction(STAB\_CTRL) = FALSE \\
saf4 : BAction(TILT\_CTRL) = TRUE
\end{aligned}
$$

EVENT WXR_modeSelection refines WXR_modeSelection
ANY $mode$
WHERE
grd1 : $mode \in WXR\_MODE\_SELC\_SET$
THEN
act1 : $ModeSelection := mode$
act2 : $RadioBox := (\{i \mapsto j | i \in WXR\_MODE\_SELC\_SET \wedge j = FALSE\} \cup \{mode \mapsto TRUE\}) \setminus \{mode \mapsto FALSE\}$
END

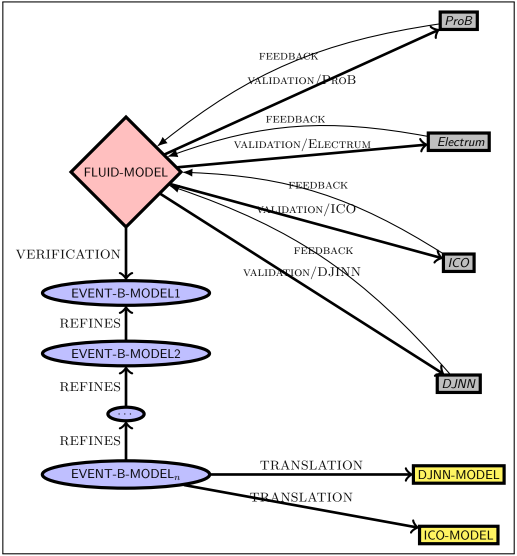
Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Objectives (2)

- Development of a new pivot language, FLUID (Formal Language of User Interface Design), for modelling and designing a complex HMI;
- Formal development of a complex HMI using a correct by construction approach;
- Formal verification and validation of requirements, scenarios, tasks, interactions and safety properties of HMI;
- Integration of several techniques and tools in a single modelling framework for developing HMIs;
- To demonstrate the use of proposed framework to an industrial case study;
- Use of formal proofs and animations as an evidence in HMI certification.

The FORMEDICIS development chain



Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
    - Using refinement-based methodology
    - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
    - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
    - MPIA in FLUID
    - MPIA FLUID Model in Event-B
    - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

FLUID (Formal Language of User Interface Design)

INTERACTION *Component_Name*
DECLARATION
SETS s
CONSTANT c
STATE
Input State Variables
Output State Variables
SysInput State Variables
SysOutput State Variables
v //A variable without $@tag$
$v@tag$ //A variables with domain specific $@tag$
EVENTS
INIT
Acquisition Events
Presentation Events
Internal Events
event $evt@tag[x]$
where
$G(s, c, v, x, v@tag, x@tag)$
then
$v :| BA(s, c, v, x, v', v@tag, x@tag, v'@tag)$
end

ASSUMPTIONS
$A(s, c)$
EXPECTATIONS
$Exp(s, c)$
REQUIREMENTS
PROPERTIES
$Prop(s, c, v, v@tag)$
SCENARIOS
NOMINAL
$SC(s, c, v, v@tag)$
NON NOMINAL
$NSC(s, c, v, v@tag)$
END *Component_Name*

FLUID (Formal Language of User Interface Design)

- ASSUMPTIONS section: introducing the required assumptions related to environment that includes the user and machine agents.
- EXPECTATIONS section: describing *prescriptive* statements that are expected to be fulfilled by parts of the environment of an interactive system.
- REQUIREMENTS section: divided into two subsections, known as PROPERTIES and SCENARIOS.
  - PROPERTIES section describes in logic all the required properties of an interactive system that must be preserved by a defined system.
  - SCENARIOS section describes both nominal and non-nominal scenarios using algebraic expressions, close to CTT, for analyzing possible acceptable and non-acceptable interactions.

FLUID Semantics (big step)

Configuration

$$\langle Elt_{Synt}, s\rangle$$
$$Env = Loc \longrightarrow Val$$
$$s = \{(l_1 \mapsto v_1), \cdots, (l_n \mapsto v_n)\}$$

Expressions

$$Exp \ \times \ Env \longrightarrow Val$$

Transitions

$$\langle Elt_{Synt}, s\rangle \implies \langle Skip \ , \ s'\rangle$$

Actions (Deterministic Assignment)

$$\frac{}{\langle x := Exp, s\rangle \implies \langle Skip, s[l_x \mapsto v]\rangle}$$

Continue...

Events (Guarded Event)

$$\frac{\langle G, s\rangle \Longrightarrow \langle True, s\rangle \qquad \langle A, s\rangle \Longrightarrow \langle Skip, s'\rangle}{\langle \textbf{where } G \textbf{ then } A, s\rangle \Longrightarrow \langle Skip, s'\rangle}$$

Interleaving Rule

$$\frac{\langle e_1, s\rangle \Longrightarrow \langle Skip, s'\rangle, \qquad \langle e_2, s'\rangle \Longrightarrow \langle Skip, s''\rangle}{\langle e_1 || e_2, s\rangle \Longrightarrow \langle Skip, s''\rangle}$$

$$\frac{\langle e_1, s'\rangle \Longrightarrow \langle Skip, s''\rangle, \qquad \langle e_2, s\rangle \Longrightarrow \langle Skip, s'\rangle}{\langle e_1 || e_2, s\rangle \Longrightarrow \langle Skip, s''\rangle}$$

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Interactive Cooperative Objects

The ICO formalism is a formal description technique for describing interactive systems using high level Petri nets. There are four main components:

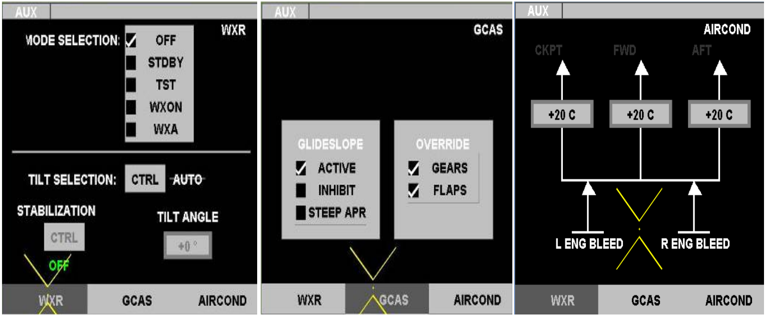
- cooperative object (describes the behavior of the object)
- presentation part (i.e. the graphical interface)
- activation function
- rendering function

PetShop tool can be used for execution and verification of ICO Models.

Current Summary

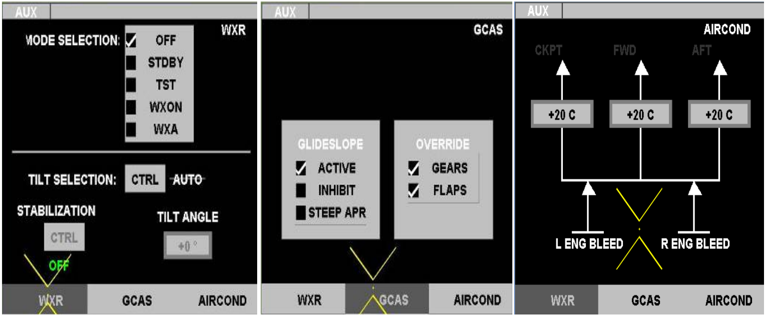
1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Multi-Purpose Interactive Application (MPIA)



Snapshots of the MPIA (a real User Application)

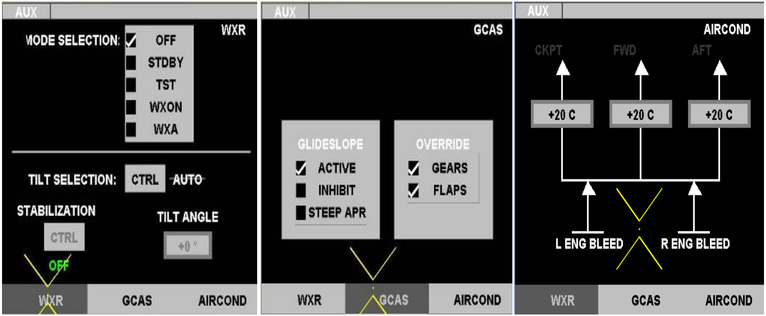
Multi-Purpose Interactive Application (MPIA)



Snapshots of the MPIA (a real User Application)

- WXR: managing weather radar informations.

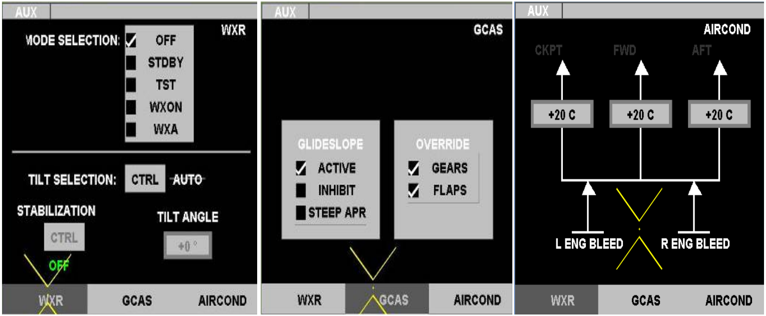
Multi-Purpose Interactive Application (MPIA)



Snapshots of the MPIA (a real User Application)

- WXR: managing weather radar informations.
- GCAS: Ground Anti Collision System parameters.

Multi-Purpose Interactive Application (MPIA)



Snapshots of the MPIA (a real User Application)

- WXR: managing weather radar informations.
- GCAS: Ground Anti Collision System parameters.
- AIRCOND: AIR CONDitioning settings.

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
    - Using refinement-based methodology
    - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
    - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
    - MPIA in FLUID
    - MPIA FLUID Model in Event-B
    - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

MPIA in FLUID:

```
// WXR Mode enumeration set
TYPE WXR_MODE_SELC_SET = enumeration (M_OFF, STDBY, TST, WXON,
// WXR Tilt and Stabilisation message enumeration set
TYPE WXR_TILT_STAB_MSG = enumeration (ON, OFF, AUTO, MANUAL)
// WXR Tilt angle range
CONSTANT WXR_ANGL_RANG = [ -15 ..  15 ]
// WRX actions
TYPE WXR_ACTIONS = enumeration (TILT_CTRL, STAB_CTRL)
```

MPIA in FLUID:

```
// Acquisition states
A_ModeSelection@{Input, Checked} :  WXR_MODE_SELC_SET // Mode st

A_TiltSelection@{Input, Enabled} :  WXR_TILT_SELC_SET // Tilt stat
...
...

// Presentation states
// Radio buttons presentation states
P_checkMode@{Output, Checked} :  WXR_MODE_SELC_SET → BOOL

// CTRL tilt button presentation state
P_ctrlModeTilt_Button@{Output, Enabled} :  WXR_ACTIONS
...
...
```

MPIA in FLUID:

// Initialisation Event

INIT =

```
A_ModeSelection := OFF
A_ModeSelection@Checked := TRUE
...
 // Only OFF mode is selected at initialisation
P_checkMode := {i ↦ j | i ∈ WXR_MODE_SELC_SET ∧
j = FALSE } ∪ { M_OFF ↦ TRUE } )\{M_OFF ↦ FALSE}
P_checkMode@Checked := TRUE
...
```

MPIA in FLUID:

```
// ACQUISITION Events
// Any mode is allowed to select from WXR to acquisition state
event modeSelection@Acquisition [mode]=
  WHERE
   mode :  WXR_MODE_SELC_SET
  THEN
   A_ModeSelection := mode
   A_ModeSelection@Checked := TRUE
  END

event tiltCtrl@Acquisition =...
event stabCtrl@Acquisition =...
event tiltAngle@Acquisition =...
event tiltAngle_Greater_15@Acquisition =...
event tiltAngle_Less_15@Acquisition =...
```

MPIA in FLUID:

```
// PRESENTATION Events
// Presentation of radio button: Only selected mode will be checked as TRU
event  checkMode@Presentation =
  WHEN
   A_ModeSelection@Checked = TRUE
  THEN
   P_checkMode:=( {i ↦ j | i ∈ WXR_MODE_SELC_SET
   ∧ j = FALSE }∪{ A_ModeSelection ↦ TRUE } )\
   {A_ModeSelection ↦ FALSE}
   P_checkMode@checked := TRUE
  END
event ctrlModeTilt_Auto@Presentation =...
event ctrlModeTilt_Manual@Presentation =...
event ctrlModeStab_On@Presentation =...
event  Event ctrlModeStab_Off@Presentation =...
event tiltAngle_True@Presentation =...
event tiltAngle_False@Presentation =...
```

MPIA in FLUID:

PROPERTIES

```
Prop1 :∀ m1,m2· m1∈ WXR_MODE_SELC_SET ∧ m2∈ WXR_MODE_SELC_SET ∧ m1↦ TRUE ∈ prj1(prj1(P_checkMode)) ∧
       m2↦ TRUE ∈ prj1(prj1(P_checkMode)) ⇒ m1=m2

Prop2 :G(e(modeSelection@Acquisition) ⇒ X (e(checkMode@Presentation) )))
...
...
```

SCENARIOS

NOMINAL

```
SC_1 = INIT; ((modeSelection@Acquisition; checkMode@Presentation)
|| (tiltCtrl@Acquisition; (ctrlModeTilt_Auto@Presentation [] ctrlModeTilt_Manual@Presentation))
|| (stabCtrl@Acquisition; (ctrlModeStab_On@Presentation [] ctrlModeStab_Off@Presentation))
|| (tiltAngle@Acquisition [] tiltAngle_Greater_15@Acquisition [] Evt_tiltAngle_Less_15@Acquisition);
(tiltAngle_True@Presentation [] Evt_tiltAngle_False@Presentation))*
```

NON NOMINAL

```
SC_1 = INIT; ((modeSelection@Acquisition; checkMode@Presentation)
||(tiltCtrl@Acquisition;ctrlModeTilt_Auto@Presentation;(stabCtrl@Acquisition[]tiltAngle@Acquisition)
```

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Translation Rules: FLUID to Event-B

INTERACTION *Component_Name*
DECLARATION
T_{decl}(*SETS s*) //Sets
T_{decl}(*CONSTANT c*) //Constant
STATE
$T_{st}(v)$ //Variable without $@tag$
$T_{st}(v@tag)$ //Variables with domain specific $@tag$
EVENTS
INIT
...
$T_{evt}($ //Events
Event $evt@tag[x]$
where
$T_{grd}(G(s,c,v,x,v@tag,x@tag))$ //Guard
then
$T_{act}((v,v@tag) : |(BA(s,c,v,x,v',v@tag,x@tag,v'@tag))$ //Action
end
$)$
ASSUMPTIONS
$T_{axm}(\ A(s,c))$ //Axioms
EXPECTATIONS
$T_{exp}(Exp(s,c))$ //Expectations
REQUIREMENTS
PROPERTIES
$T_{prop}(Prop(s,c,v,v@tag))$ //Properties
SCENARIOS
NOMINAL
$T_{sc}(SC(s,c,v,v@tag))$ //Nominal Scenarios
NON NOMINAL
$T_{nsc}(NSC(s,c,v,v@tag))$ //Non nominal Scenarios
END *Component_Name*

Table: Translation for FLUID Model

MPIA FLUID Model in Event-B

HMI TAGS

$daxm1 : partition(HMI\_TAG, \{Input\}, \{Output\}, \{SysInput\}, \{SysOutput\})$
$daxm2 : CHECKED = BOOL \land VISIBLE = BOOL \land ENABLED = BOOL$

MPIA CONTEXT

$axm1 : partition(WXR\_MODE\_SELC\_SET, \{M\_OFF\}, \{STDBY\}, \{TST\}, \{WXON\}, \{WXA\})$
$axm2 : partition(WXR\_TILT\_STAB\_MSG, \{AUTO\}, \{MANUAL\}, \{ON\}, \{OFF\})$
$axm3 : partition(WXR\_ACTIONS, \{TILT\_CTRL\}, \{STAB\_CTRL\})$
$axm4 : WXR\_ANGL\_RANG = -15 \,..\, 15$

MPIA INVARIANTS

$inv1 : A\_ModeSelection \in WXR\_MODE\_SELC\_SET \times HMI\_TAG \times CHECKED$
$inv2 : A\_TiltSelection \in WXR\_ACTIONS \times HMI\_TAG \times ENABLED$
...
...
$inv5 : P\_checkMode \in (WXR\_MODE\_SELC\_SET \rightarrow BOOL) \times HMI\_TAG \times CHECKED$
$inv6 : P\_ctrlModeTilt\_Button \in WXR\_ACTIONS \times HMI\_TAG \times ENABLED$
...
...

$saf1 : \forall m1, m2 \cdot m1 \in WXR\_MODE\_SELC\_SET \land m2 \in WXR\_MODE\_SELC\_SET \land$
$\quad m1 \mapsto TRUE \in prj1(prj1(P\_checkMode)) \land m2 \mapsto TRUE \in prj1(prj1(P\_checkMode)) \Rightarrow m1$

MPIA EVENT

EVENT INITIALISATION
BEGIN
act1 : $A\_ModeSelection := M\_OFF \mapsto Input \mapsto TRUE$
act2 : $A\_TiltSelection := TILT\_CTRL \mapsto Input \mapsto TRUE$
...
...
act6 : $P\_checkMode := ((\{i \mapsto j | i \in WXR\_MODE\_SELC\_SET \wedge j = FALSE\} \cup \{M\_OFF \mapsto TRUE\}) \setminus \{M\_OFF \mapsto FALSE\}) \mapsto Output \mapsto TRUE$
act7 : $P\_ctrlModeTilt\_Button := TILT\_CTRL \mapsto Output \mapsto TRUE$
...
...
END

EVENT modeSelection@Acquisition
ANY $mode$
WHERE
grd1 : $mode \in WXR\_MODE\_SELC\_SET$
THEN
act1 : $A\_ModeSelection := mode \mapsto Input \mapsto TRUE$
END

MPIA FLUID Model in Event-B

EVENT tiltCtrl@Acquisition
ANY n_tilt
WHERE
grd1 : $n\_tilt \in WXR\_ACTIONS \times HMI\_TAG \times ENABLED \wedge$
$prj1(prj1(n\_tilt)) = TILT\_CTRL \wedge prj2(n\_tilt) = TRUE$
THEN
act1 : $A\_TiltSelection := n\_tilt$
END

EVENT checkMode@Presentation
ANY n_tilt
WHERE
grd1 : $prj2(A\_ModeSelection) = TRUE$
THEN
act1 : $P\_checkMode := ((\{i \mapsto j | i \in WXR\_MODE\_SELC\_SET \wedge j = FALSE\} \cup$
$\{prj1(prj1(A\_ModeSelection)) \mapsto TRUE\}) \setminus$
$\{prj1(prj1(A\_ModeSelection)) \mapsto FALSE\}) \mapsto Output \mapsto TRUE$
END

Model Validation and Analysis

ProB model checker is used to animate and to check additional properties and the deadlock freeness.

Model	Total number of POs	Automatic Proof	Interactive Proof
Event-B Model	44	41(93%)	3(7%)

Table: Proof statistics

$Prop1 : (G(e(AE\_modeSelection) => X(e(PE\_checkMode))))$

$Prop2 : (e(AE\_tiltAngle) => (e(PE\_tiltAngle\_True) or e(PE\_tiltAngle\_False)))$

$$Prop3 : \{P\_ctrlModeTilt\_Label = (AUTO|->Output)|->TRUE => P\_ctrlModeStab\_Label = (OFF|->Output)|->TRUE\}$$

$$Prop4 : \{P\_ctrlModeTilt\_Label = (MANUAL|->Output)|->TRUE => P\_ctrlModeStab\_Label = (ON|->Output)|->TRUE\}$$

$$Prop5 : \{P\_ctrlModeTilt\_Label = (AUTO|->Output)|->TRUE => P\_ctrlModeStab\_Button = (STAB\_CTRL|->Output)|->FALSE\}$$

$$Prop6 : \{P\_ctrlModeTilt\_Label = (MANUAL|->Output)|->TRUE => P\_ctrlModeStab\_Button = (STAB\_CTRL|->Output)|->TRUE\}$$

$$Prop7 : \{P\_ctrlModeTilt\_Label = (MANUAL|->Output)|->TRUE => P\_TiltAngle = (10|->Output)|->TRUE\}$$

...
...

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

MPIA FLUID Model in PetShop

```
Public interface WXR_PAGE extends ICOWidget {
// List of user events.
public enum WXR_PAGE_events {asked_off, asked_stdby, asked_wxa,
asked_wxon, asked_tst, asked_auto asked_stabilization,
asked_changeAngle}

// List of activation rendering methods.
void setWXRModeSelectEnabled(WXR_PAGE_events, List<ISubstitution>);
void setWXRTiltSelectionEnabled (WXR_PAGE_events, List<ISubstitution>);

// List of rendering methods.
void showModeSelection (IMarkingEvent anEvent);
void showTiltAngle (IMarkingEvent anEvent);
void showAuto (IMarkingEvent anEvent);
void showStab (IMarkingEvent anEvent);
}
```

Figure: Software interface of the page WXR from the user application MPIA

MPIA FLUID Model in PetShop

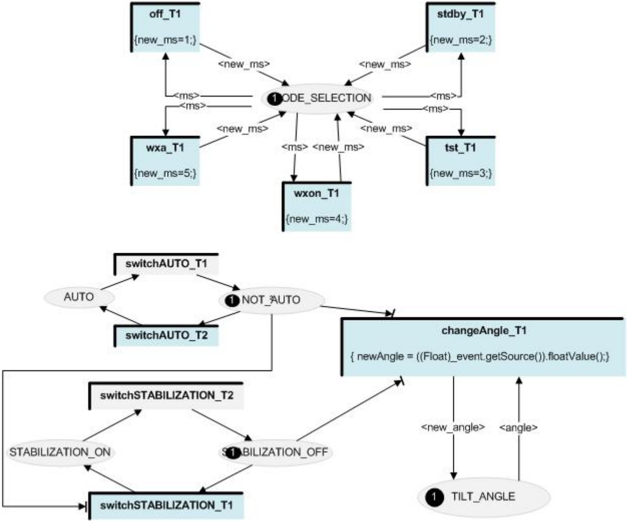


Figure: High-level Petri net model describing the behaviour of the page WXR

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Discussion

- First integrated framework for modelling and designing interactive systems.
- A pivot language FLUID for expressing:
  - Interaction behaviour
  - Properties
  - Nominal and Non-nominal Scenarios
  - Task Analysis
- Integration of several approaches (i.e. model checking, theorem prover and animators) for analysing different aspects of HMI.

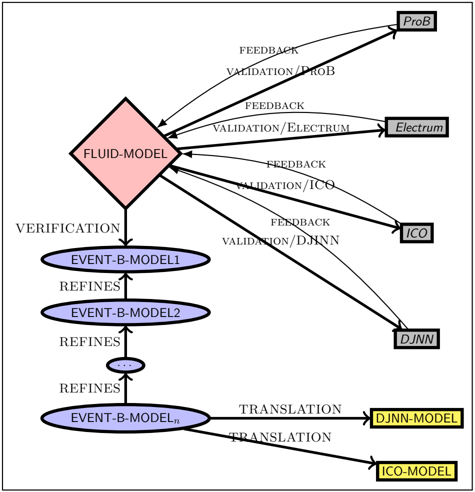


Figure: The FORMEDICIS development chain

Current Summary

1. Context and Problems
2. Correctness by Construction
3. Event-B
4. First steps in HMI using Event-B
   - Using refinement-based methodology
   - MPIA in Event-B
5. Second step in HMI using Pivot Modelling Language
6. FLUID
7. Modelling Framework
   - ICO and PetShop CASE Tools
8. MPIA Case Study
9. Development of MPIA
   - MPIA in FLUID
   - MPIA FLUID Model in Event-B
   - MPIA FLUID Model in PetShop
10. Discussion
11. Conclusion and Future Work

Conclusion

- Development of a new language, FLUID, including formal semantics for specifying and analysing HMIs.
- A new framework for developing HMIs addressed the key challenges related to interaction behaviour, properties, nominal and non-nominal scenarios.
- Introduction of domain specific HMI concepts (i.e. enabled, visible, active) using TAGS.
- Use of formal methods for designing and developing the HMIs to cover, particularly in
  - functional and perceived requirements
  - stepwise formal development
  - verification of the required safety properties
  - task analysis
  - nominal and non-nominal scenarios validation
- Demonstrate the use of our modelling language and development framework in MPIA case study.

Future Work

- Development of a set of resources (domain specific tags) for defining HMI concepts.
- Mine a methodology for,
  - Modelling, designing and analysing a bag of patterns (e.g. refinement, proof patterns, simulation, animation etc.).
  - Developing refinement strategies for FLUID language.
- Development of tools to automate the translation strategies from FLUID models to several other target modelling language (i.e djnn, ICO).
- Produce source code from formal models for implementation purpose.