Toward dependable interactive systems: dealing with system faults at development and run time

Camille Fayollas

Interactive Critical Systems research group (IRIT)

& Dependable Computing and Fault Tolerance research group (LAAS-CNRS)

http://www.irit.fr/~Camille.Fayollas - fayollas@irit.fr

January 6th 2016

HCI in Critical Context

Interactive Systems

- User Centered Design *(ISO 9241-210)*
- Usability *(ISO 9241-11)*
  - Effectiveness
  - Efficiency
  - Satisfaction

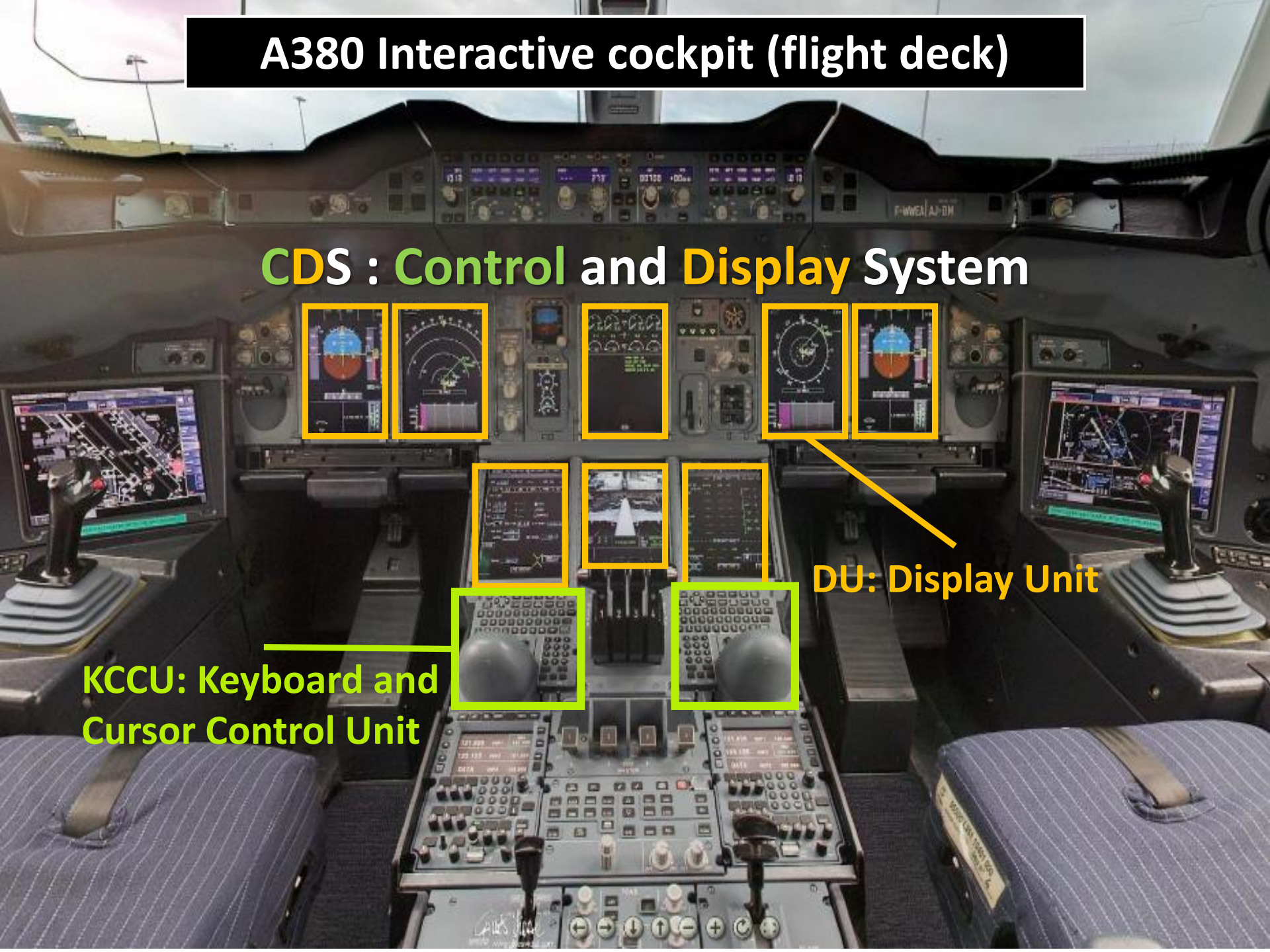
Critical Systems

- Dependable Approach
  (prevention, tolerance, removal, forecasting)
- Standards (ARP 4754), Development processes (DO-178C), Certification

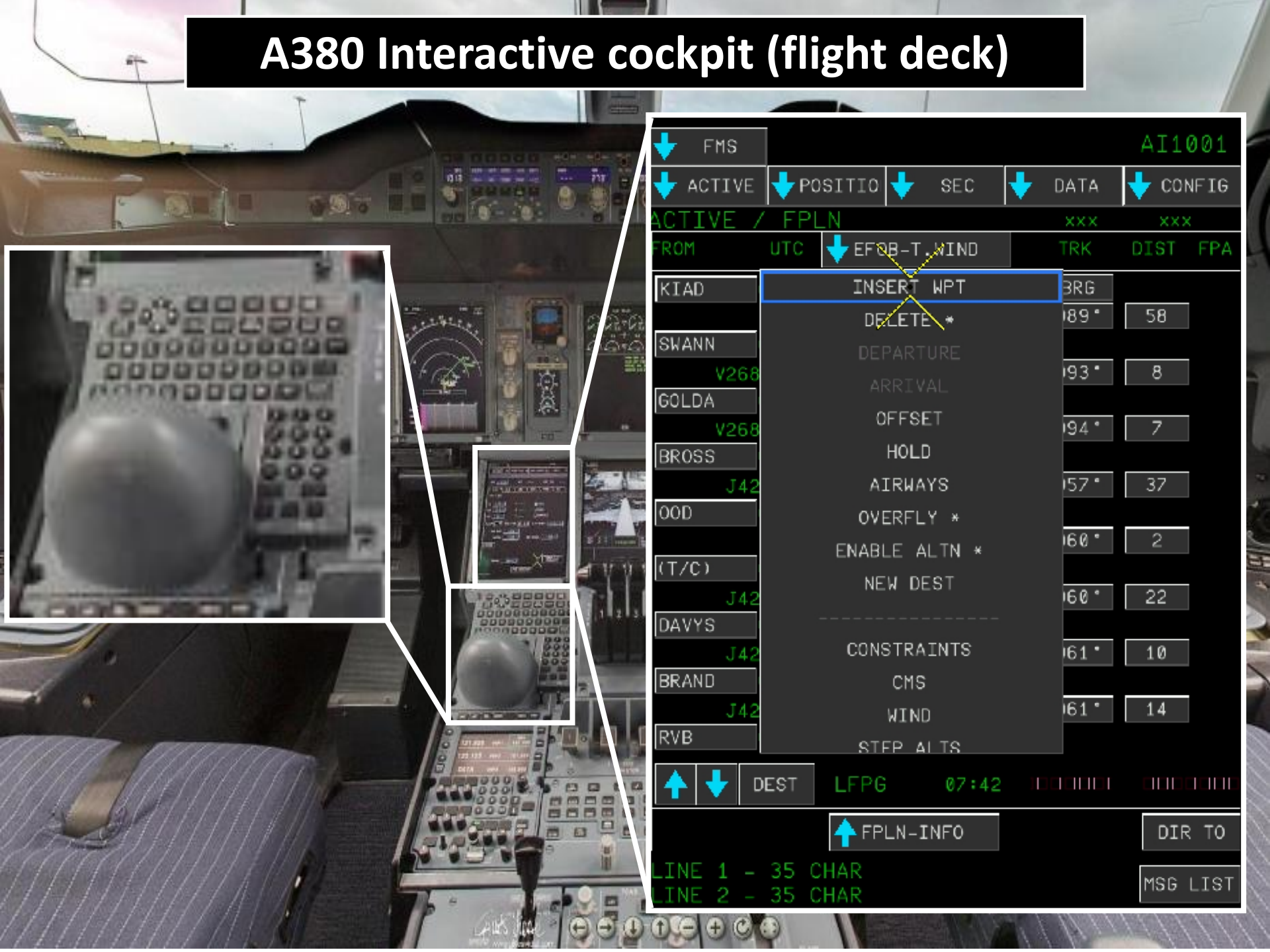
A380 Interactive cockpit (flight deck)

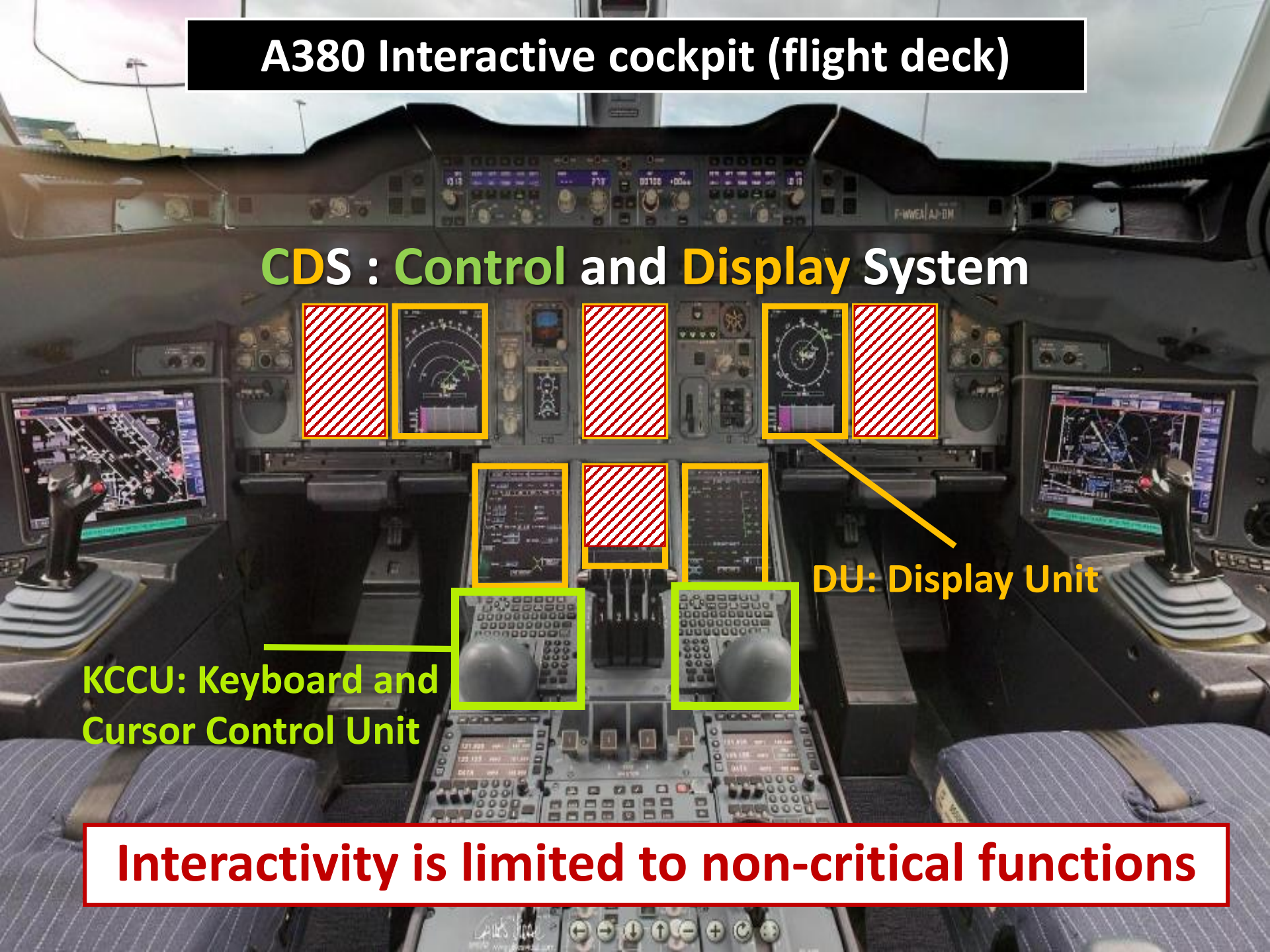
A380 Interactive cockpit (flight deck)

A380 Interactive cockpit (flight deck)



A380 Interactive cockpit (flight deck)




A380 Interactive cockpit (flight deck)
CDS : Control and Display System
DU: Display Unit
KCCU: Keyboard and Cursor Control Unit
Interactivity is limited to non-critical functions

Problem Statement

Interactivity is limited to non-critical functions

- But it has some advantages
  - Better evolvability
  - Less expensive
  - Better operation performances

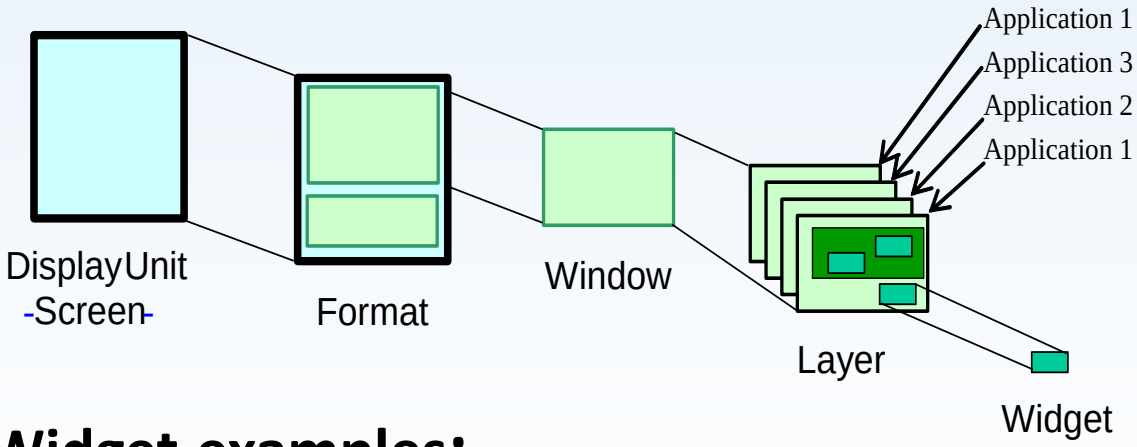
How to develop dependable interactive systems to make them amenable for the command and control of critical functions ?

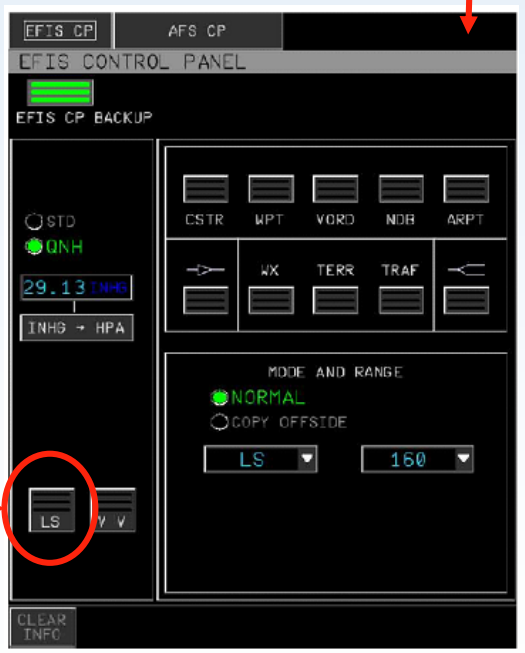
Outline of the talk

- Introduction and Problem Statement
- **Context (Interactive Cockpits)**
- Proposed Approach for Dependable Interactive Systems/Cockpits
- Case Study
- Conclusions and Perspectives

Interactive Cockpits - GUI

- Based on an avionic standard: ARINC 661
  - Standardized Graphical User Interface
  - Standardized protocol between User Applications (UA) and Cockpit Display System (CDS)



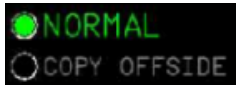


Widget examples:

PushButton

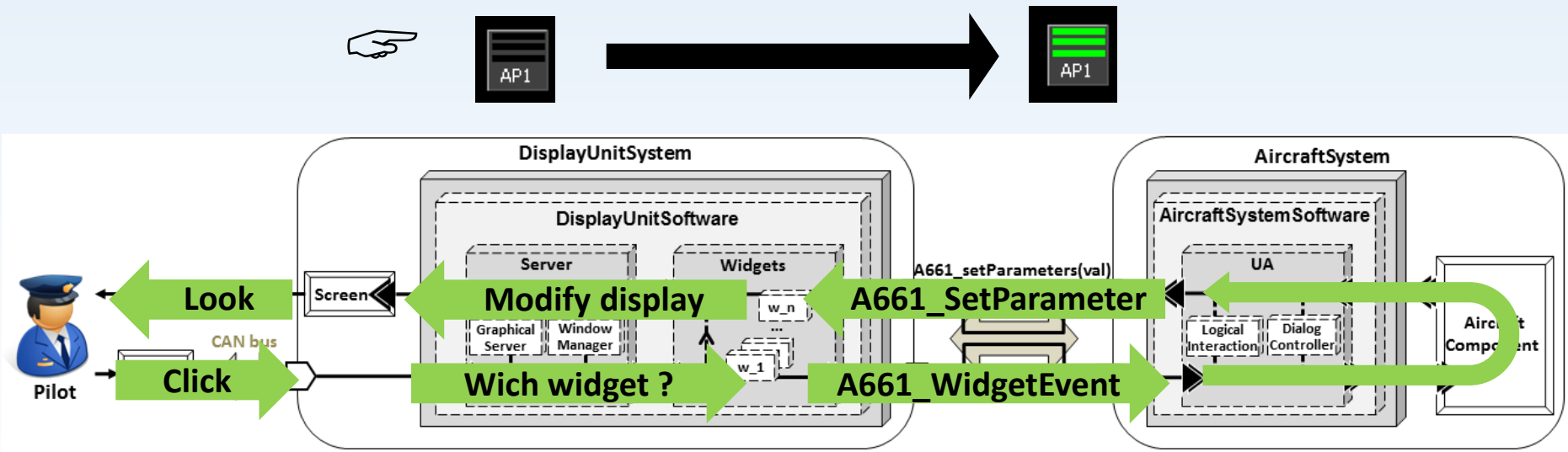
EditBoxNumeric

Radio Buttons



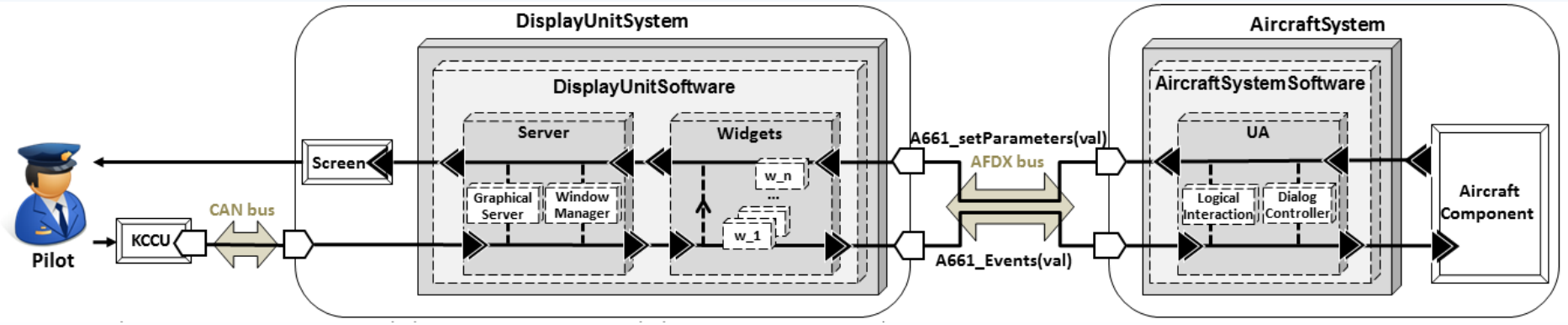
Interactive Cockpits - Functionning

Example: the engagement of the auto-pilot through a click on the corresponding PicturePushButton



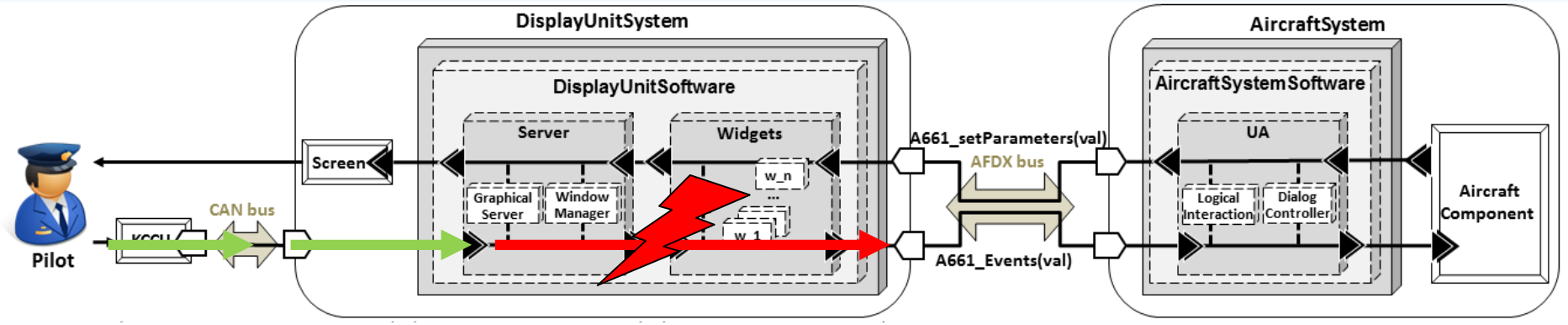
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (wrong control or spontaneous control)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (wrong display or spontaneous display)



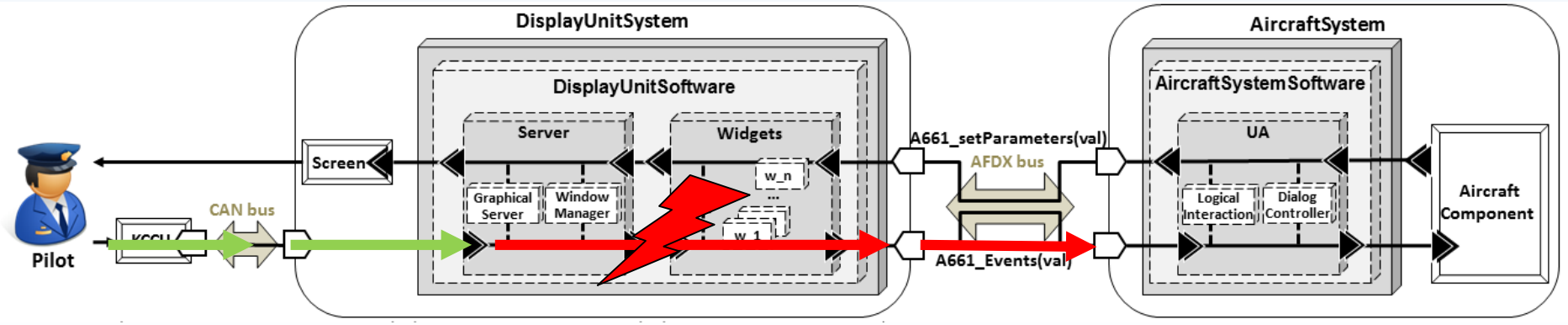
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - **Loss of control**
  - Erroneous control (wrong control or spontaneous control)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (wrong display or spontaneous display)



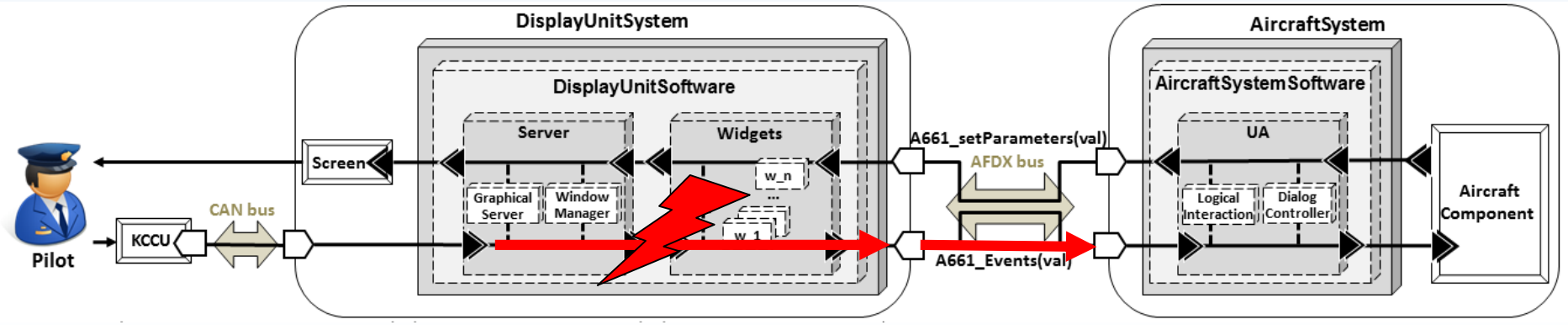
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (**wrong control** or spontaneous control)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (wrong display or spontaneous display)



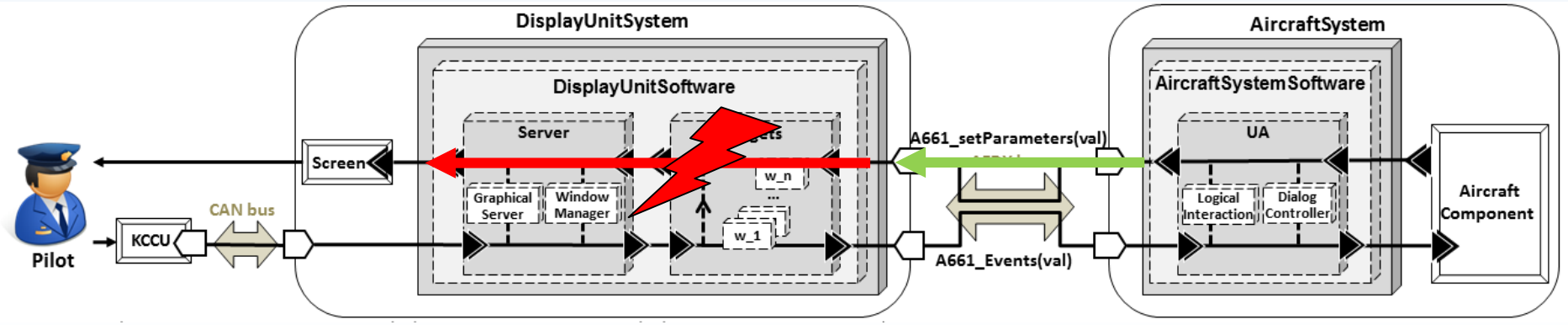
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (wrong control or **spontaneous control**)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (wrong display or spontaneous display)



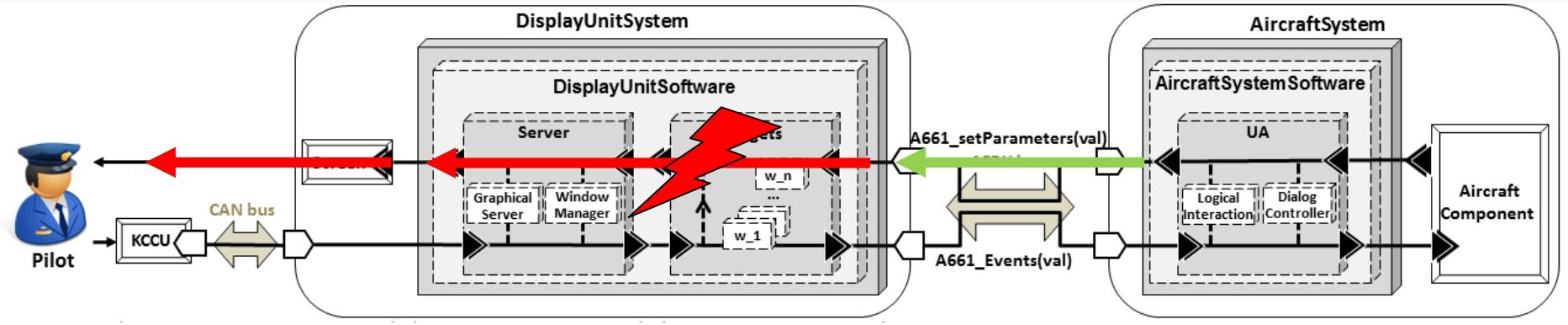
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (wrong control or spontaneous control)
- Display flow (UA -> pilot)
  - **Loss of display**
  - Erroneous display (wrong display or spontaneous display)



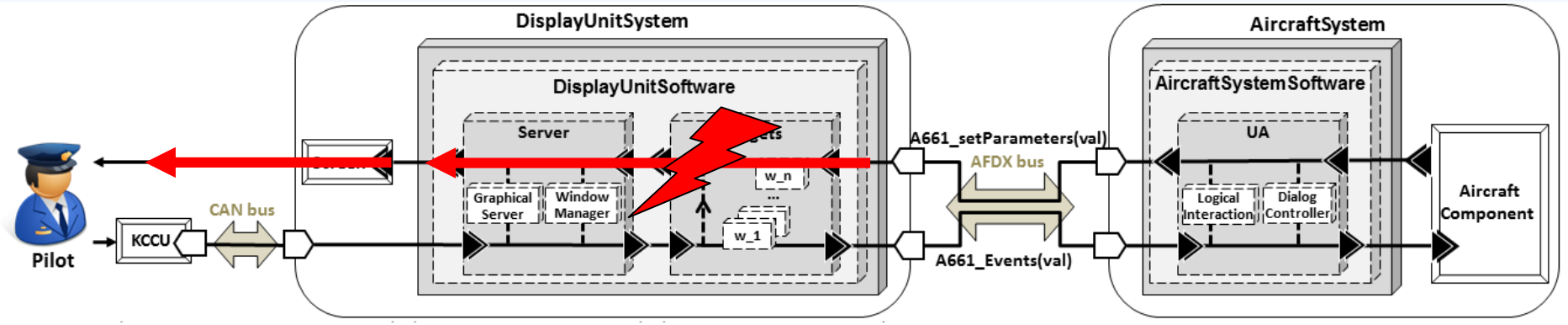
Interactive Cockpits – Failure Modes

- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (wrong control or spontaneous control)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (**wrong display** or spontaneous display)



Interactive Cockpits – Failure Modes

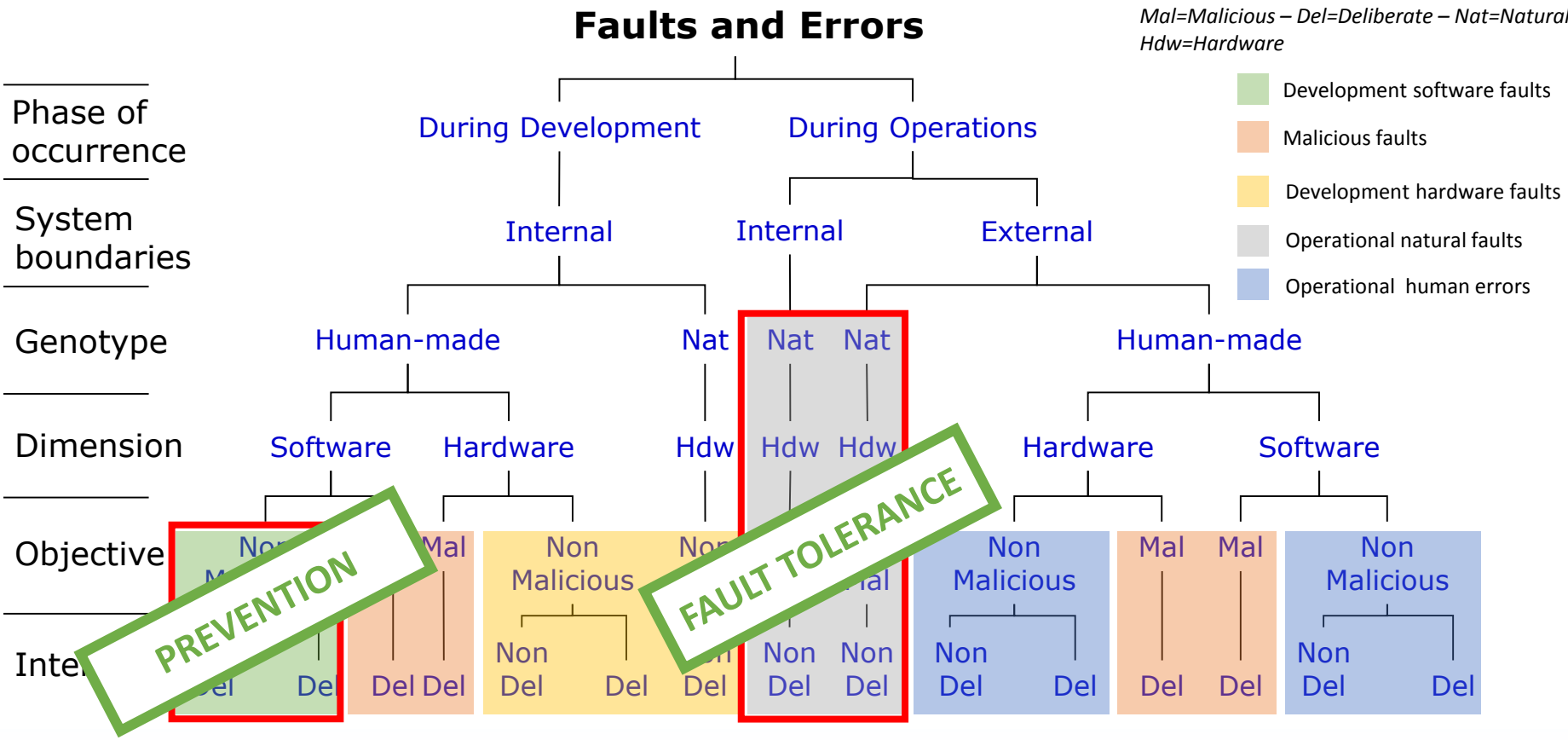
- Control flow (pilot -> UA)
  - Loss of control
  - Erroneous control (wrong control or spontaneous control)
- Display flow (UA -> pilot)
  - Loss of display
  - Erroneous display (wrong display or **spontaneous display**)



Interactive Cockpits – Fault Model

… → Fault → Error → Failure → Fault → …

activation — propagation — causation



Adapté de : *Avizienis, A., Laprie, J.-C., Randell, B., Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. In IEEE Trans. on Dependable and Secure Computing, vol.1, no.1, pp. 11- 33, Jan.-March 2004*

Outline of the talk

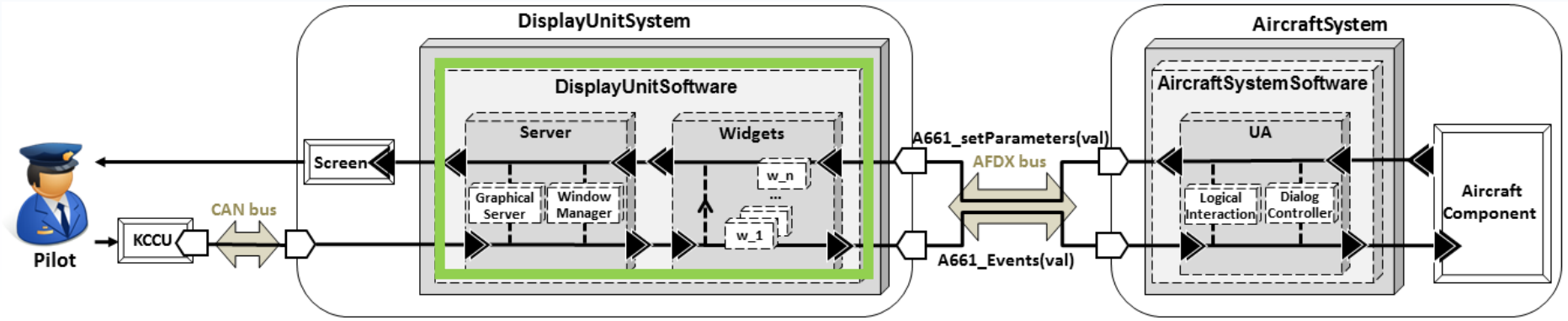
- Introduction and Problem Statement
- Context (Interactive Cockpits)
- **Proposed Approach for Dependable Interactive Systems/Cockpits**
- Case Study
- Conclusions and Perspectives

A Two-Fold Approach

- Hypothesis
  - No faults at hardware and network level
  - No human error
- Approach
  - Model-Based Approach

    => Software faults prevention
  - Process and Architecture

    => Tolerance to physical faults & to residual software faults of executive layers



Outline of the talk

- Introduction and Problem Statement
- Context (Interactive Cockpits)
- **Proposed Approach for Dependable Interactive Systems/Cockpits**
  - **Model-Based Development**
  - Process and Architecture
- Case Study
- Conclusions and Perspectives

Model-Based Approach – Principle

Prevention Approach

Zero-defect software

Use of formal notation

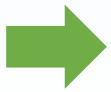
- Complete and unambiguous description
- Analysis and verification of properties

Formal Notation for Interactive Systems

- Specific needs
  - Interaction specificities
    - Covering of the interactive system architecture (server, widgets and UA)
    - Input/output management (rendering/activation)
  - Expressiveness

    (Event, state, object and their values, quantitative time…)

- Generic needs
  - Scalability
  - Usable tool



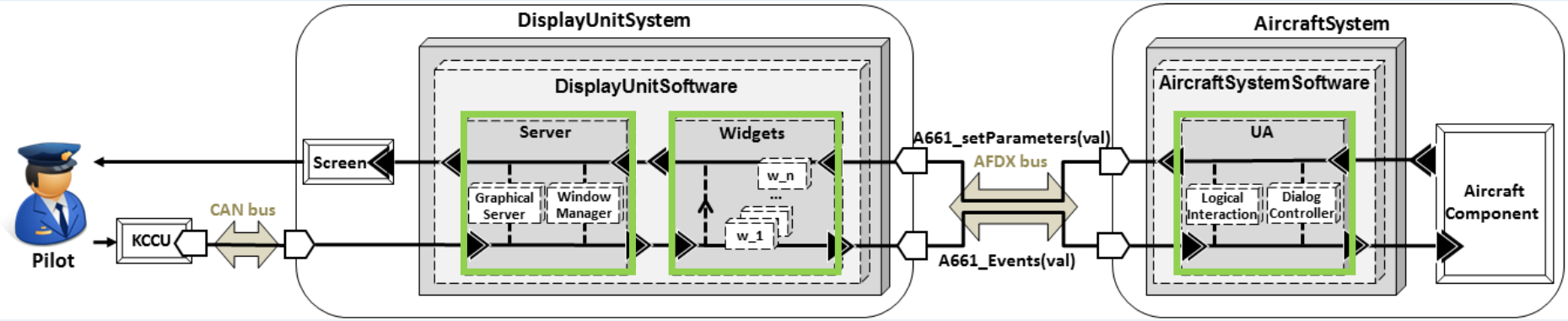
ICO

Navarre D., Palanque P., Ladry J.-F., Barboni E. ICOs: a Model-Based User Interface Description Technique dedicated to Interactive Systems Addressing Usability, Reliability and Scalability. In : ToCHI, ACM SIGCHI, Vol. 16 N. 4, p. 1-56, 2009

ICO (Interactive Cooperative Objects)

- Formal notation for interactive systems
- High-level Petri nets for behavioral description
- Expressiveness
- Tool support (PetShop)
  - Model edition
  - Analysis means
  - Models execution and simulation

ICO description of the architecture components



Server

- Widget instanciation
- SceneGraph
- Picking

Widgets

- Parameters
- Events

UA

- Application behaviour
- FCU Backup

Formalization of SceneGraph and Picking components Enables verification, validation, application of fault tolerant approach (e.g. for detecting overlapping widgets at execution)

Example : PicturePushButton description using ICO

PicturePushButton

- Presents an information
- Enables command triggering

PicturePushButton Parameters

Design time parameters

- PosX, PosY,
- SizeX, SizeY
- Etc…

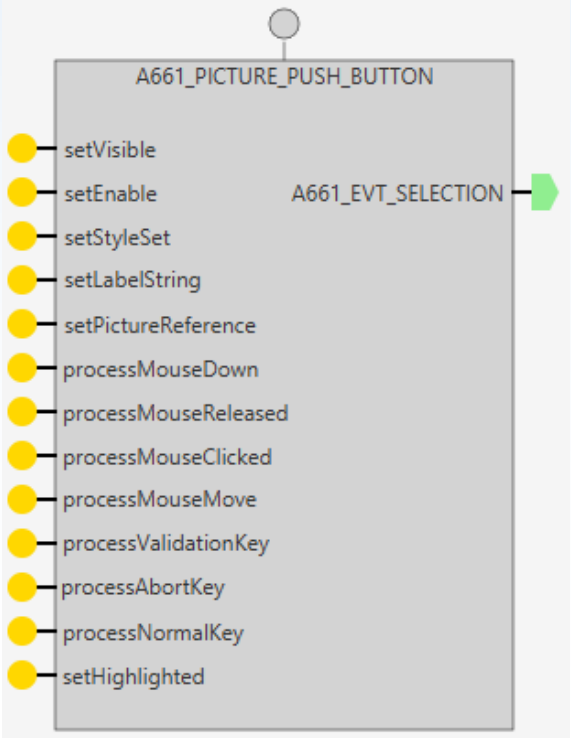
Runtime modifiable parameters

- Visible
- Enable
- Styleset
- LabelString
- PictureReference

Events

A661_evt_selection

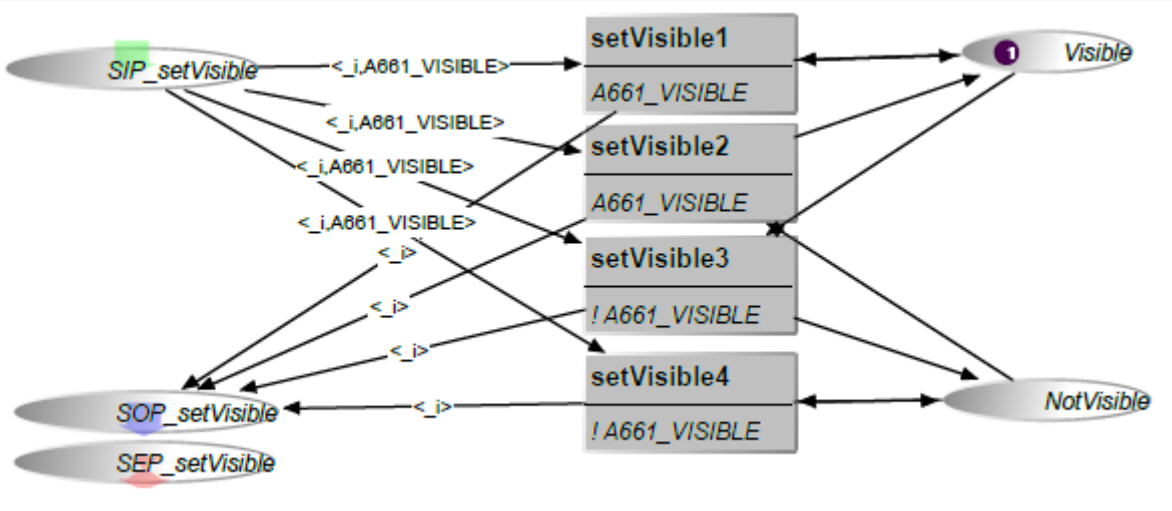
PicturePushButton Interface



Example : PicturePushButton description using ICO

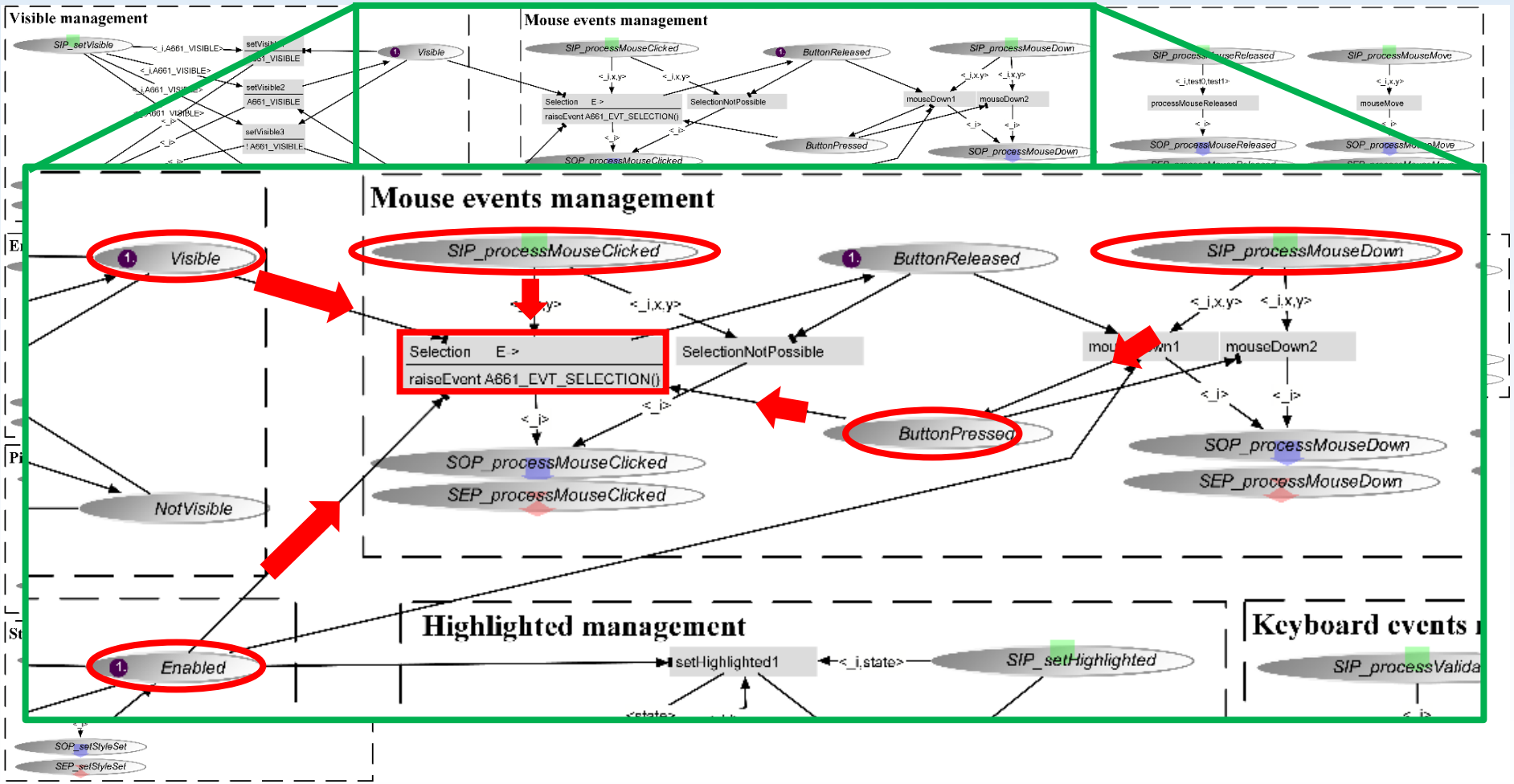
Service specification: parameter Visible enabling to change the visibility of a PicturePushButton

void setVisible(boolean A661_VISIBLE);



Example : PicturePushButton description using ICO

35 places and 20 transitions



Model-Based Approach - Summary

- Model-based approach for the specification and development of interactive systems software components
  - Use of ICO formal notation (as an example)
  - Use of PetShop for running ICO models (no additional step towards implementation)
- Complete and unambiguous behavioral description of software components
- Enable the description of each components of the architecture
- Better modelling (coverage) of the interactive system functioning
- Formal analysis of model supported

Outline of the talk

- Introduction and Problem Statement
- Context (Interactive Cockpits)
- **Proposed Approach for Dependable Interactive Systems/Cockpits**
  - Model-Based Development
  - **Process and Architecture**
- Case Study
- Conclusions and Perspectives

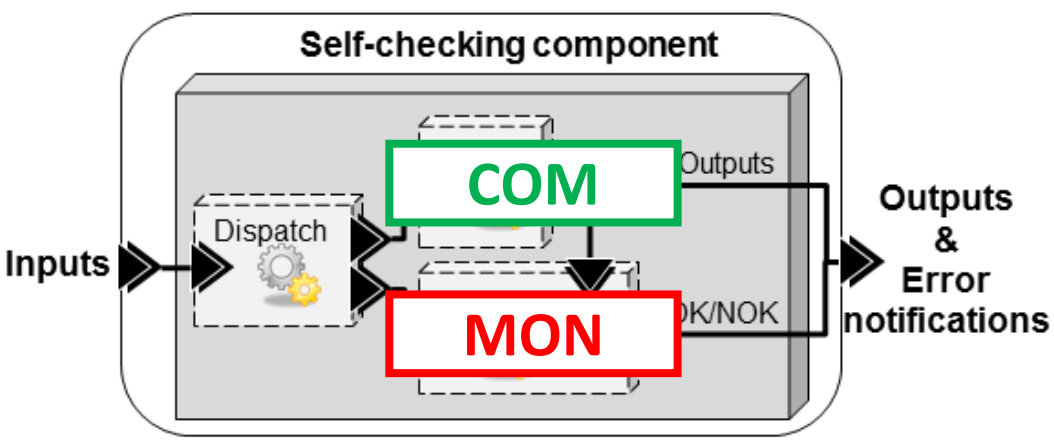
Software Architecture

- Requirements
  - Fault tolerant architecture
  - Compatible with certification requirements of avionics functions
  - Covering of all the components
  - Compatible with ARINC 661 standard

COM-MON principle

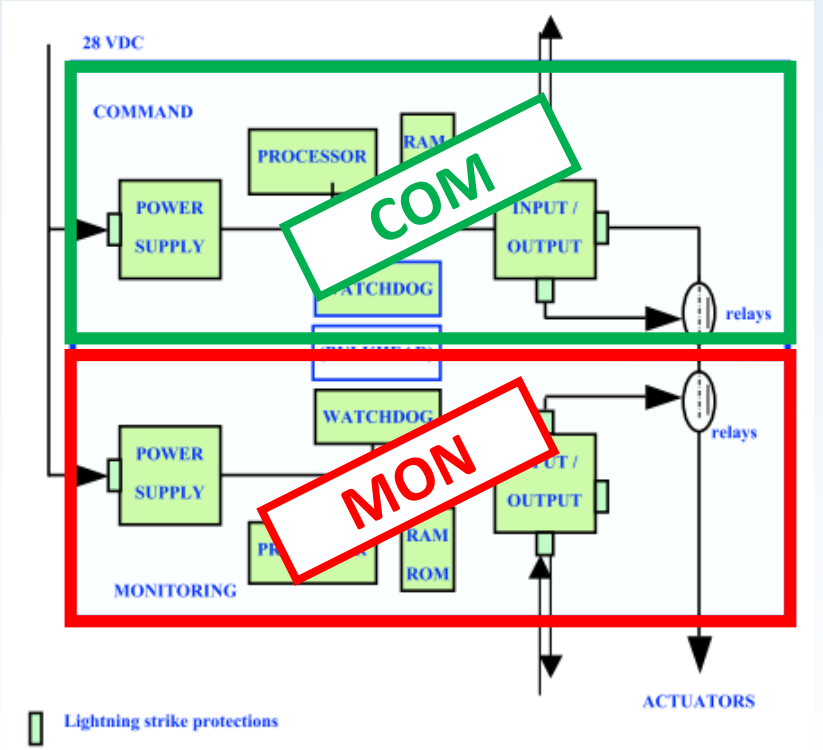
MON = set of assertion monitors

Self-Checking Component (COM-MON)



- Monitoring component definition
- Partitioning is needed

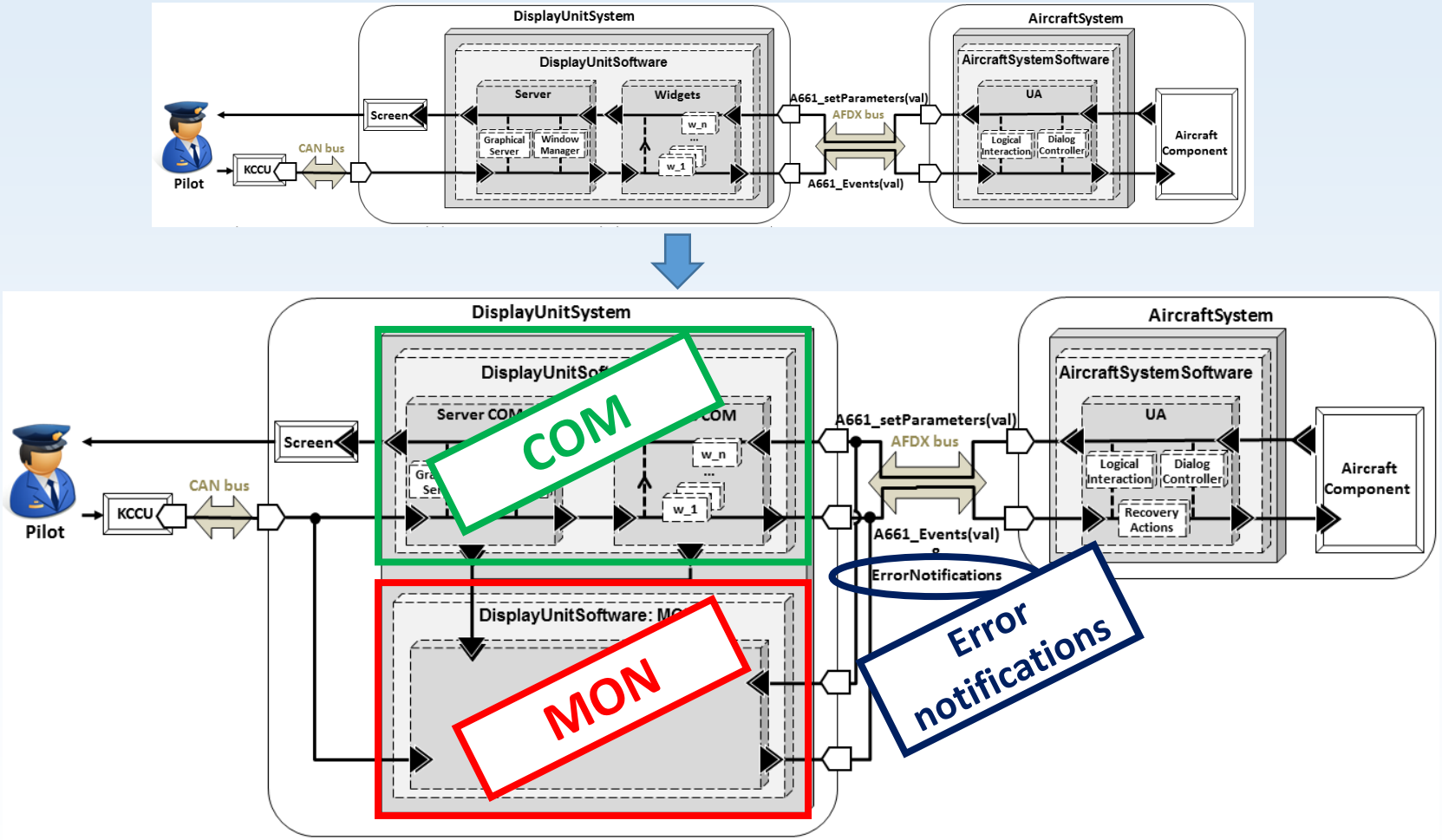
Applied to Electric Flight Control Units



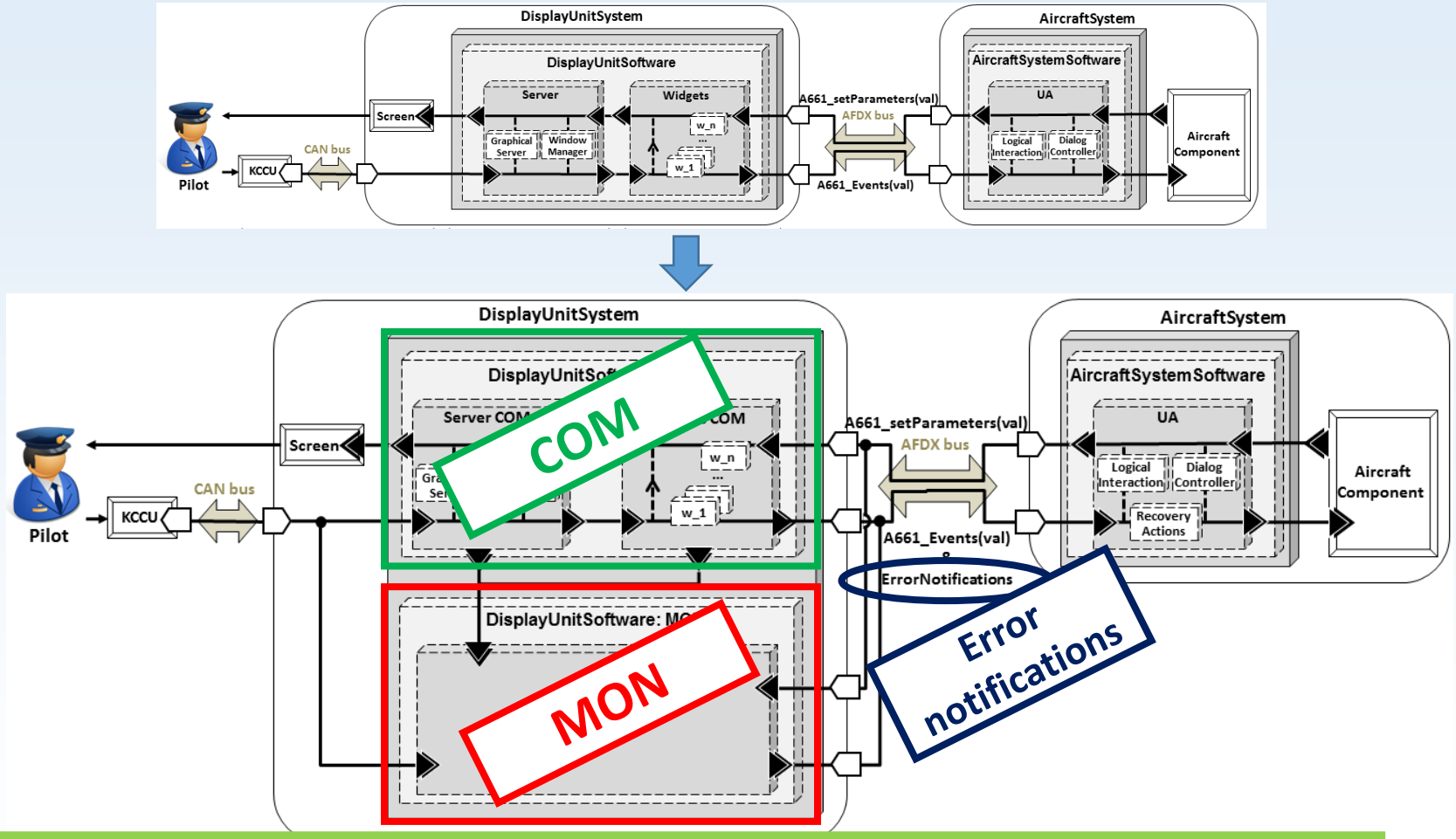
Laprie J.-C., Arlat J., Beounes C., Kanoun K. Definition and analysis of hardware- and software-fault-tolerant architectures. Computer 23, no. 7 (1990): 39-51

Traverse P., Lacaze I., Souyris J. Airbus fly-by-wire - A total approach to dependability. Building the Information Society, IFIP 18th World Computer Congress, Topical Sessions, 22-27 August 2004, Toulouse, France. 2004. 191-212.

System Software Architecture



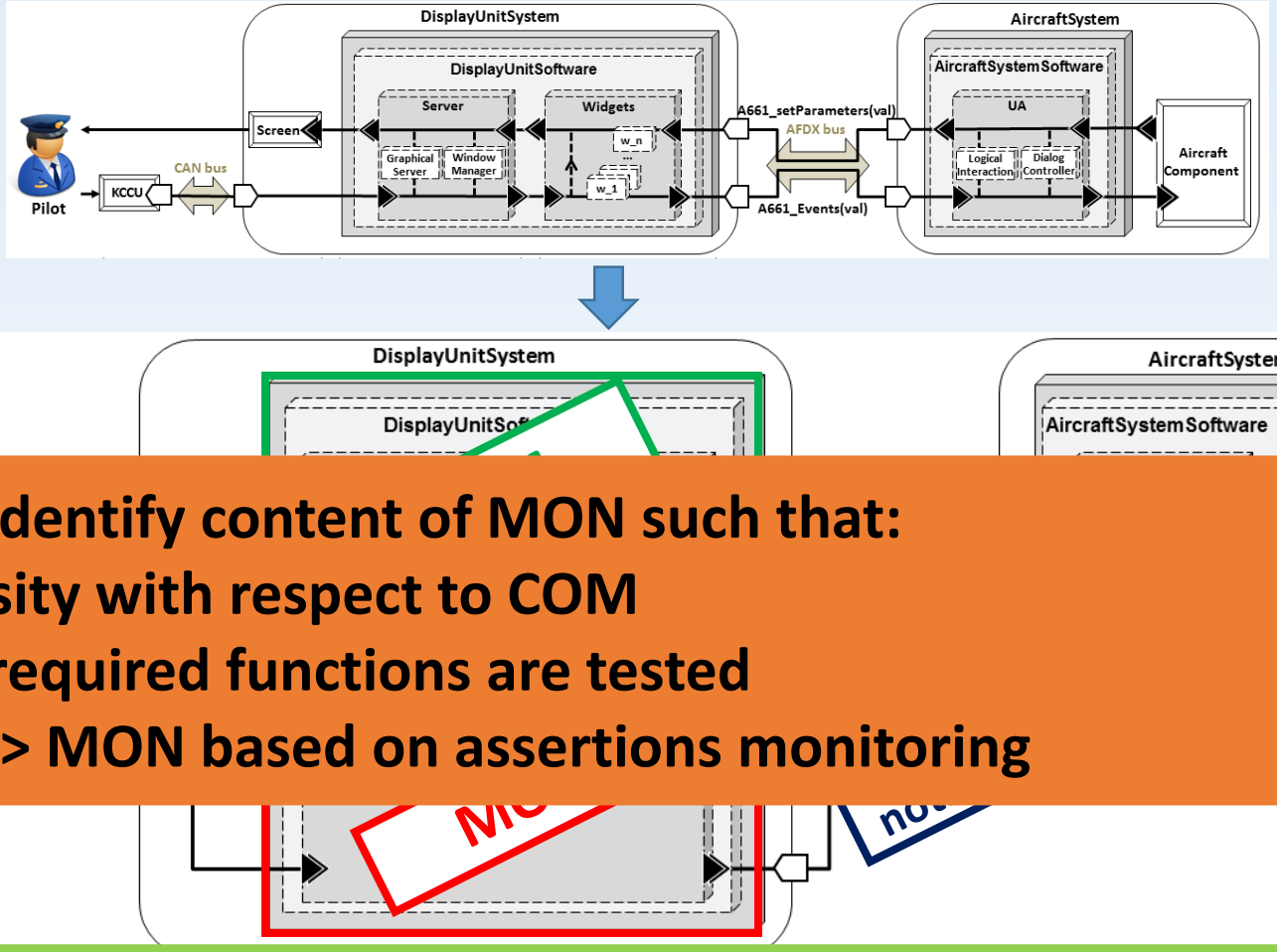
System Software Architecture



Definition of a global safety architecture

- *Taking into acount the server*
- *Enabling segregation*

System Software Architecture



How to identify content of MON such that:

- **Diversity with respect to COM**
- **Only required functions are tested**

=> MON based on assertions monitoring

Definition of a global safety architecture

- *Taking into acount the server*
- *Enabling segregation*

Assertions Definition Process

System definition and analysis

Architecture, ICO models and sequence diagrams

Failure modes identification

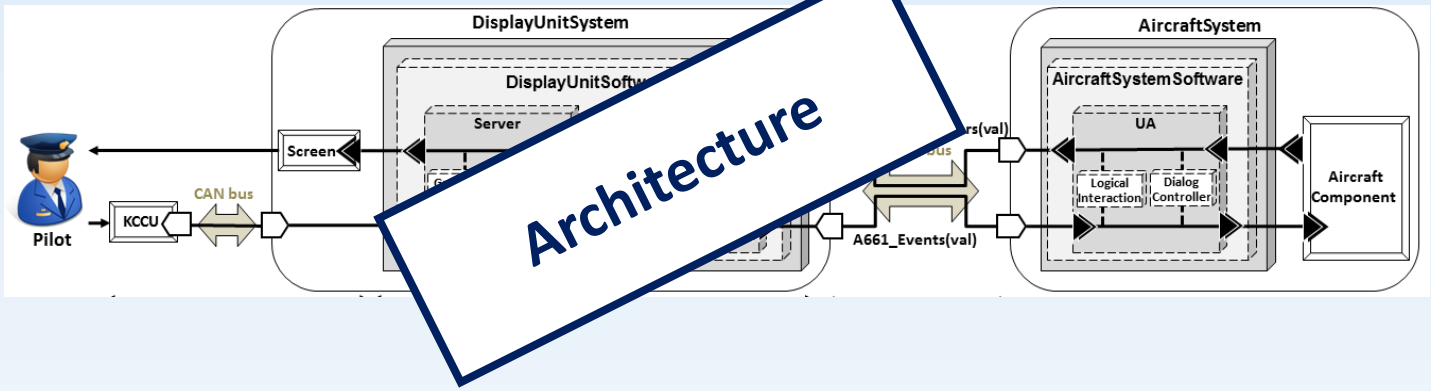
FMECA

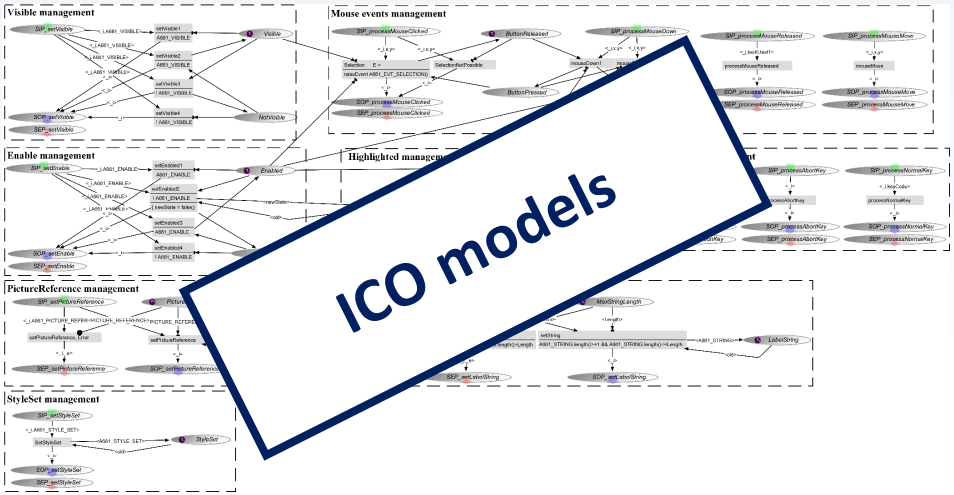
(Failure Mode Effects and Criticality Analysis)

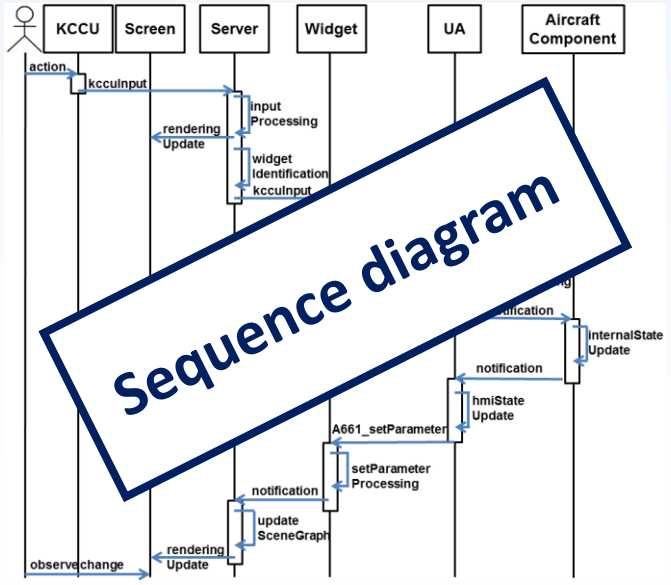
Assertion identification and assertion-based monitoring

Process for a systematic safety analysis

System Definition and Analysis







Failure Modes Identification

- FMECA template

1	2	3	4	5	6	7	8
Target/ item	Failure Modes	Potential Causes	Local effects	Upper-level effects	Risk level	Safety mechanisms (SM)	Upper-level effect with SM

HW and remaining SW faults

Control & data errors

Assertion-based monitoring & recovery actions

- Failure modes classification *(inspired by EASA)*
  - Loss of control
  - Erroneous control *(wrong control & spontaneous control)*
  - Loss of data display
  - Erroneous data display *(wrong display & spontaneous display)*

Failure Modes Identification

- Excerpt of server failure modes

1	2	3	4	5	6
Item	**Failures modes**		**Local effects**	**Upper-level effects**	**Consequence classification**
Server.inputProcessing Identify the kccu input and update cursor rendering	No execution Server.inputProcessing.FM1		Upon the receipt of a kccu input event, the server does not forward it	The pilot control is not sent to the aircraft system	Loss of control
	Erroneous execution Server.inputProcessing.FM2		Upon the receipt of a kccu input event, the server forwards the wrong kccu input event	A wrong control is sent to the aircraft system	Erroneous control
	Unexpected execution Server.inputProcessing.FM3		The server processes a kccu input event without receiving it	A control is sent to the aircraft system without any user action	Erroneous control
Server.widgetIdentification Identify the targeted widget, and forward of the inputEvent	No execution Server.widgetIdentification.FM1		Upon the receipt of a kccu input event, the server does not forward it	The pilot control is not sent to the aircraft system	Loss of control
	Erroneous execution Server.widgetIdentification.FM2		Upon the receipt of a kccu input event, the server forwards it to a wrong widget	A wrong control is sent to the aircraft system	Erroneous control
	Unexpected execution Server.widgetIdentification.FM3		The server sends a kccu input event to a widget without receiving it	A control is sent to the aircraft system without any user action	Erroneous control
Server.updateSceneGraph Update the scene graph and the graphical rendering of the application	No execution Server.updateSceneGraph.FM1		Upon the receipt of an update notification, the server does not update the scene graph or the graphical rendering	The pilot is not notified of the aicraft system state change	Loss of data display
	Erroneous execution Server.updateSceneGraph.FM2		Upon the receipt of an update notification, the server updates the scene graph and/or the graphical rendering in a wrong way	The pilot is notified of a wrong aicraft system state change	Erroneous data display
	Unexpected execution Server.updateSceneGraph.FM3		The server updates the scene graph or the graphical rendering without receiving any update notification	The pilot is notified of an aircraft system state change that did not occur	Erroneous data display

Assertion Identification & Formalization

- **Server.widgetIdentification**

  Identification of the target widget, and forwarding of the inputEvent to the target widget

A1: Correct widget identification assertion

Let s be a Server, let u be a User, let W be the set of widgets of the application and let E be the set of KCCU events

$$\forall\, w \in W\,, \forall\, e \in E,\ w = s.\text{widgetIdentification}(e)$$

$$\Leftrightarrow$$

$$w.\text{visibility} = \text{true} \wedge w.\text{enabling} = \text{true} \wedge e.(x,y) \subset w.\text{zone} \wedge e = u.\text{action}(\,)$$

Assertion Identification & Formalization

- **Server.widgetIdentification**

  Identification of the target widget, and forwarding of the inputEvent to the target widget

A1: Correct widget identification assertion

Let s be a Server, let u be a User, let W be the set of widgets of the application and let E be the set of KCCU events

$\forall$ w $\in$ W ... ation(e)

$\Leftrightarrow$

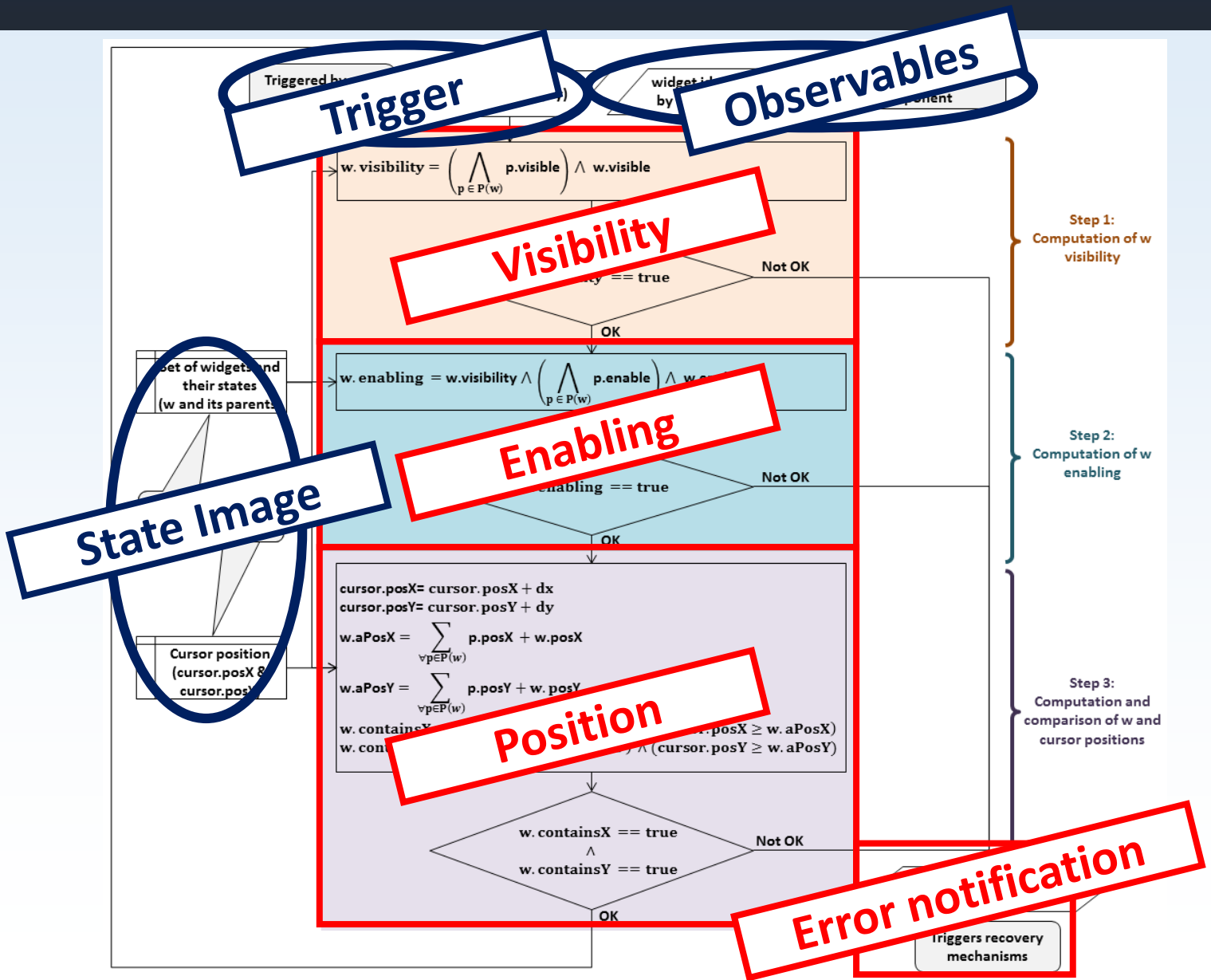
w.visibility = true $\wedge$ w.enabling ... $\subset$ w.zone $\wedge$ e = u.action( )

Trigger

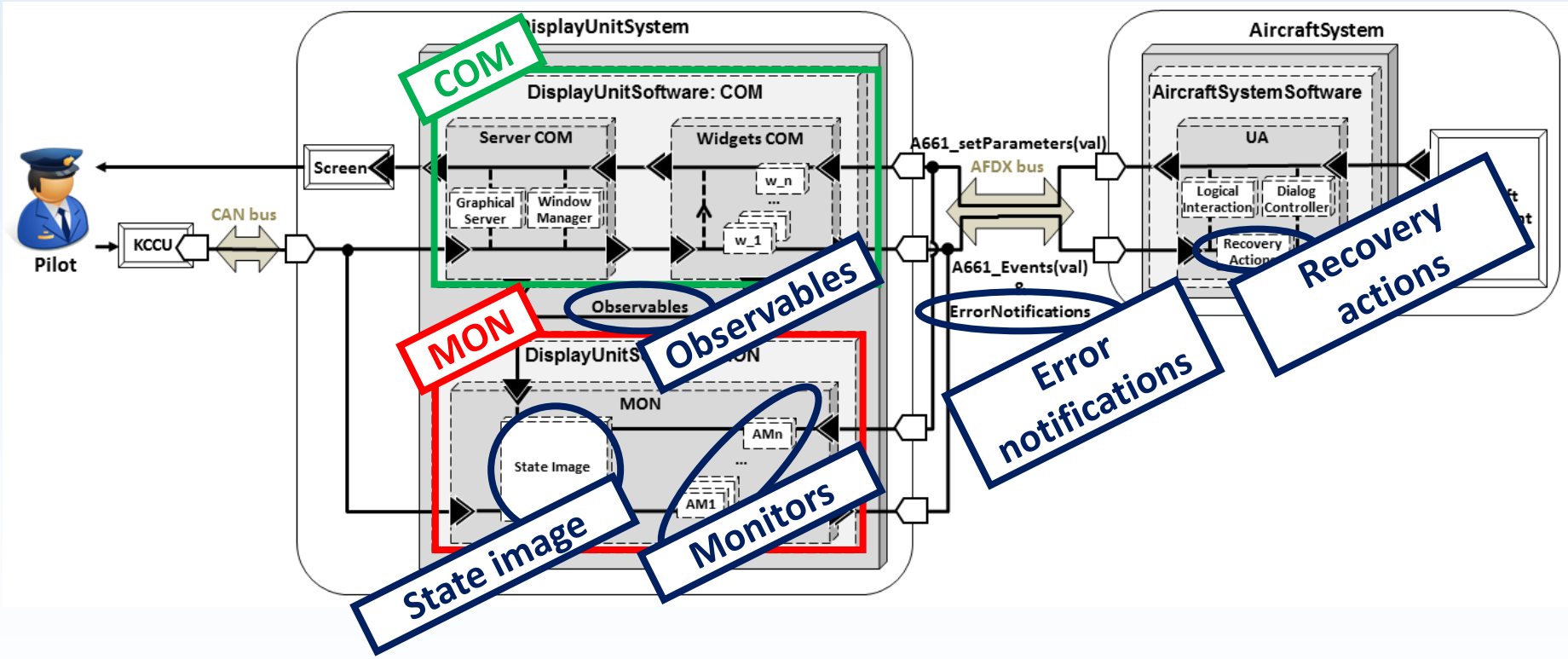
Observables

State image

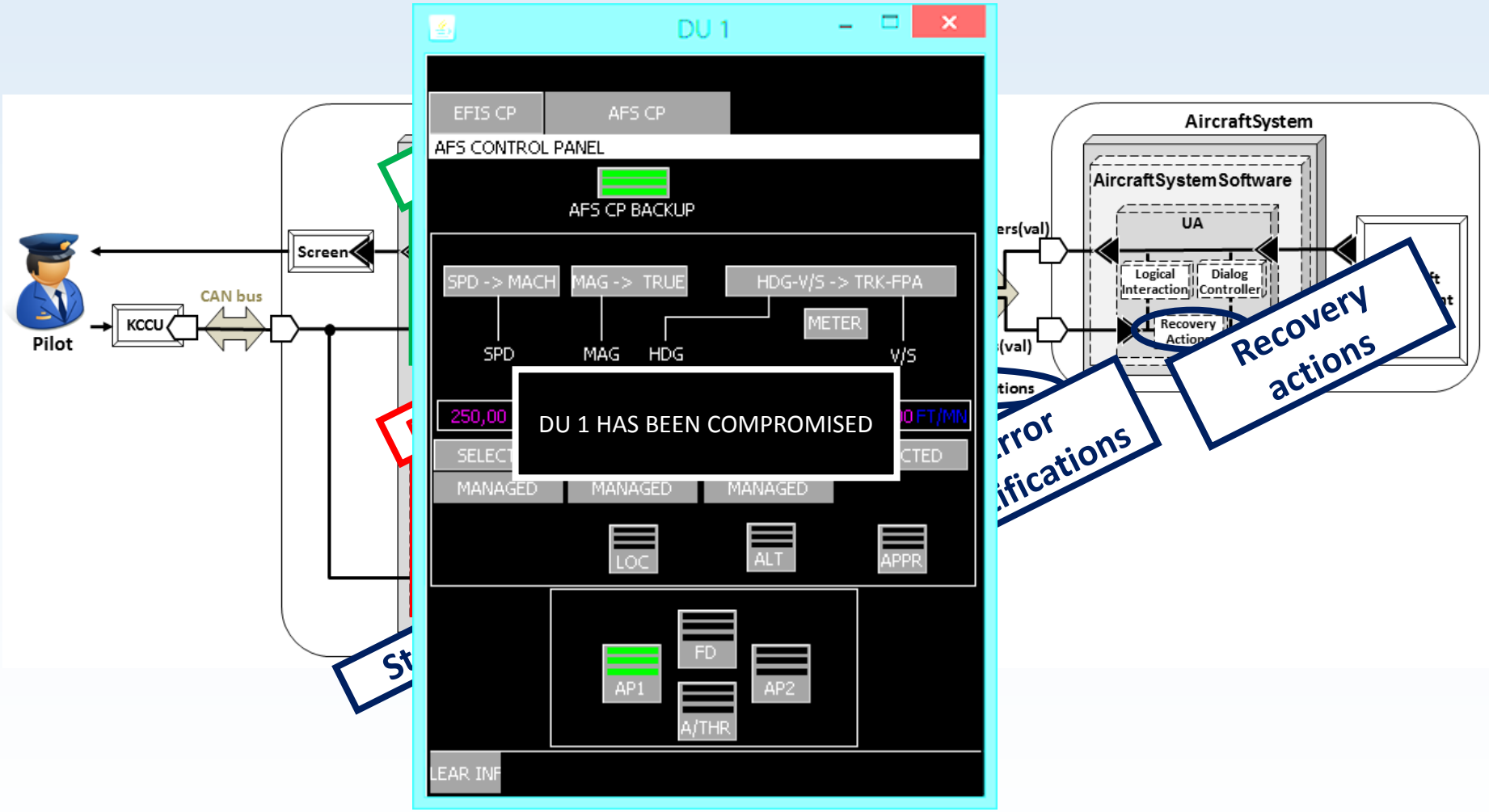
Assertion-Based Monitoring



Software Architecture

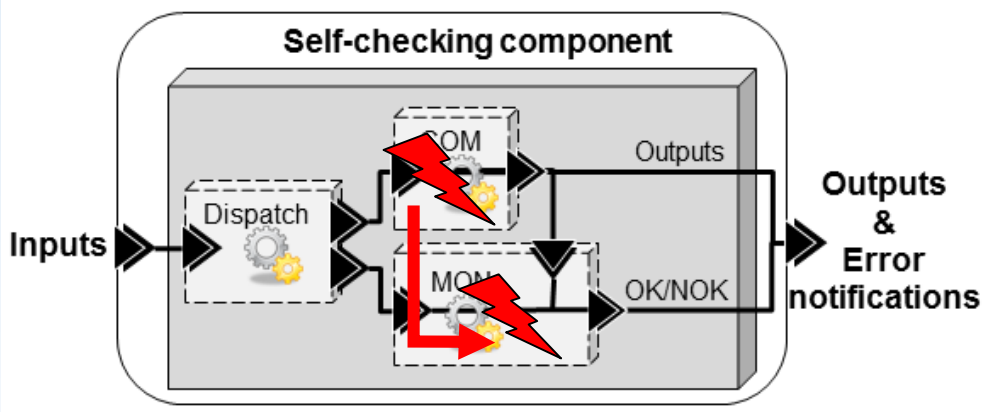


Software Architecture



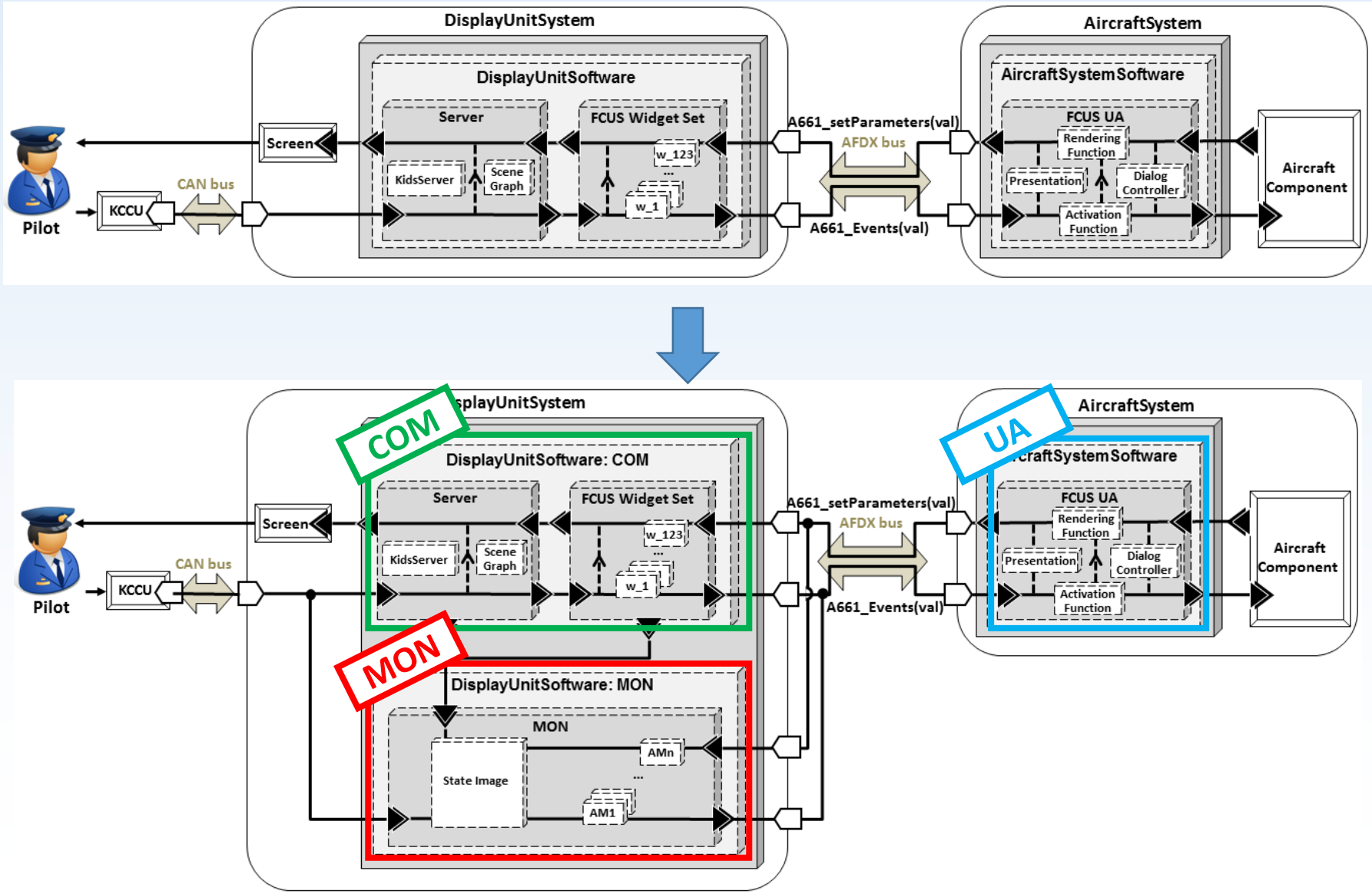
Implementation

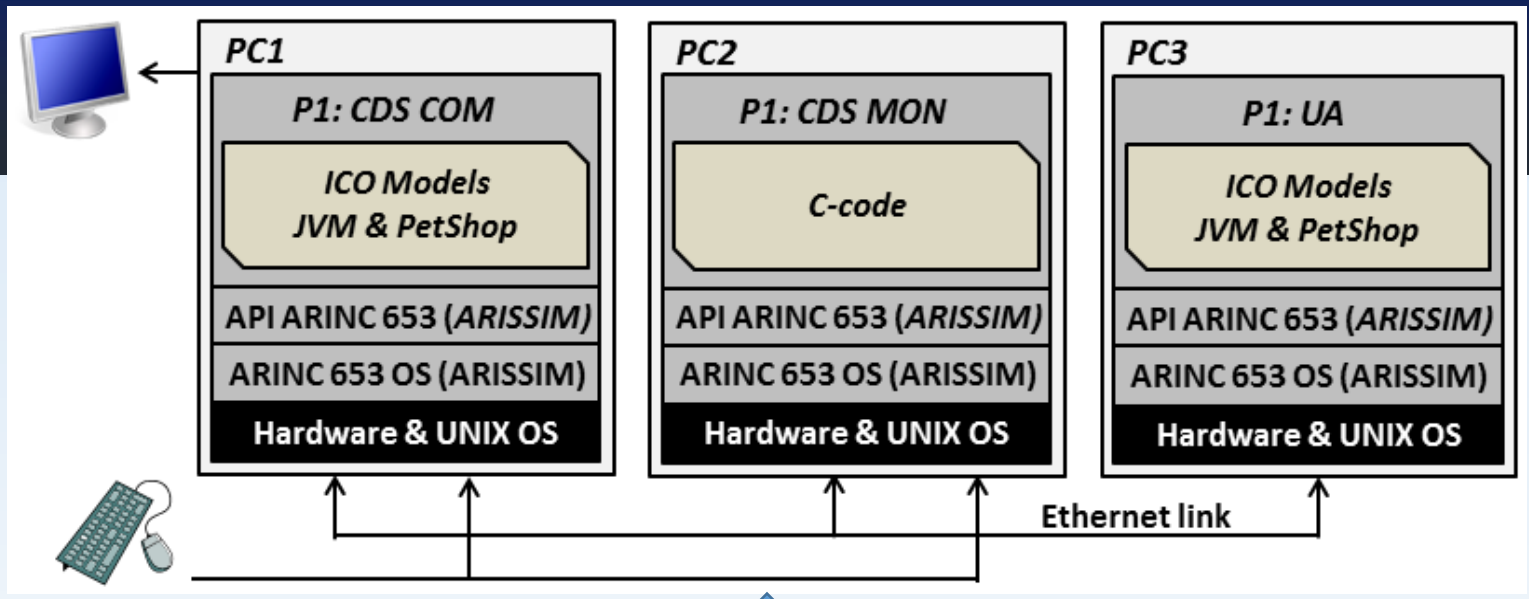
- Segregation through software partitioning

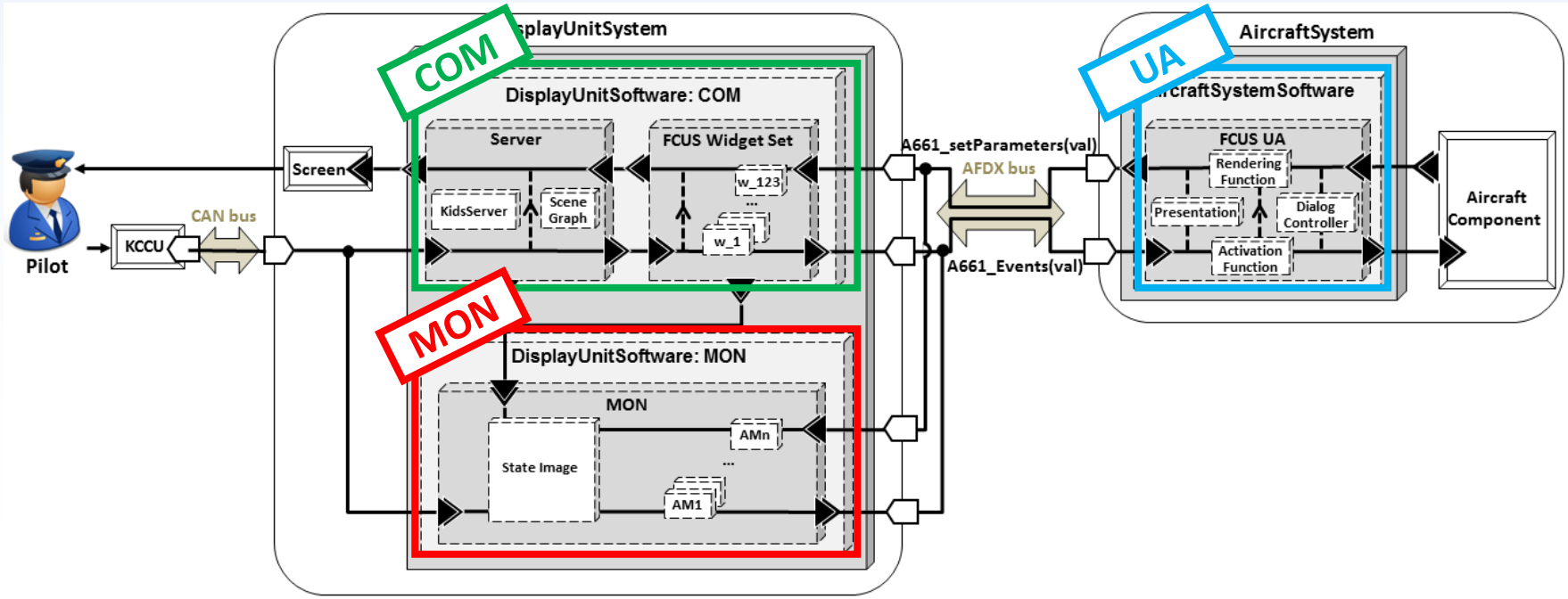


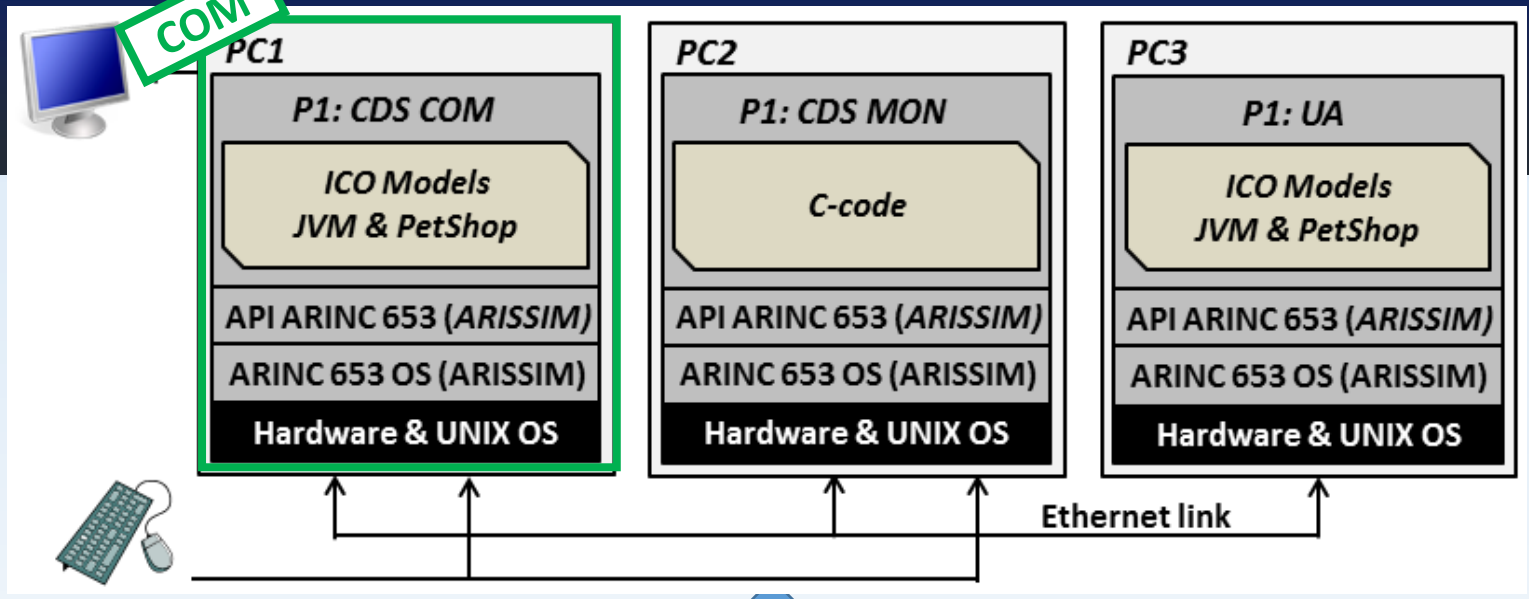
- The ARINC 653 runtime support
  - Standard architecture for avionic run-time support
  - Time & Space Partitioning

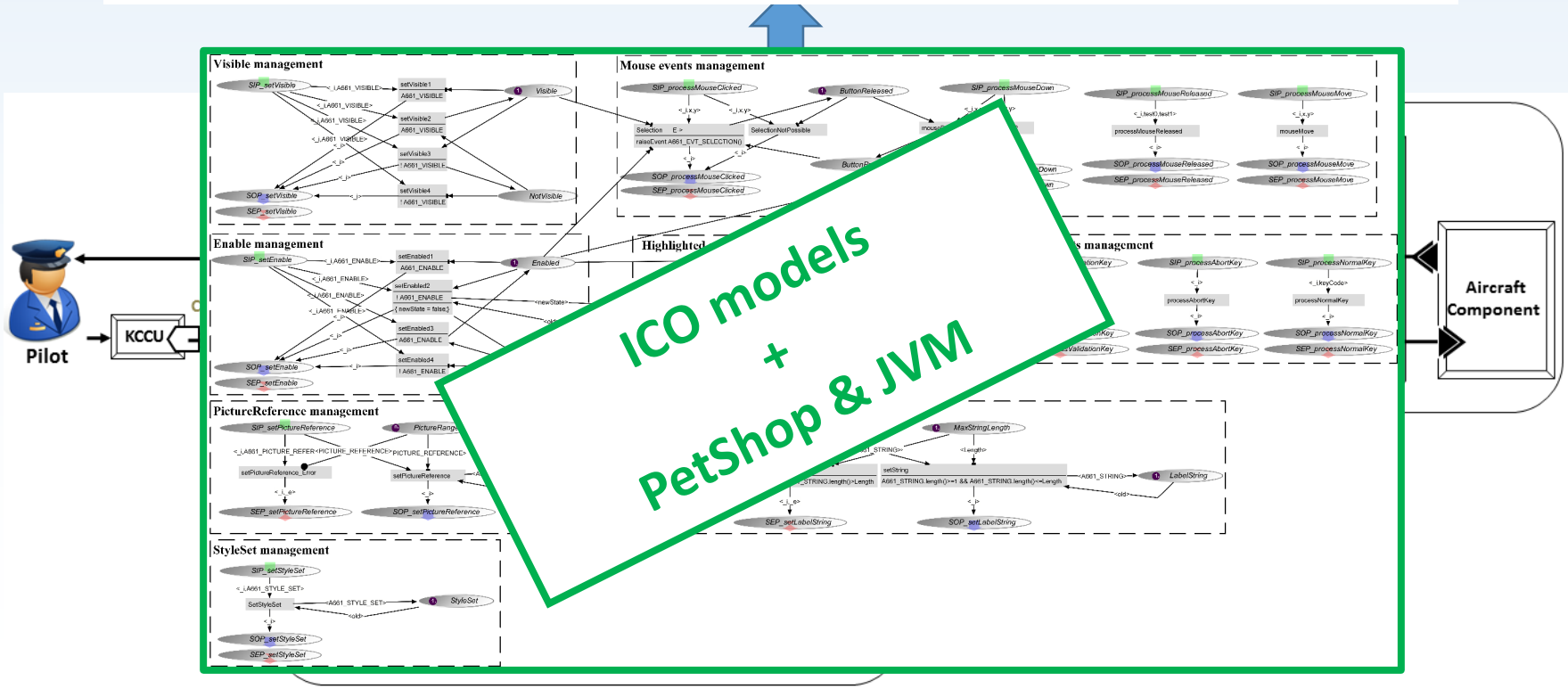
Implementation

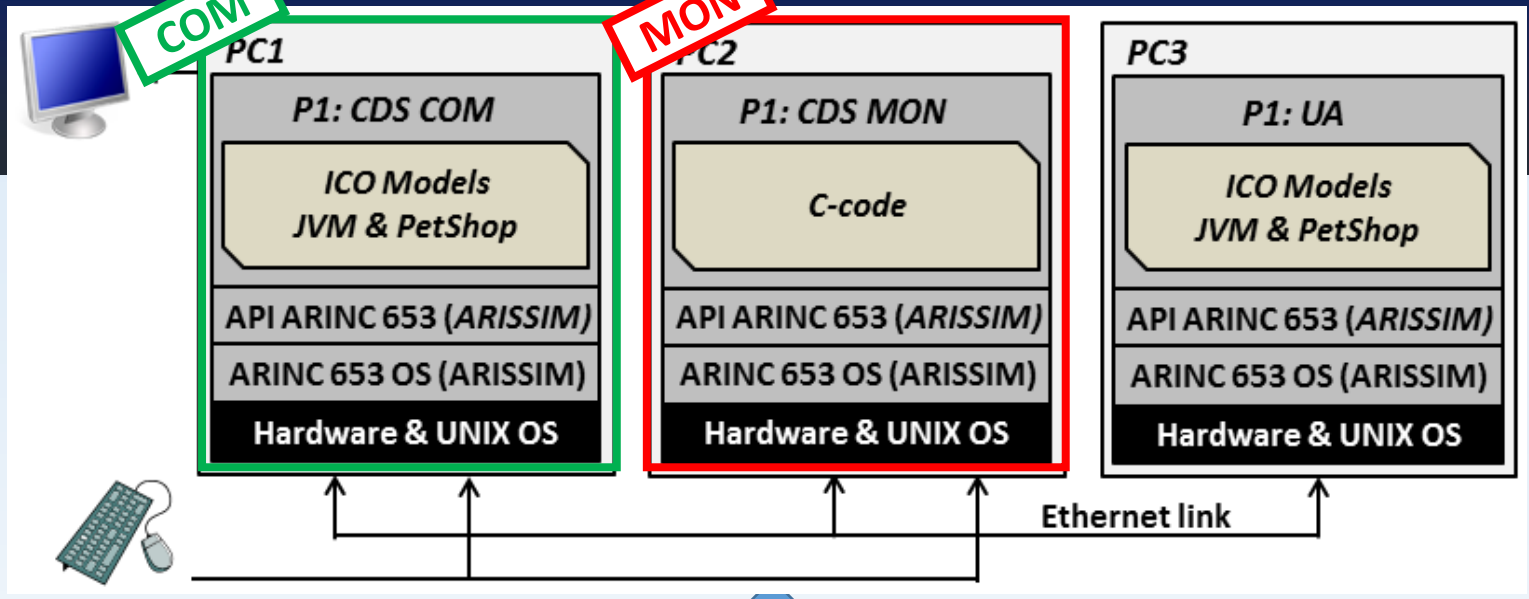


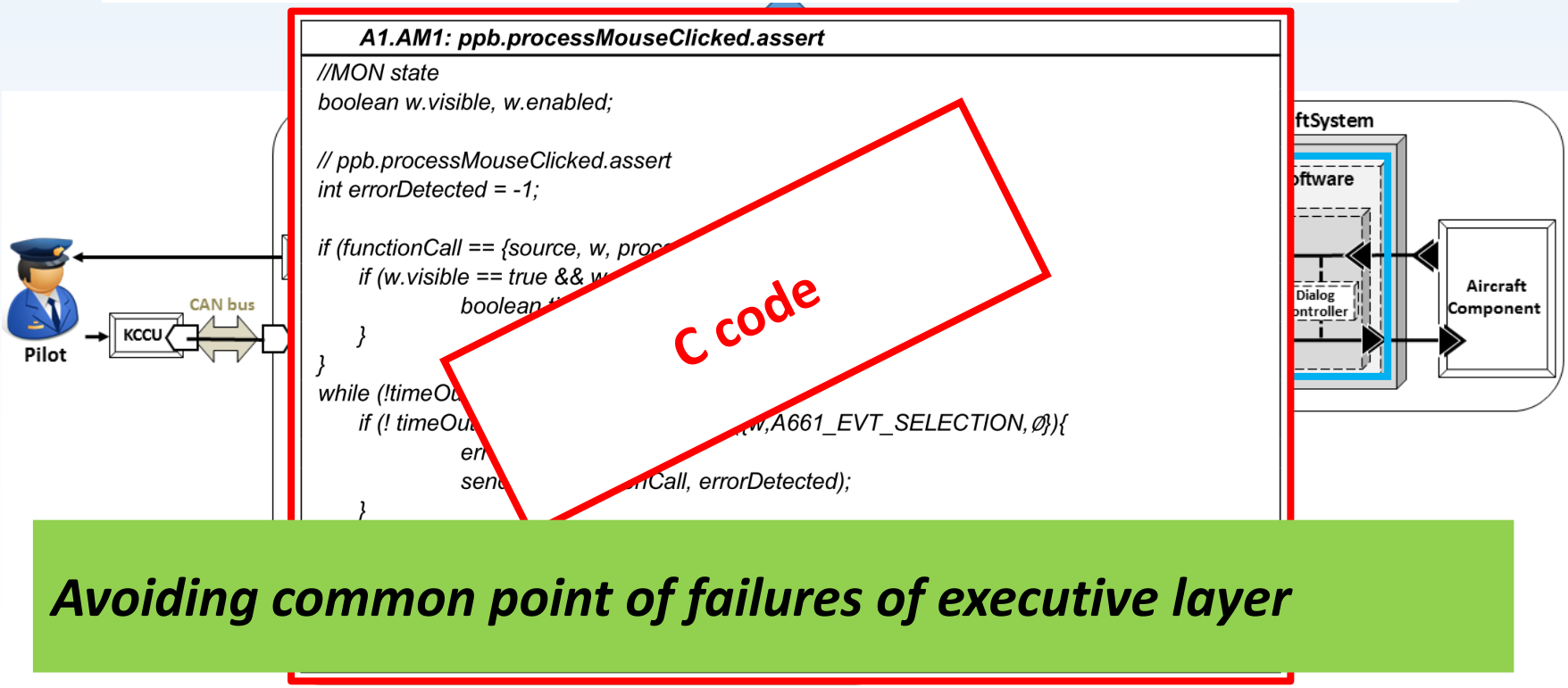

PC1
P1: CDS COM
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC2
P1: CDS MON
C-code
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC3
P1: UA
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
Ethernet link


COM
DisplayUnitSystem
DisplayUnitSoftware: COM
Server
FCUS Widget Set
w_123
KidsServer
Scene Graph
w_1
Screen
CAN bus
KCCU
Pilot
MON
DisplayUnitSoftware: MON
MON
AMn
State Image
AM1
A661_setParameters(val)
AFDX bus
A661_Events(val)
UA
AircraftSystem
AircraftSystemSoftware
FCUS UA
Rendering Function
Presentation
Dialog Controller
Activation Function
Aircraft Component

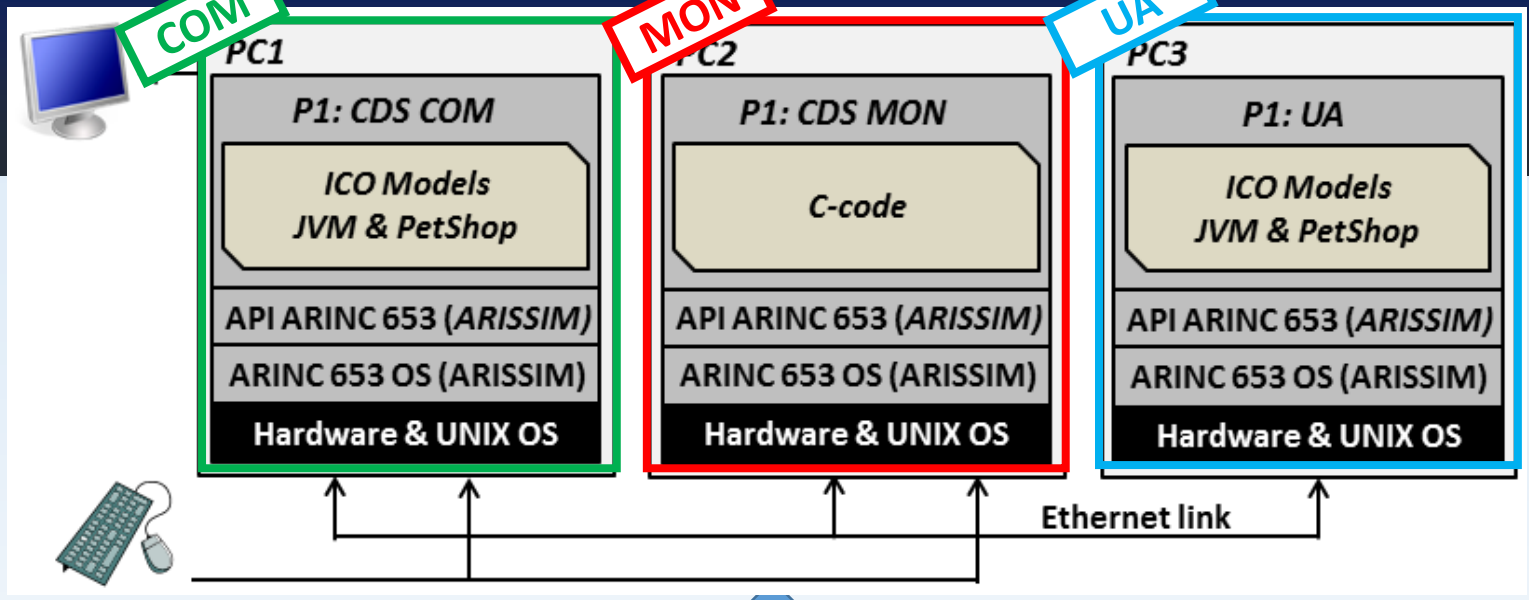

COM
PC1
P1: CDS COM
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC2
P1: CDS MON
C-code
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC3
P1: UA
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
Ethernet link

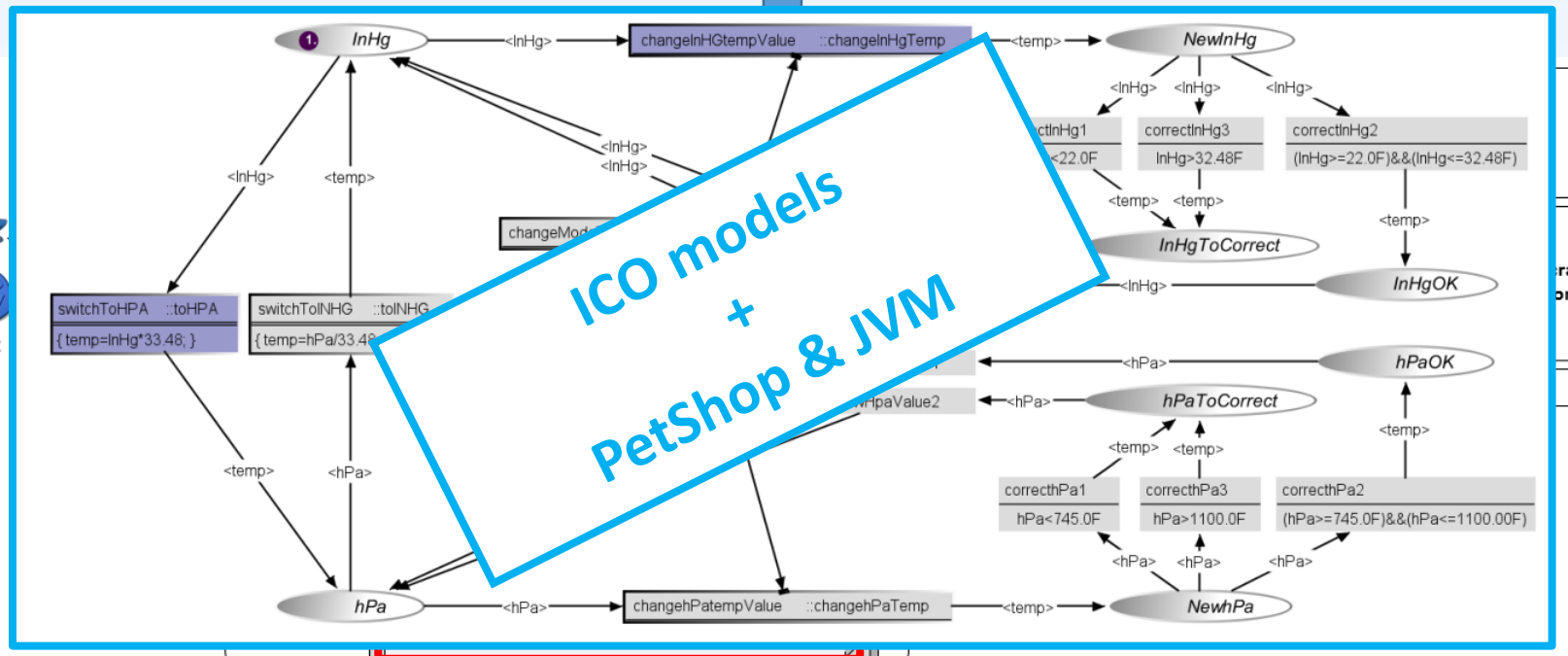

Visible management
Mouse events management
Enable management
Highlighted
PictureReference management
StyleSet management
Pilot
KCCU
Aircraft Component
ICO models
+
PetShop & JVM





Avoiding common point of failures of executive layer


COM
MON
UA
PC1
P1: CDS COM
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC2
P1: CDS MON
C-code
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
PC3
P1: UA
ICO Models
JVM & PetShop
API ARINC 653 (ARISSIM)
ARINC 653 OS (ARISSIM)
Hardware & UNIX OS
Ethernet link


InHg
changeInHGtempValue
changeInHgTemp
NewInHg
correctInHg3
correctInHg2
(InHg>=22.0F)&&(InHg<=32.48F)
InHgToCorrect
InHgOK
switchToHPA
switchToINHG
hPaOK
hPaToCorrect
correcthPa1
correcthPa3
correcthPa2
(hPa>=745.0F)&&(hPa<=1100.00F)
hPa
changehPatempValue
changehPaTemp
NewhPa
ICO models
+
PetShop & JVM
Pilot

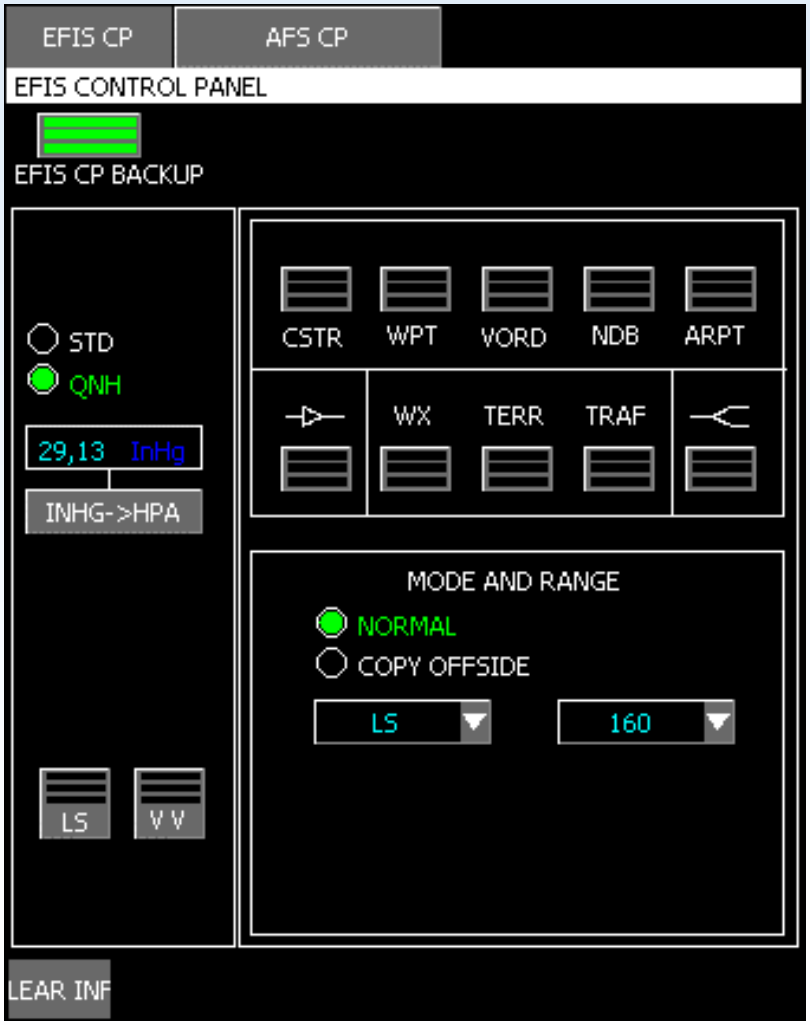
Process and Architecture - Summary

- Global fault tolerant system architecture
  - Fault detection using COM-MON principles
  - Applied to the generic part (the CDS)
- Assertion definition process
  - Safety analysis
  - Assertion formalization
  - Assertion-based monitors
- Implementation principles
  - Based on ARINC 653 principles
  - Development of an ARINC 653 simulator
  - Partitioning of COM and MON components

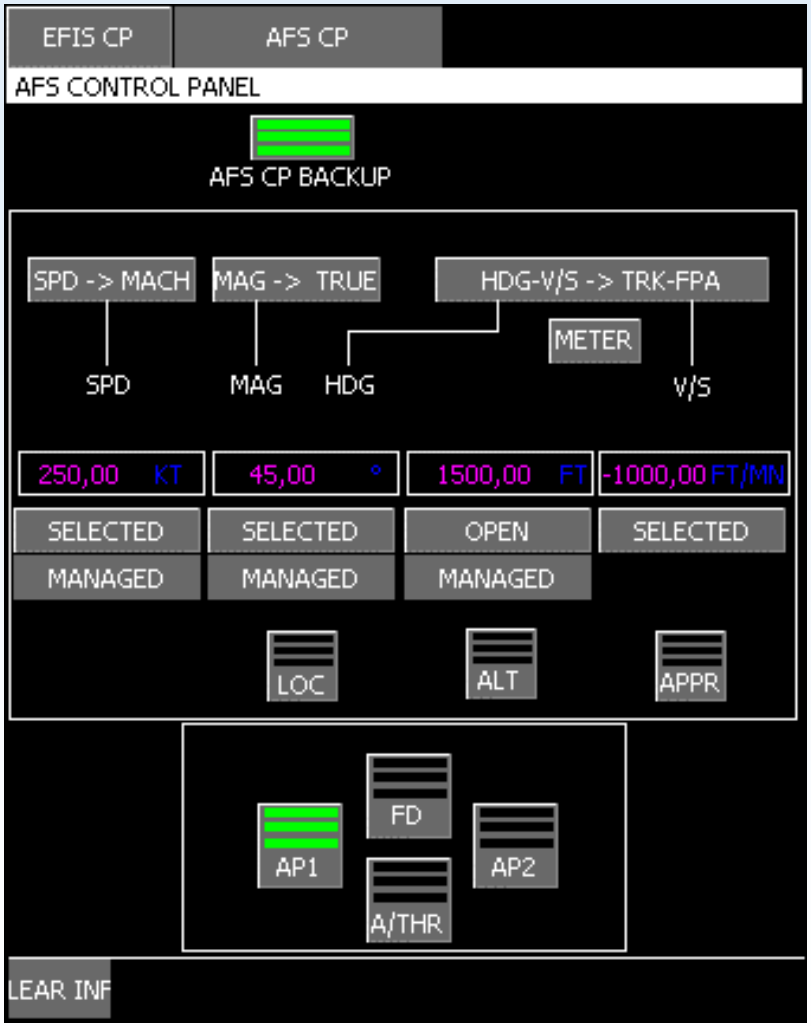
Outline of the talk

- Introduction and Problem Statement
- Context (Interactive Cockpits)
- Proposed Approach for Dependable Interactive Systems/Cockpits
- **Case Study**
- Conclusions and Perspectives

FCUS Application

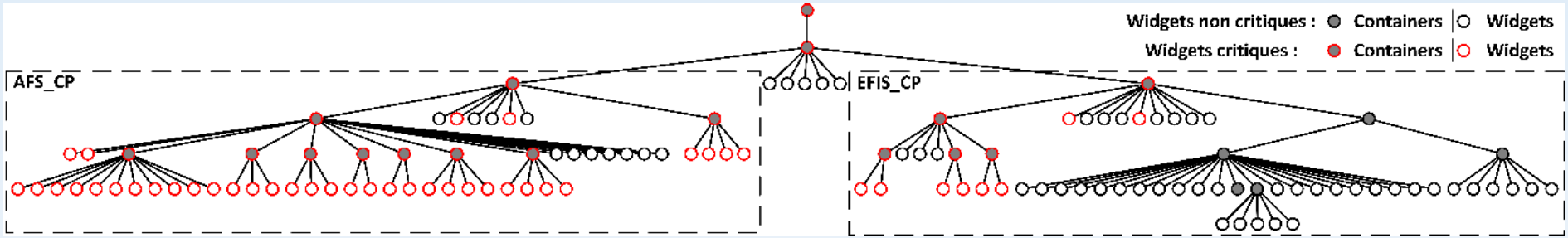


Panneau EFIS CP

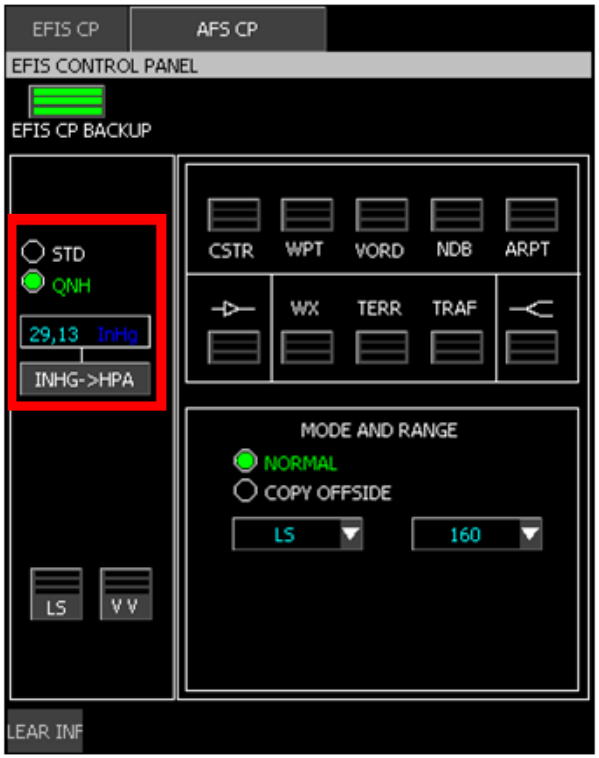


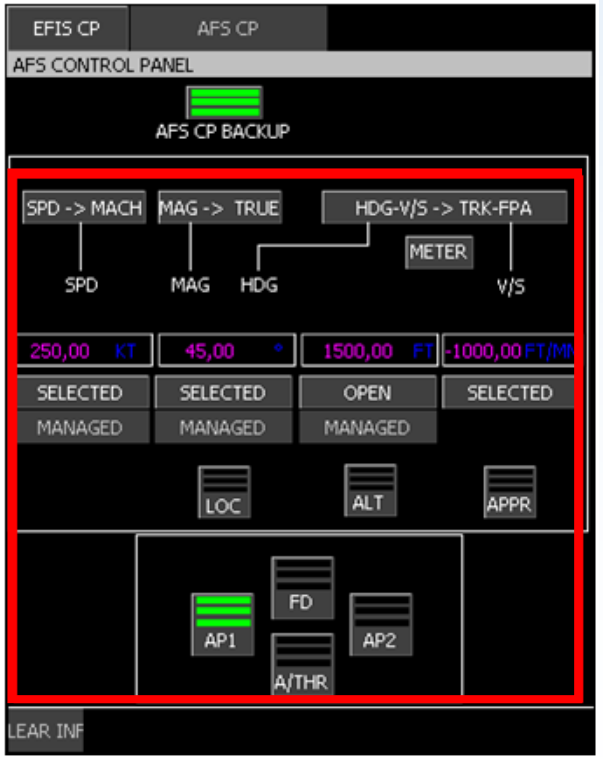
Panneau AFS CP

FCUS Application



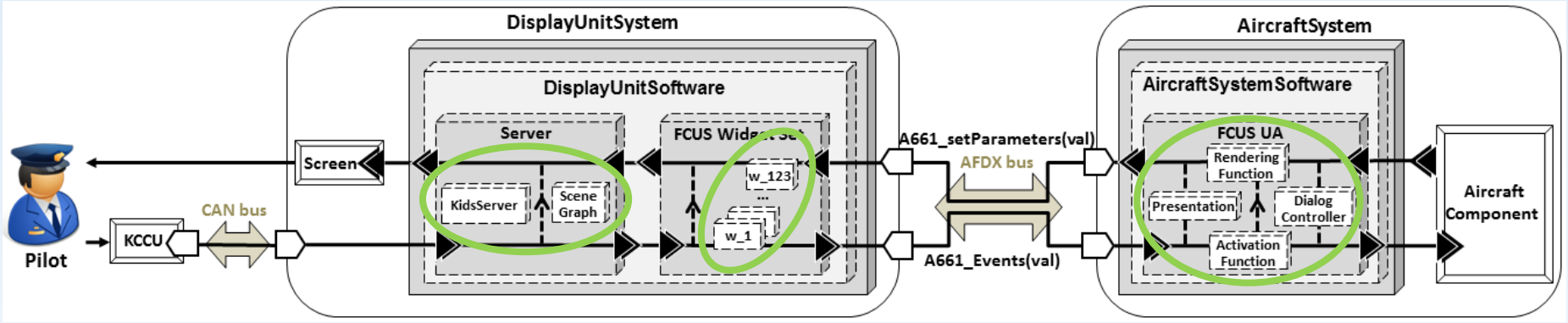
Widget types	Widgets number	Critical widgets number
Layer	1	1
BasicContainer	12	10
Panel	7	4
RadioBox	1	1
CheckButton	4	2
ComboBox	3	0
EditBoxNumeric	8	8
PicturePushButton	40	25
PictureToggleButton	2	2
Label	32	4
GP_Line	11	7
Picture	2	0





Critical widgets

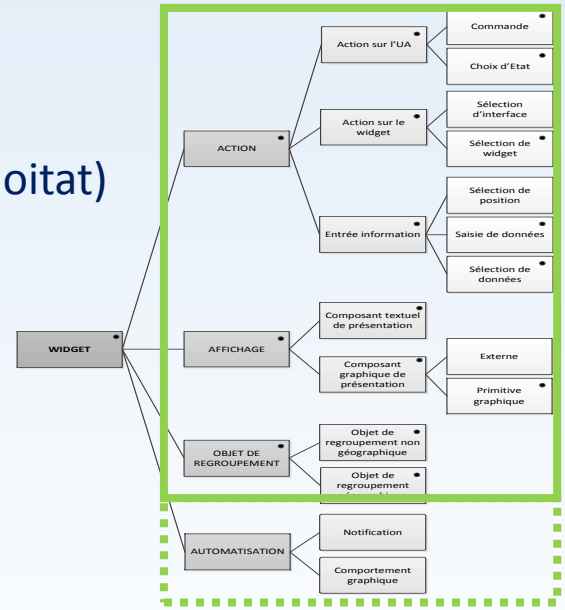
Model-Based Approach Implementation



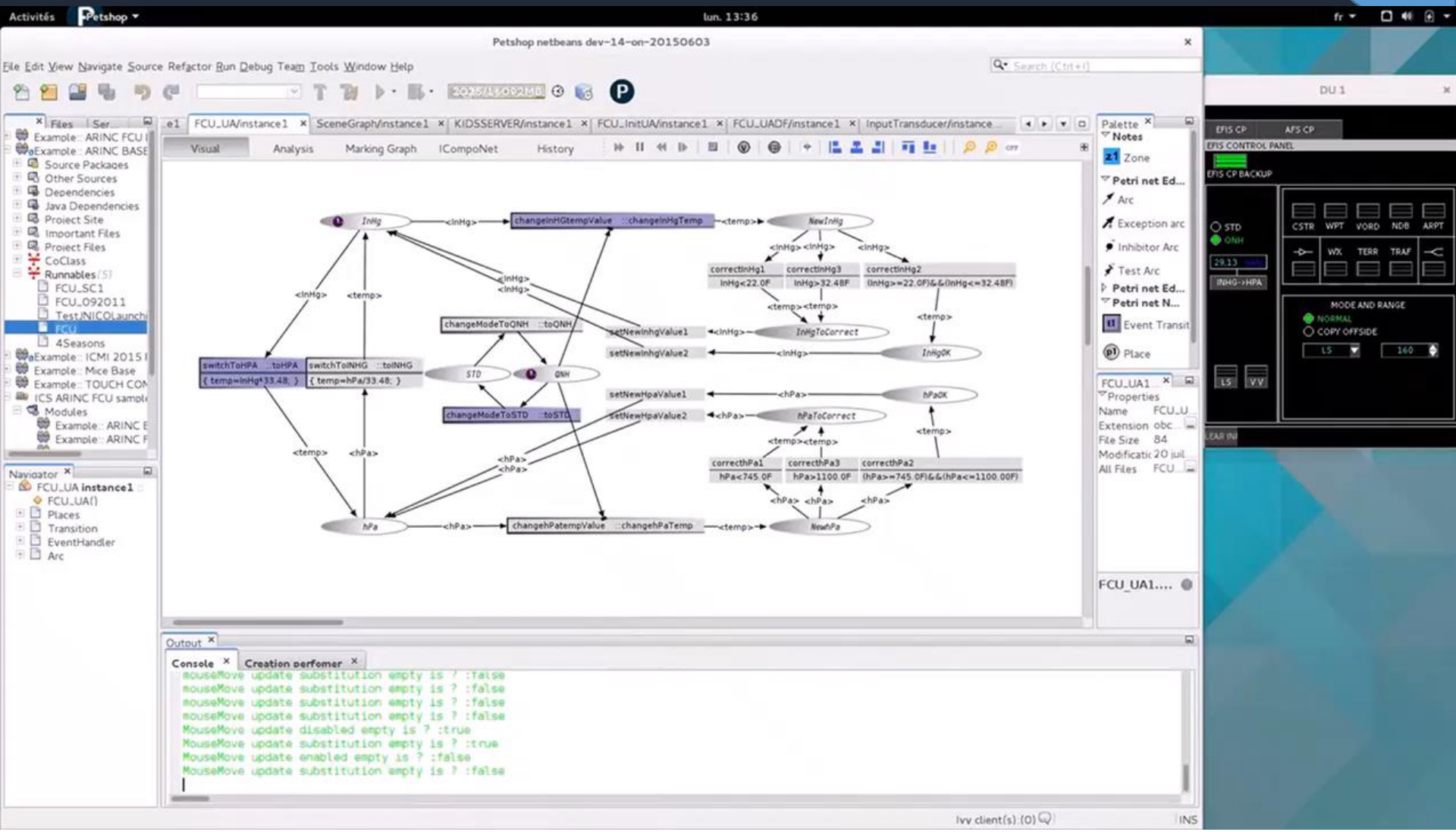
- 2 models for the server
- 12 widget types modelled
  - 123 widget instances
- 4 models for the UA
- 8 models for initialization

Model-Based Approach Implementation

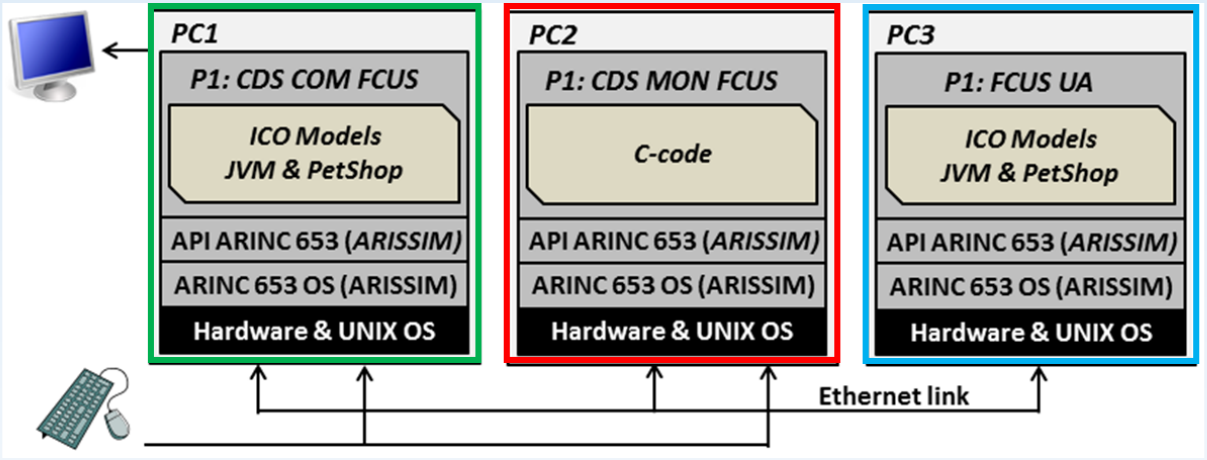
- ARINC 661 and WIMP interaction coverage
  - Widget classification coverage
    - 3 classes used in FCUS
    - 1 classe used in previous work (Adrienne Tankeu-Choitat)
  - ARINC 661 server description
    - SceneGraph
    - Picking
- Scalability

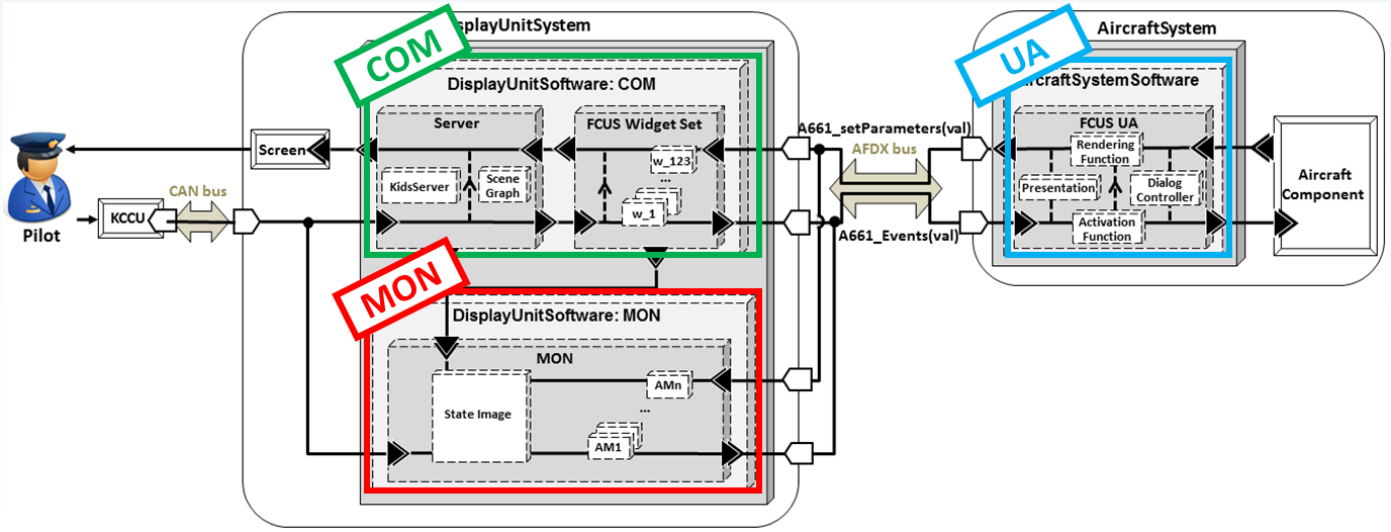


Short demo:
The FCUS running with ICO and PetShop



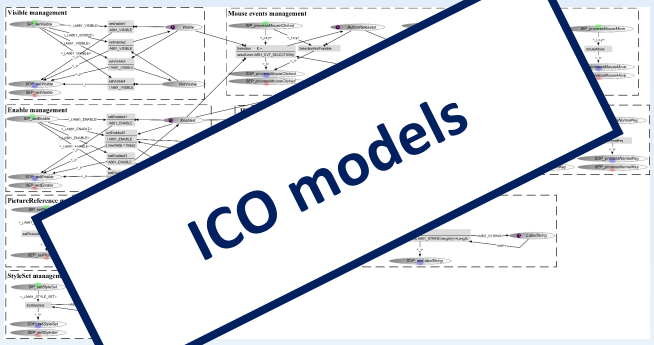
Software Architecture Implementation





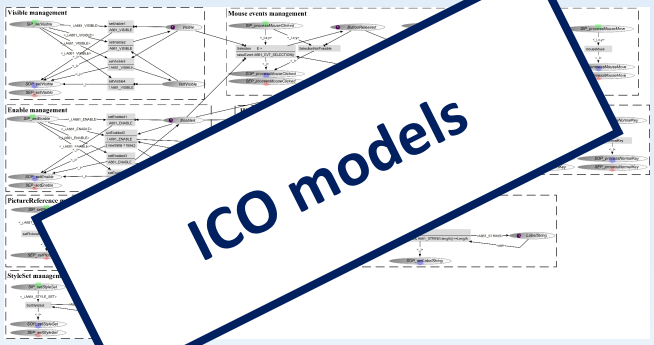
Assertion & Monitor Definition Process Application

Architecture

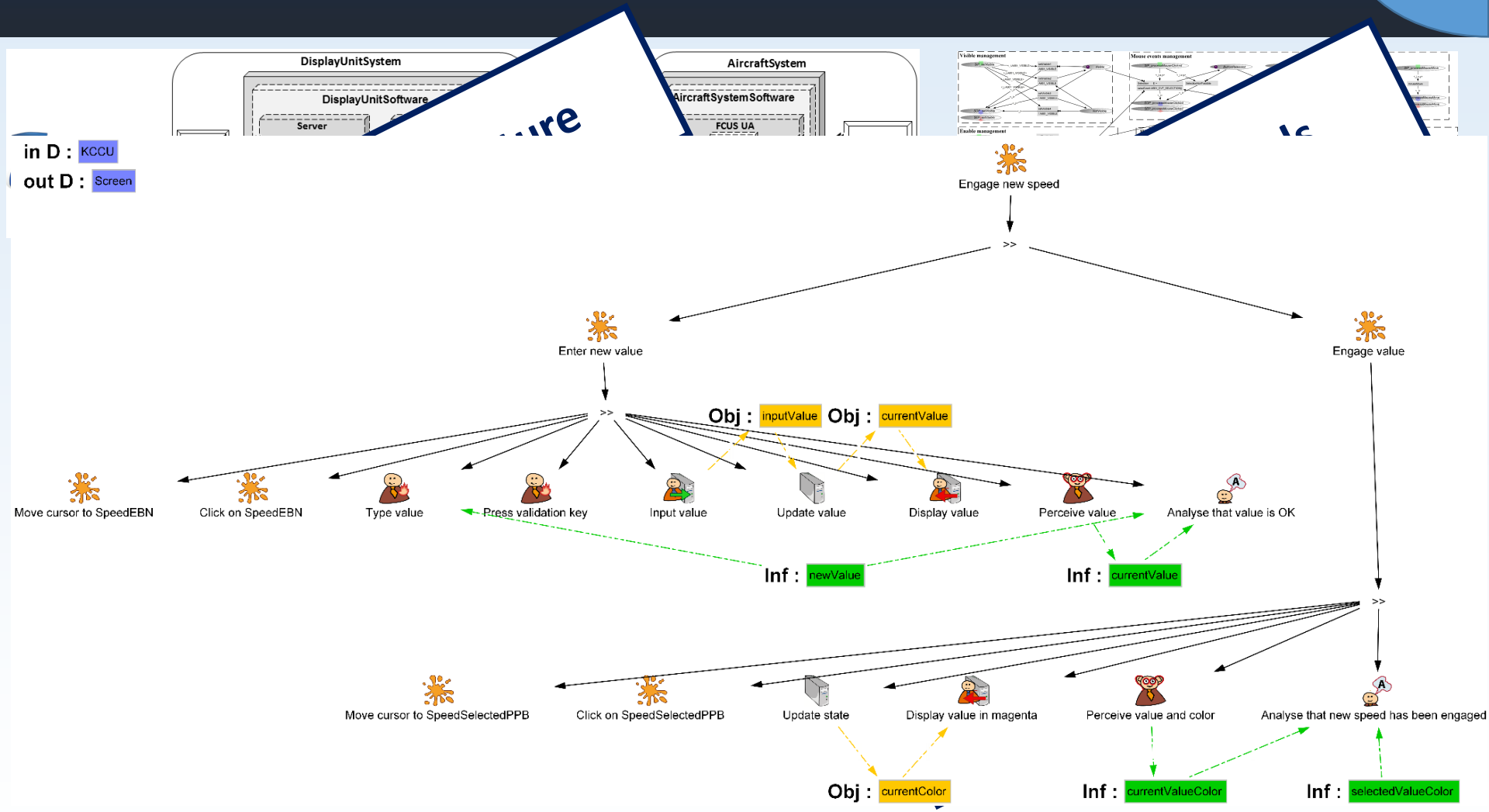


ICO models

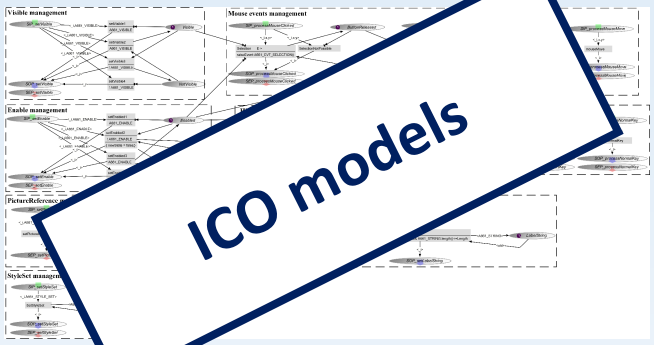
Assertion & Monitor Definition Process Application

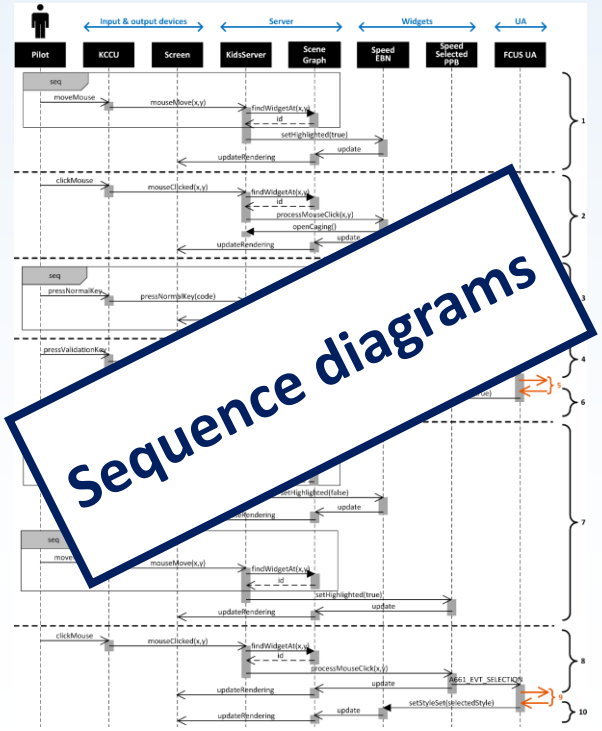


Assertion & Monitor Definition Process Application

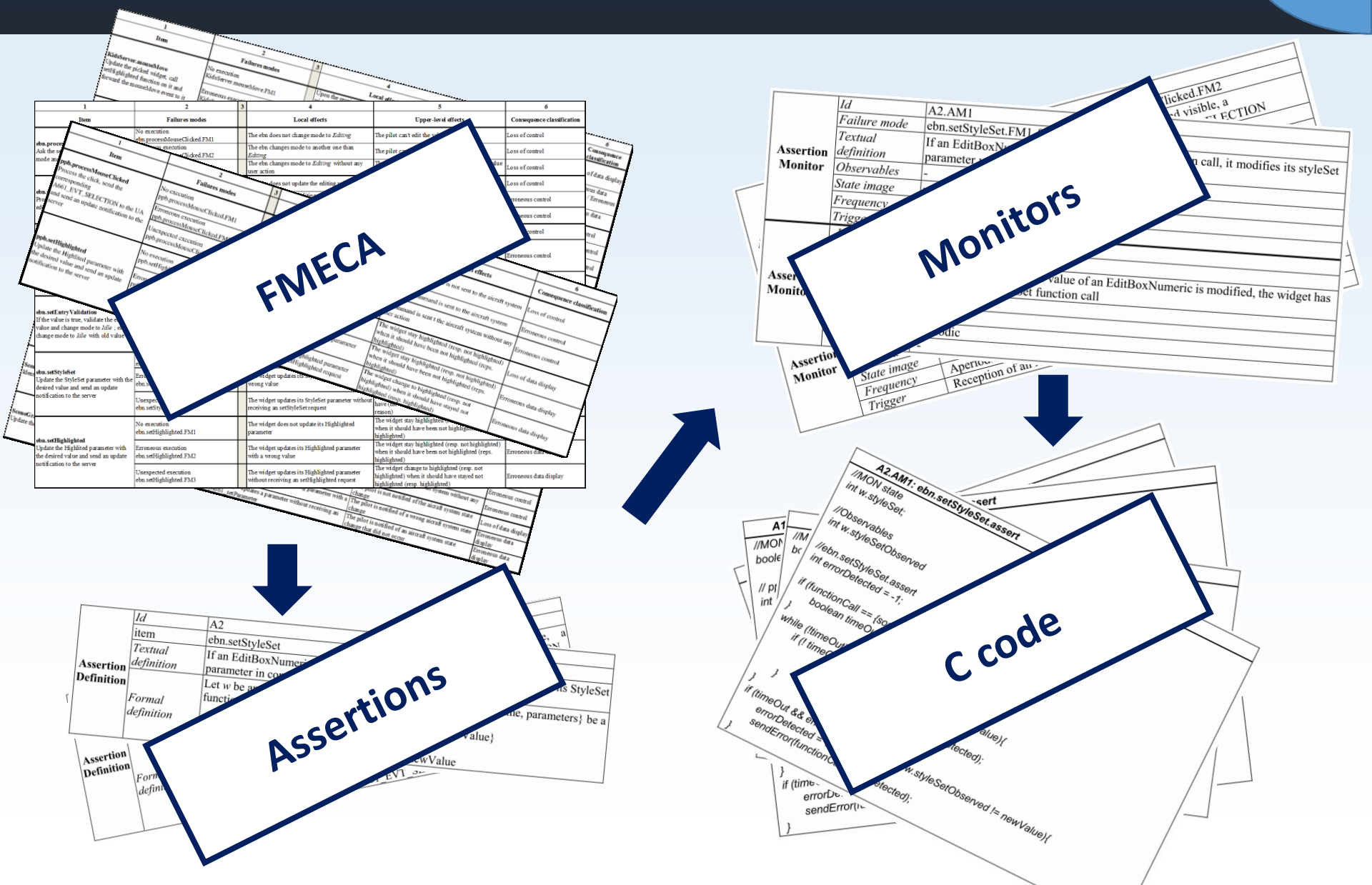


Assertion & Monitor Definition Process Application





Assertion & Monitor Definition Process Application



Assertion & Monitor Definition Process Application Software Architecture Implementation

- ARISSIM simulator development
  - ~ 10 000 lines of code
- Assertion definition process
  - ~ 30 assertion monitors
- PetShop implementation on ARISSIM
  - ~ 1000 lines of code for the connection
  - ~ 140 ICO models

Outline of the talk

- Introduction and Problem Statement
- Context (Interactive Cockpits)
- Proposed Approach for Dependable Interactive Systems/Cockpits
- Case Study
- **Conclusions and Perspectives**

Conclusion – A Two-Fold Approach

- Model-Based Approach
  - ICO formal notation
  - Software faults prevention

SceneGraph and Picking models

- Software Fault Tolerant Architecture
  - Based on COM-MON principles
  - Safety analysis process for MON definition

System architecture
Systematic safety process

- Means for implementation
  - ARISSIM simulator
  - PetShop and ICO models for COM
  - C code for MON

Partitioning
Avoidance of common point of failure (execution)

- Refinement of usability assesment approach

Perspectives

- Integration in a complete development process
  - Improved verification & validation of ICO models
- Fault tolerance mechanisms validation
  - Detection coverage (e.g. using fault injection)
  - Recovery mechanisms (e.g. involving crew operations, self-healing systems)
- Model-Based approach extension
  - Interpretation of models at runtime in an operational context
  - Code generation towards a certified environment
- Fault tolerance approach extension
  - Application to UA
  - Input/Output devices
  - Human error
- More sophisticated interaction techniques
  - Multi-touch // ARINC 661 extensions
  - Multimodality

Thank you for your attention!

Questions ?