Improved Model Based approach to address the Architecture, Design and Specification of complex systems.

Matlab Expo 2018

19 June | Paris

Jean Duprez, Airbus Operations SAS

19 June, 2018

AIRBUS

How ? ➔ Easy & efficient use.

“ Make things simple & easy !”

"**True simplicity** is derived from so much more than just the absence of clutter and ornamentation. It's about

bringing order to complexity…

You have to **deeply understand** the essence of a product in order to be able to get rid of the parts that are not essential.”

Jony Ive *(Chief Design Officer at Apple)*

www.apple.com

AIRBUS

How ?

- **Reduce model complexity**
- **Better consider expectations**
- **Better address the information**

1 Optimize model structure by **Factorizing Model** elements.

- Make modeling of re-usable elements simple & easy.
- Promote genericity of model elements by better managing specificities and variability.

2 Improve design quality & capitalization, by **Formalizing Design Expectations** into the model.

- Cascade design expectations into the model structure.
- Use model elements to formalize Requirements.
- Ensure full traceability between requirements and with associated model elements

3 Represent the System through **Multiple Views**.

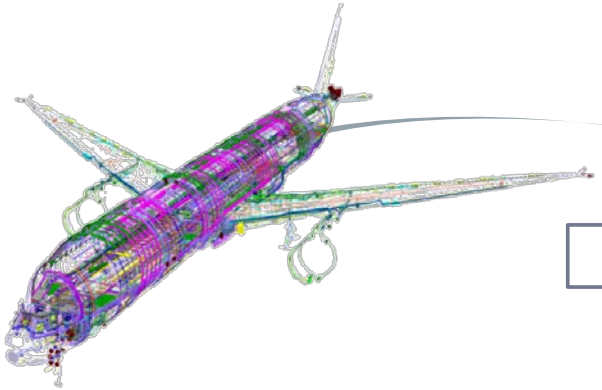
Using most adapted graphical representations to address each concerns in the most efficient way.

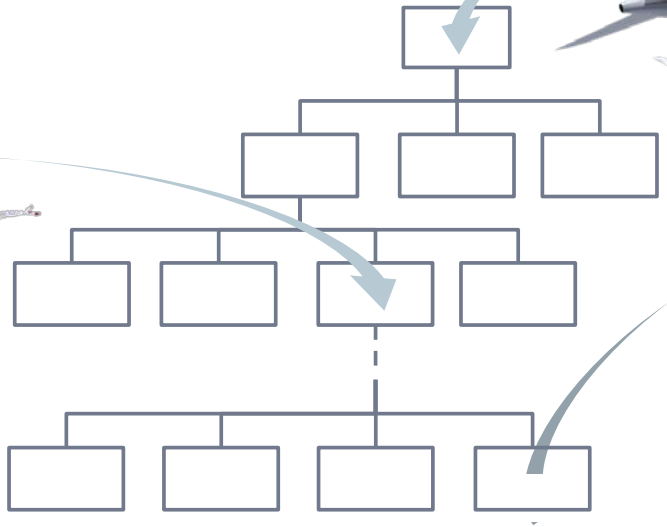
- Using several abstraction levels
- Considering only relevant information
- Using the more adapted description formalism

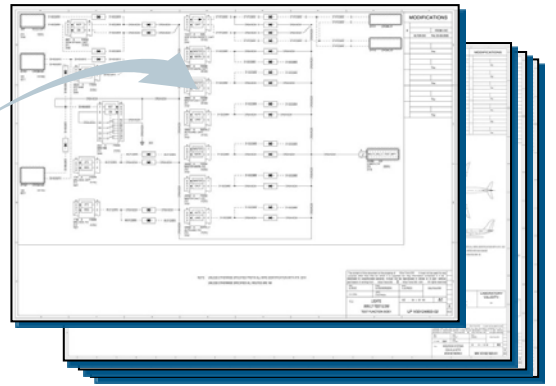
How to address Complexity ?

➔ Break complexity

into sets of simpler parts.







AIRBUS

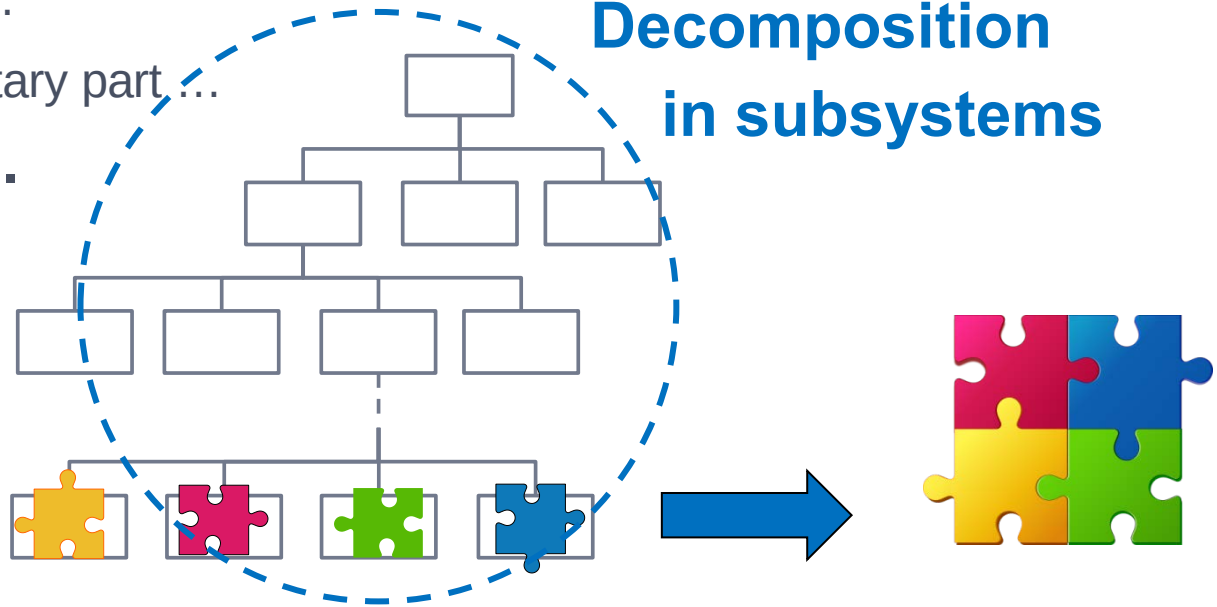
How to address Complexity ?

➔ **Break complexity**
into sets of simpler parts.

➔ **Design** each elementary part ...

➔ manage **integration.**

Decomposition in subsystems



Allow to get Deep understanding of each sub-set.

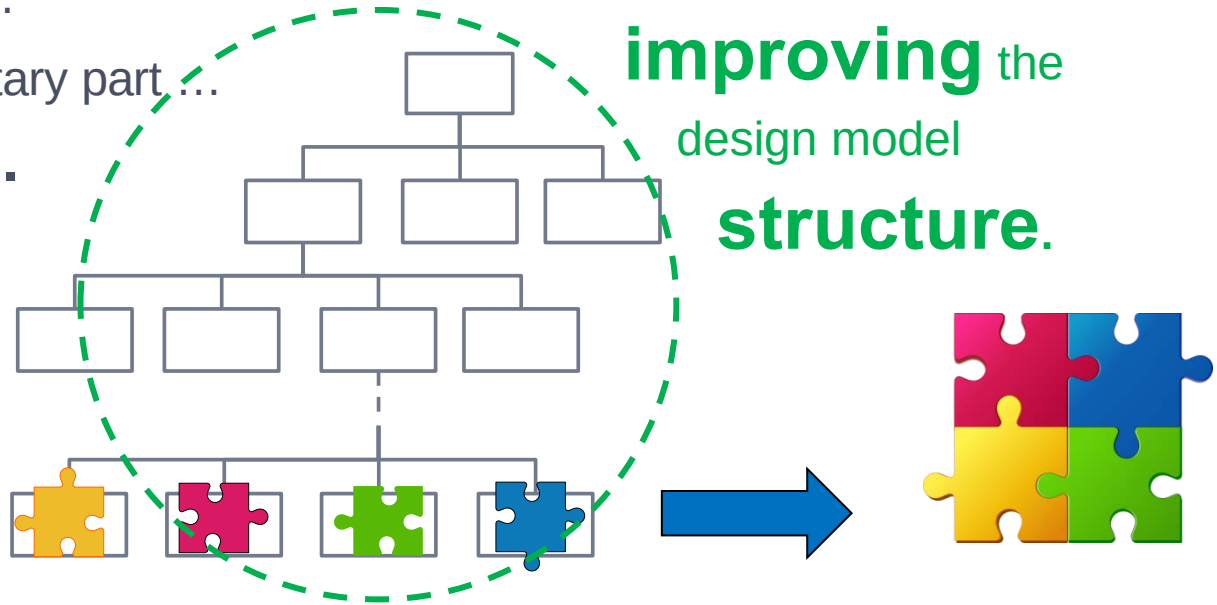
How to address Complexity ? ①

➔ **Break complexity**
into sets of simpler parts.

➔ **Design** each elementary part ...

➔ manage **integration.**

Optimize the breakdown by **improving** the design model **structure**.



Allow to get Deep understanding of each sub-set.

AIRBUS

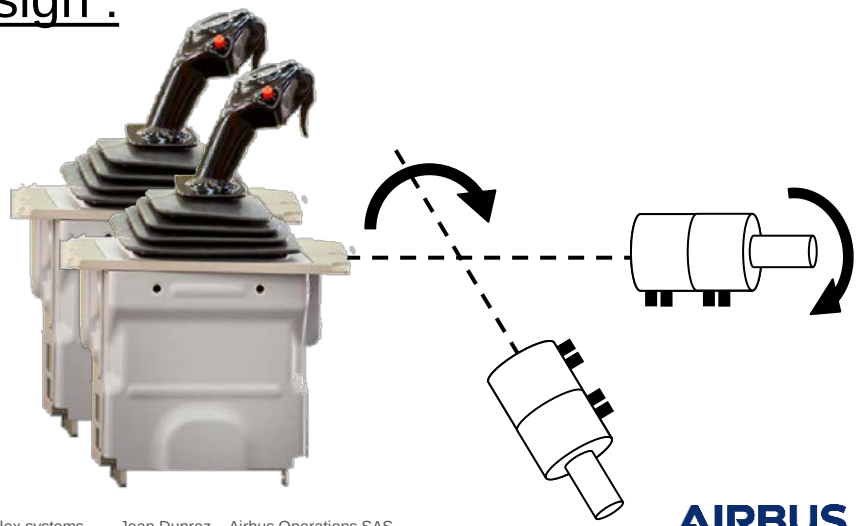
How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

Factorizing the design by : generating re-usable components.

Example of a simplistic Side-Sticks design :

2 side sticks, for the captain & first officer,

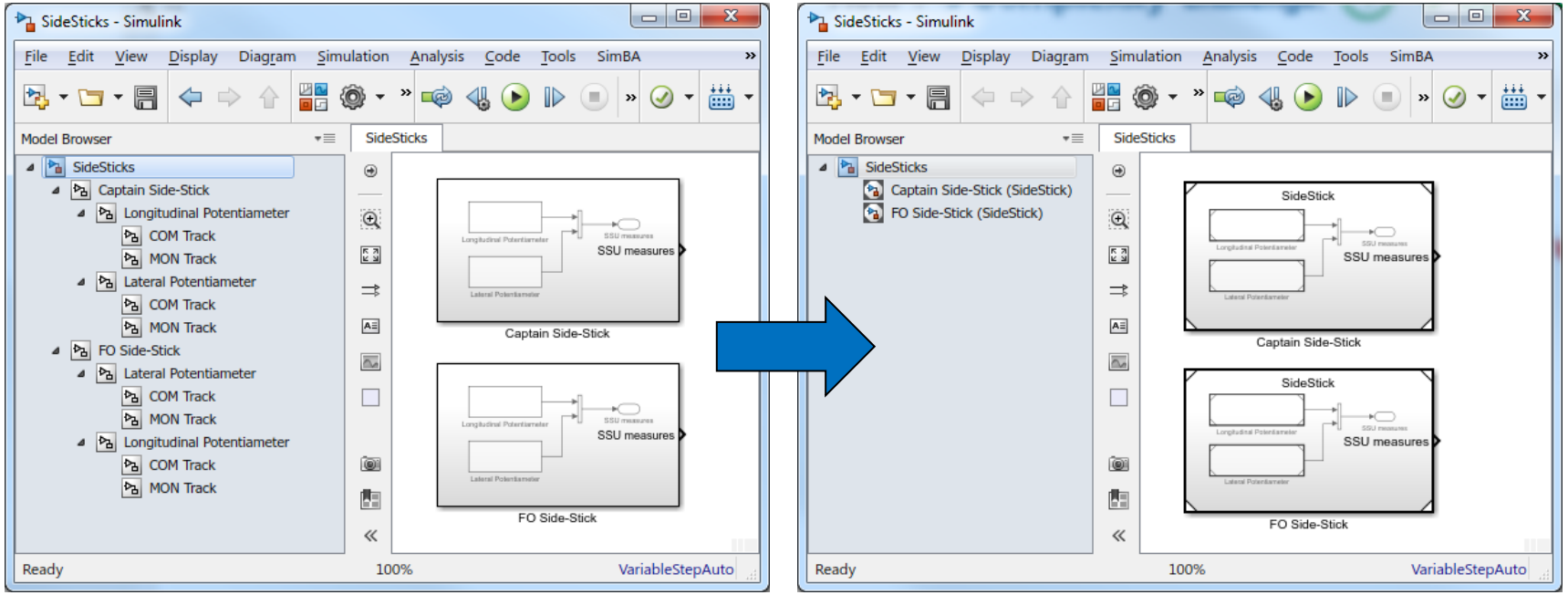
- *with 2 axis : longitudinal & lateral,*
- *with a potentiometer on each axis,*
  - *with a track to measure the angular position,*
  - *and a track for monitoring.*



AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

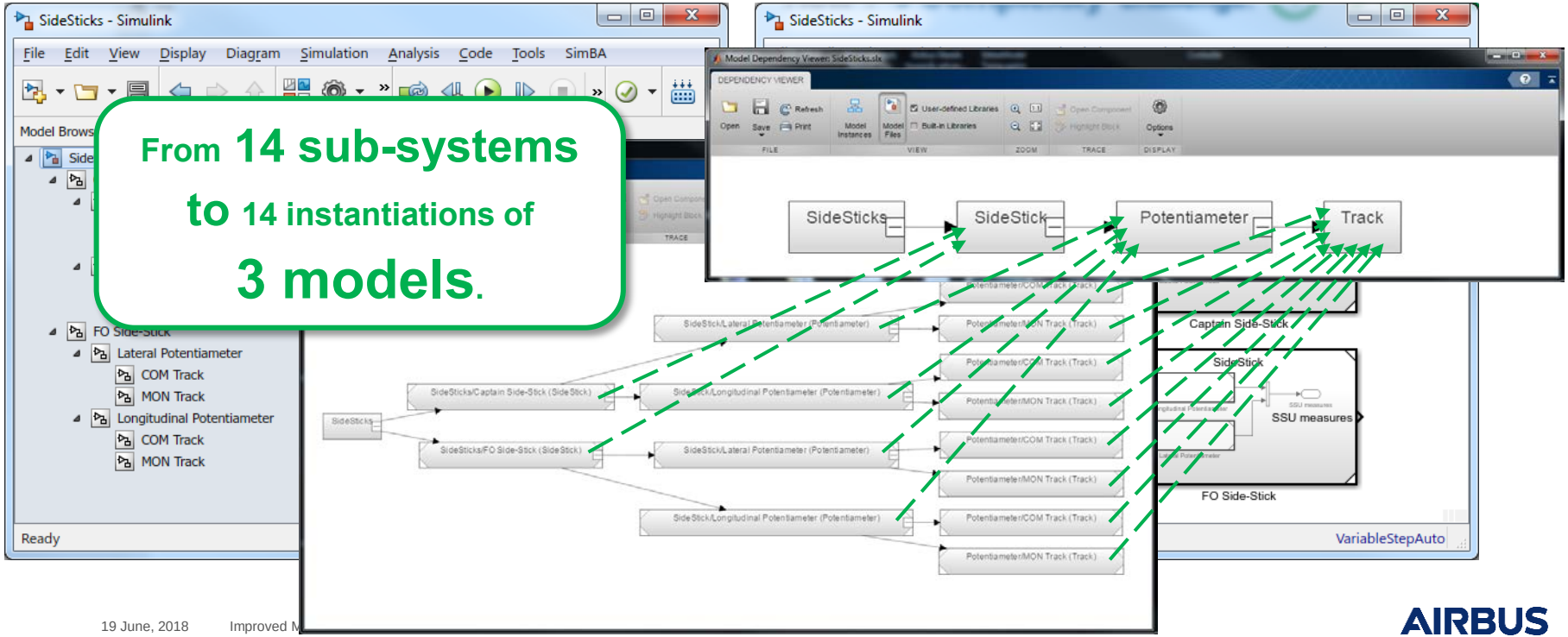
Use of Model Referencing.



AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

Use of Model Referencing.



AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

Factorizing the design by : generating re-usable components.

No duplication of re-usable components.

➔ referencing instead of copy-pasting.

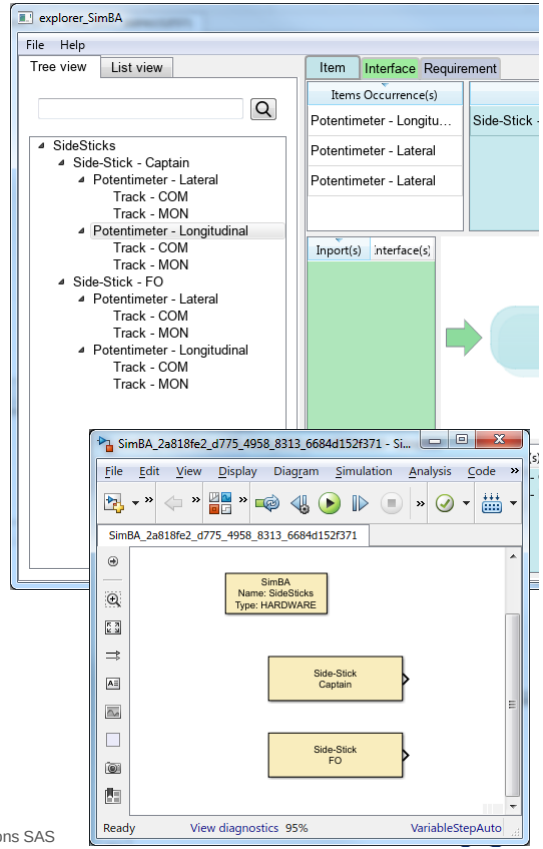
But, model referencing is not as simple and easy to use as subsystem blocks.

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

➔ Make it simple & easy to use.

(thanks to specific customizations)

- **Make modeling through models and model blocks as easy as with Sub-Systems.**
- **Make elements re-using as simple as a Copy / Paste.**
- **Automatically manage interfaces and busses**
  (management of data types, creation and modification of Bus-Objects, propagation of buses modifications to Bus-Selectors, etc…).
- **Make it robust to renaming.**



How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

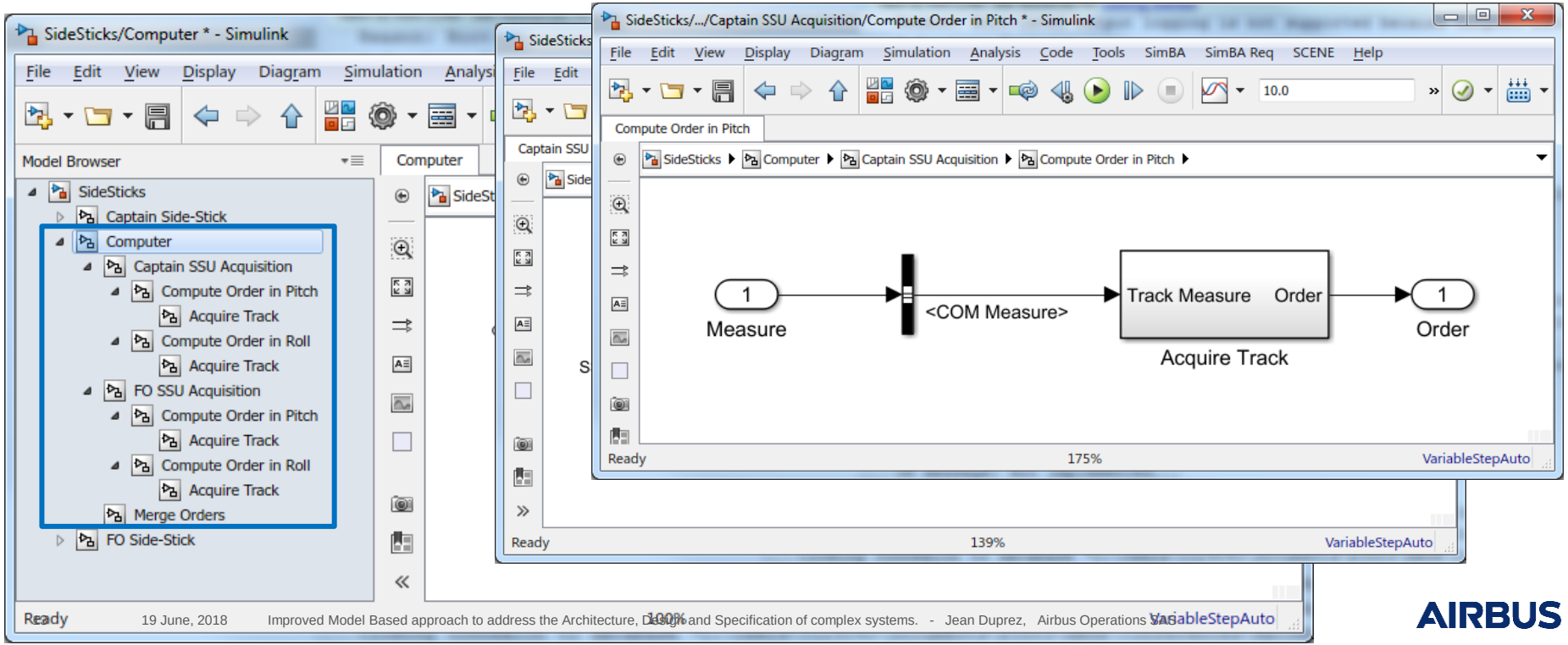
Factorizing the design by : generating re-usable components.

Promote Genericity by advance management of specificities.

➔ managing variability at the most relevant level.

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

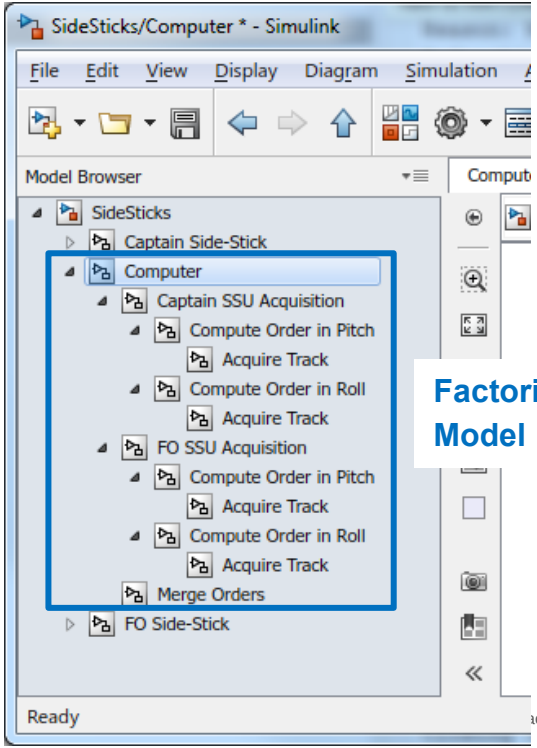
Example of the Side-Sticks Acquisition:



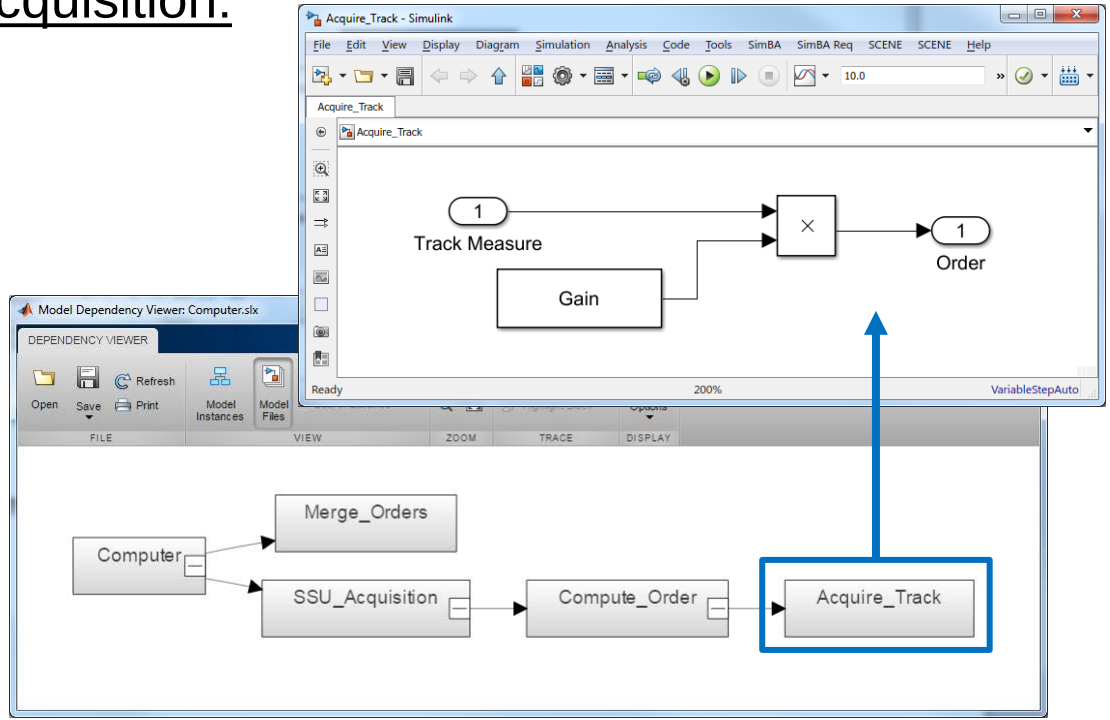
AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

Example of the Side-Sticks Acquisition:



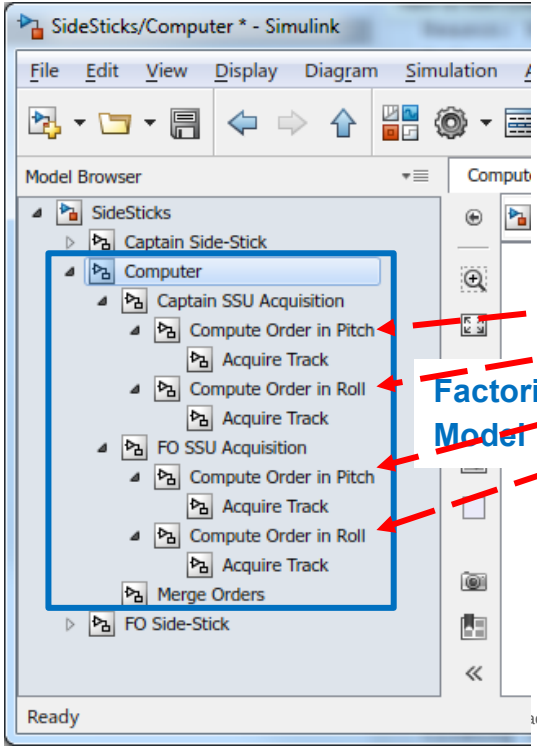
Factorized thanks to Model Referencing.



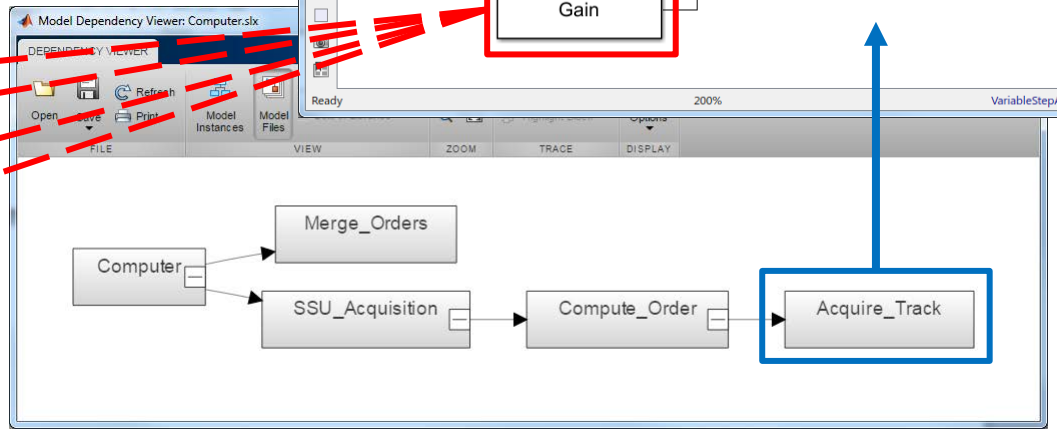
AIRBUS

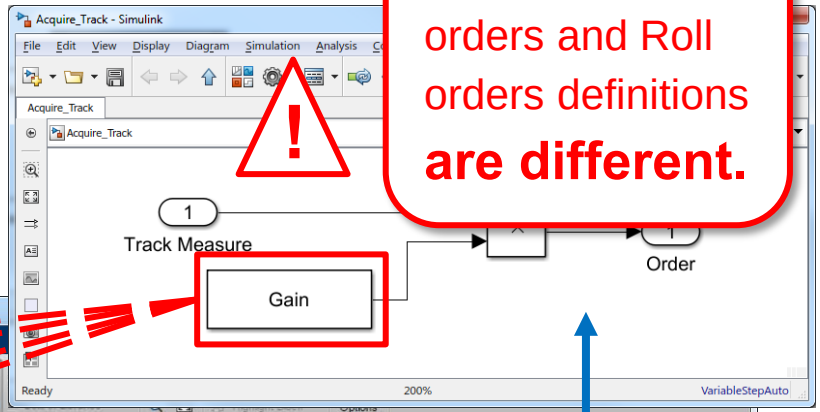
How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

Example of the Side-Sticks Acquisition:



Factorized thanks to Model Referencing.



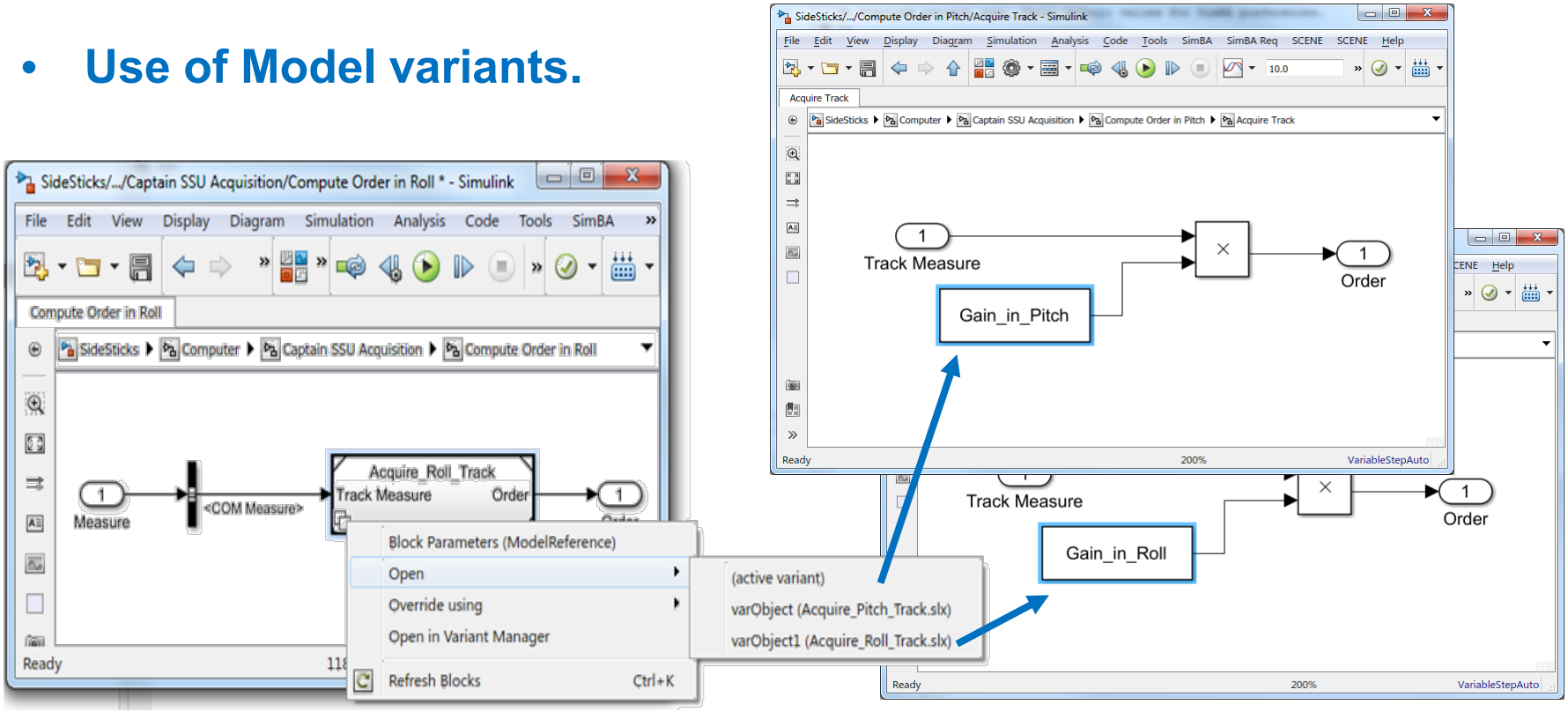


Gains for Pitch orders and Roll orders definitions **are different.**

AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

- **Use of Model variants.**

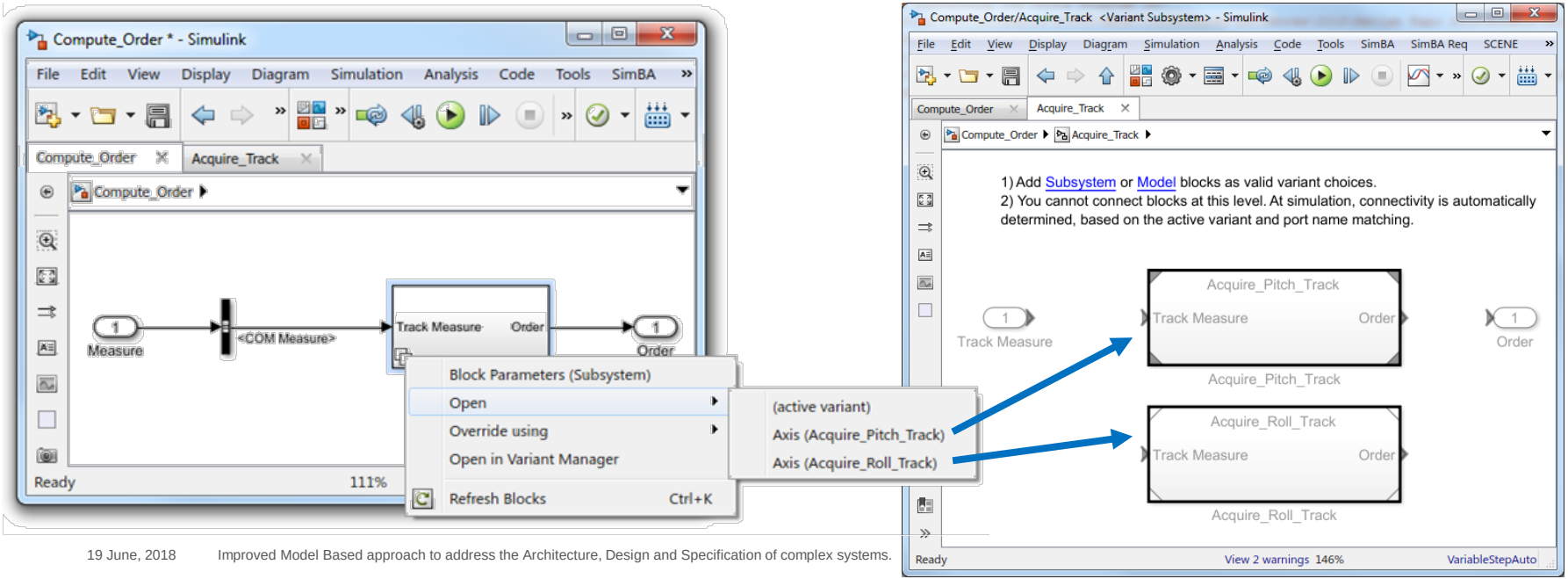


AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

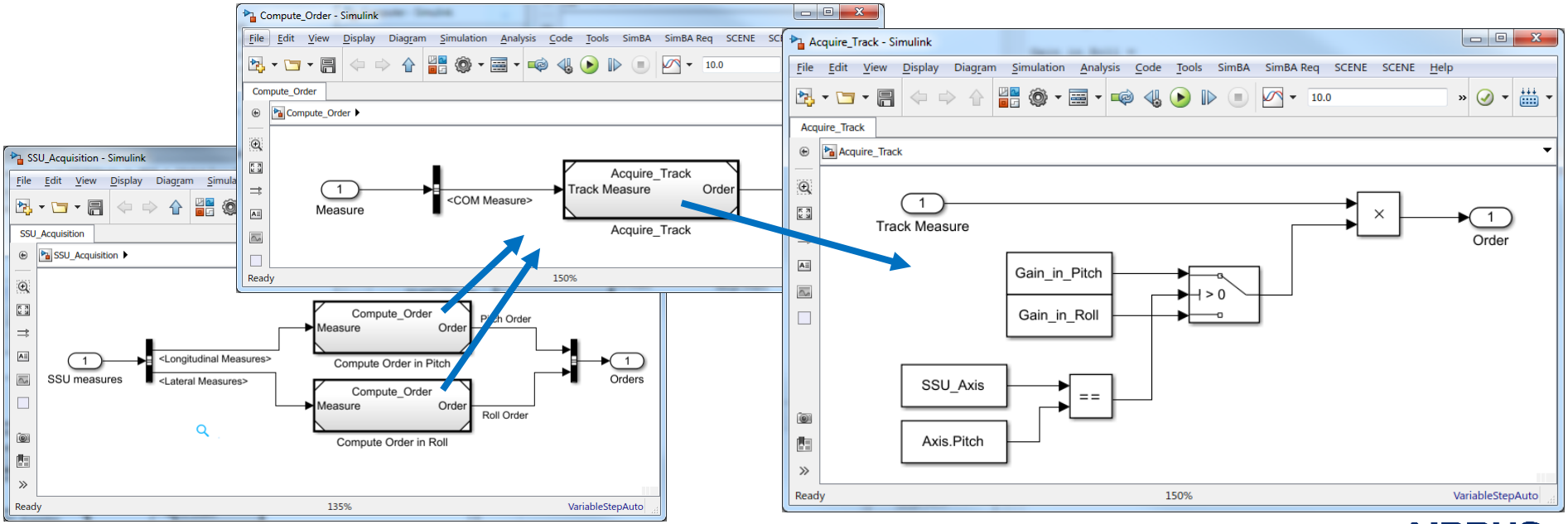
- **Use of Variant Subsystem.**

  ➔ *Describe an “abstract envelope” representing several variants.*



How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

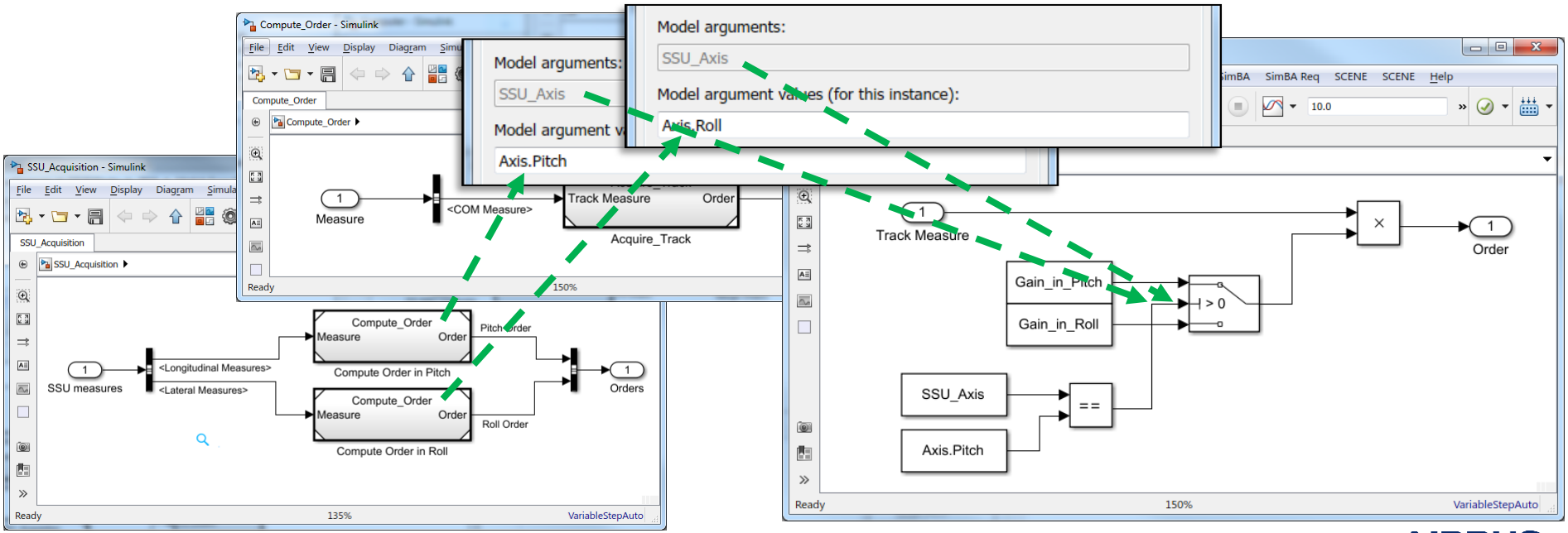
- **Use of model arguments to model specificities while keeping the model generic.**



AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

- **Use of model arguments to model specificities while keeping the model generic.**



AIRBUS

How ? ➔ Complexity challenge. ① ➔ Optimize model structure.

➔ Make it simple & easy to use.

Thanks to customizations to ***create and manage enumerates*** *to:*

- **Manage variants** configuration.
- **Manage local specificities** by passing enumerates as Model Argument.

By capturing and formalizing,

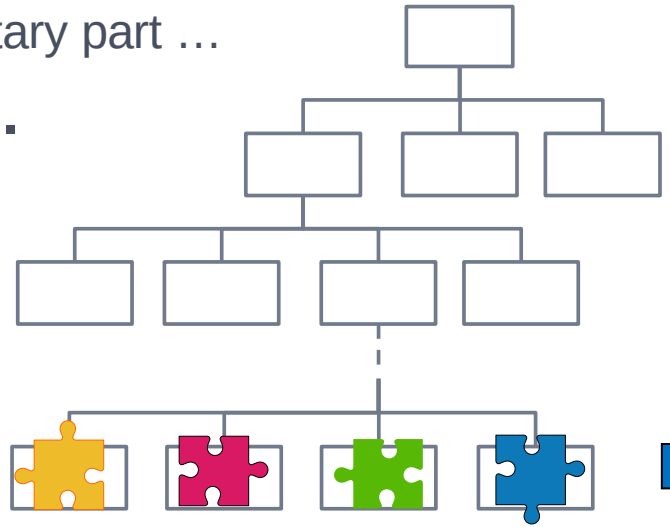
- the **context of use** of each element,
- **design configurations**.

How to address Complexity ?

➔ **Break complexity**
into sets of simpler parts.

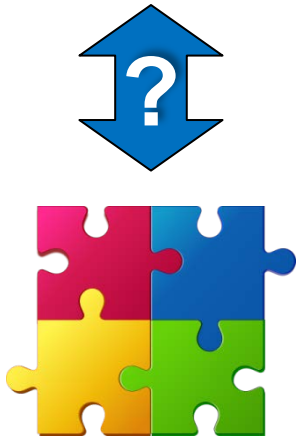
➔ **Design** each elementary part …

➔ manage **integration.**



How to ensure that this giant puzzle will

answer need ?



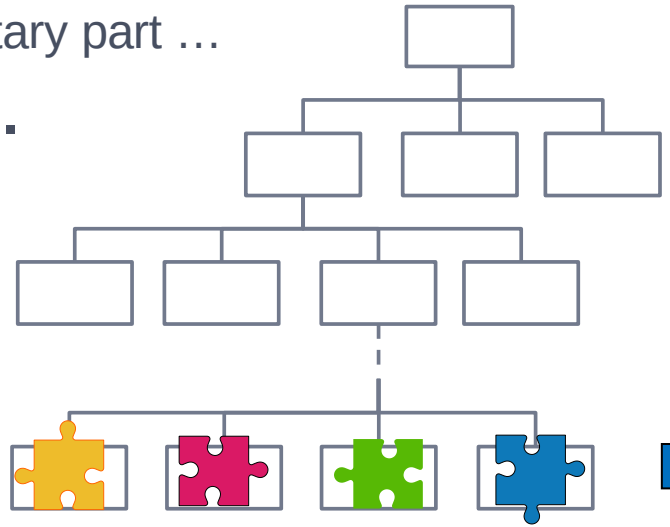
AIRBUS

How to address Complexity ?

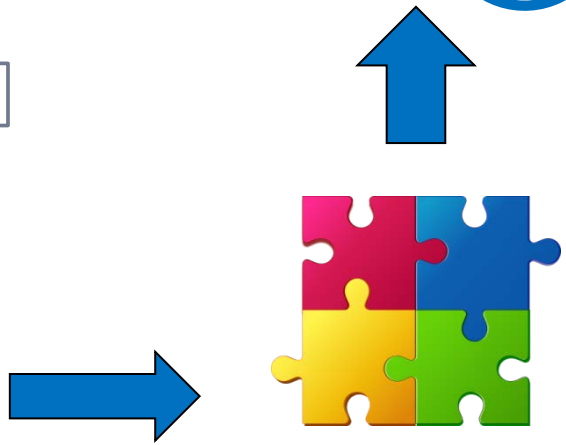
➔ **Break complexity**
into sets of simpler parts.

➔ **Design** each elementary part …

➔ manage **integration.**



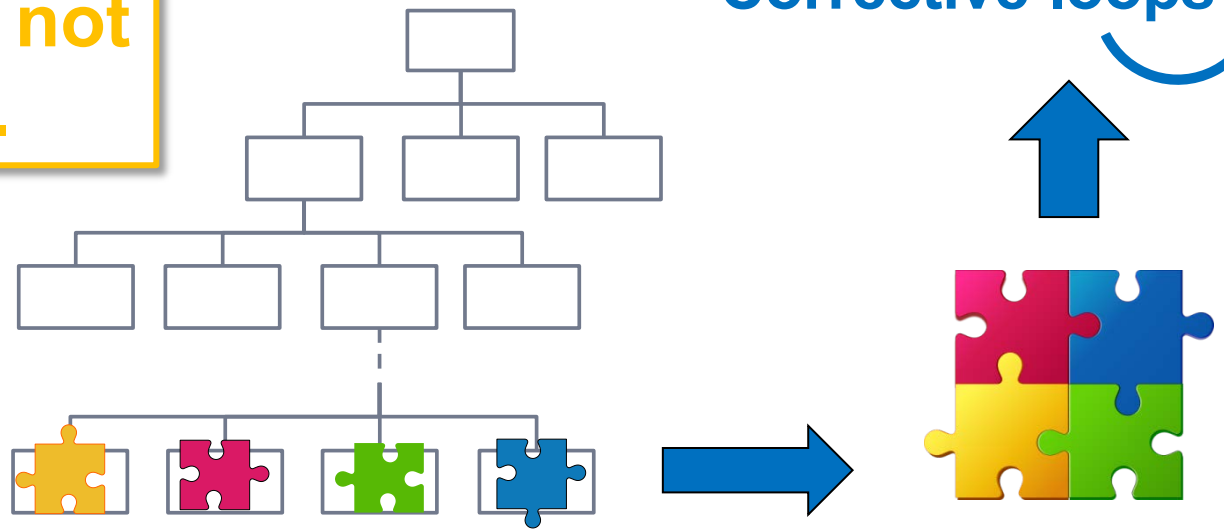
Early Testing &
Corrective loops



How to address Complexity ?

Allow to manage complexity but not to master it.

Early Testing &
Corrective loops

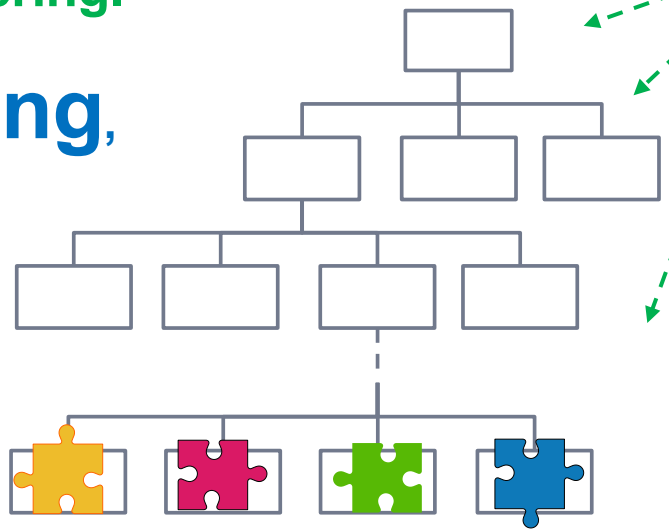


AIRBUS

How to address Complexity ?

② Use of Requirement based System Engineering.

➔ Use of **Modelling**, to **Structure** the Specification,

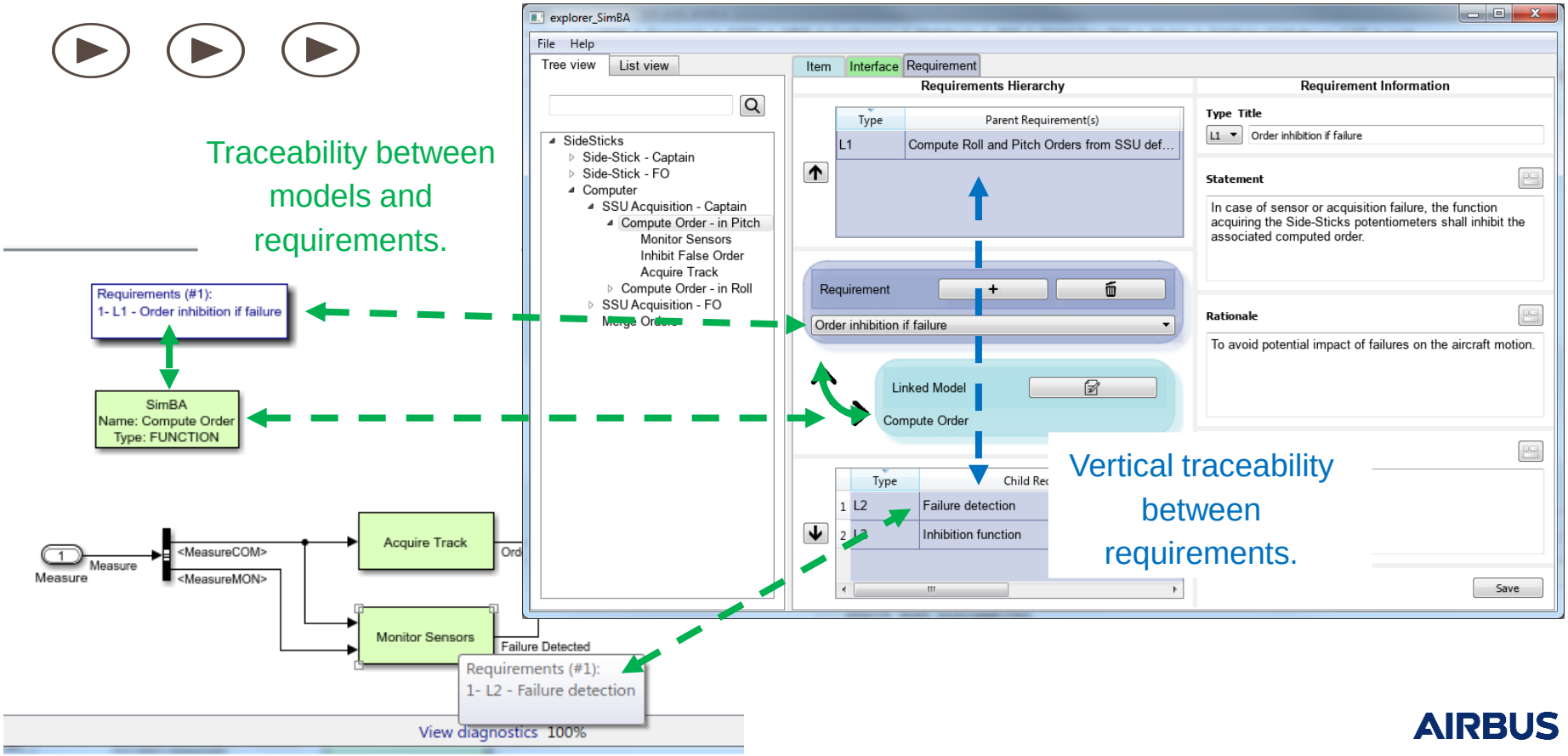


Formalizing expectations at each level.

AIRBUS

How ? ➔ Requirements Engineering. (2) ➔ Use of Requirement.

Traceability between models and requirements.



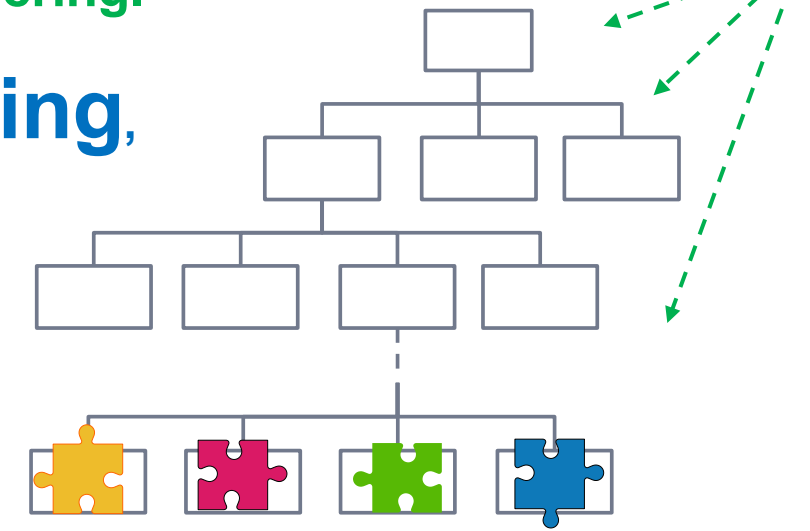
AIRBUS

How to address Complexity ?

Use of Requirement based System Engineering.

➔ Use of **Modelling**,

- to **Structure** the Specification,
- to **Formalize** Requirements.

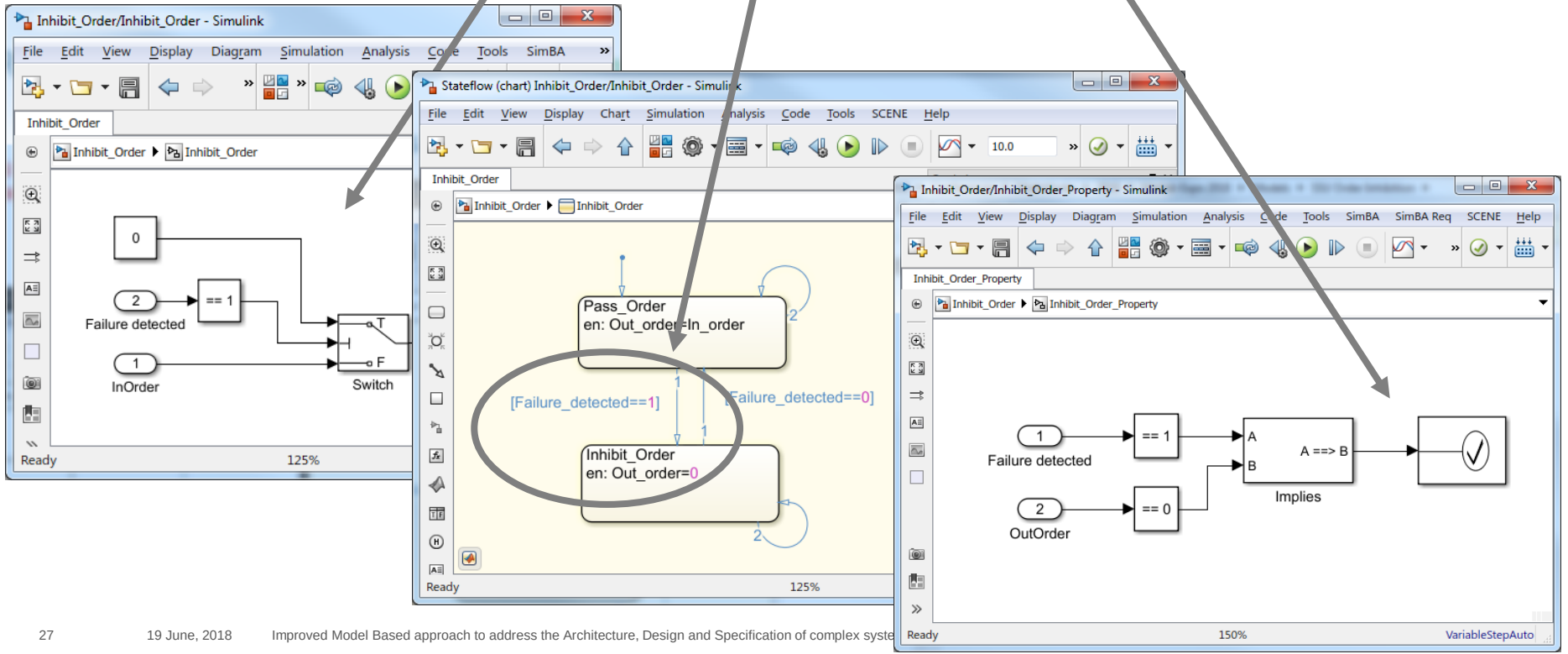


Formalizing expectations at each level.

AIRBUS

How ? ➔ Requirements Engineering. ② ➔ Use of Requirement.

The function shall inhibit the order when a failure is detected.



How ? ➔ Requirements Engineering. ② ➔ Use of Requirement.

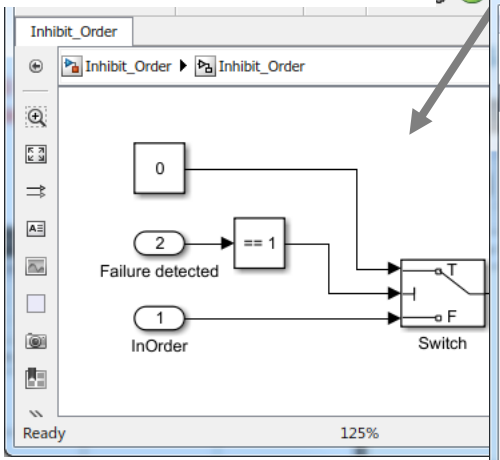
Traceability links

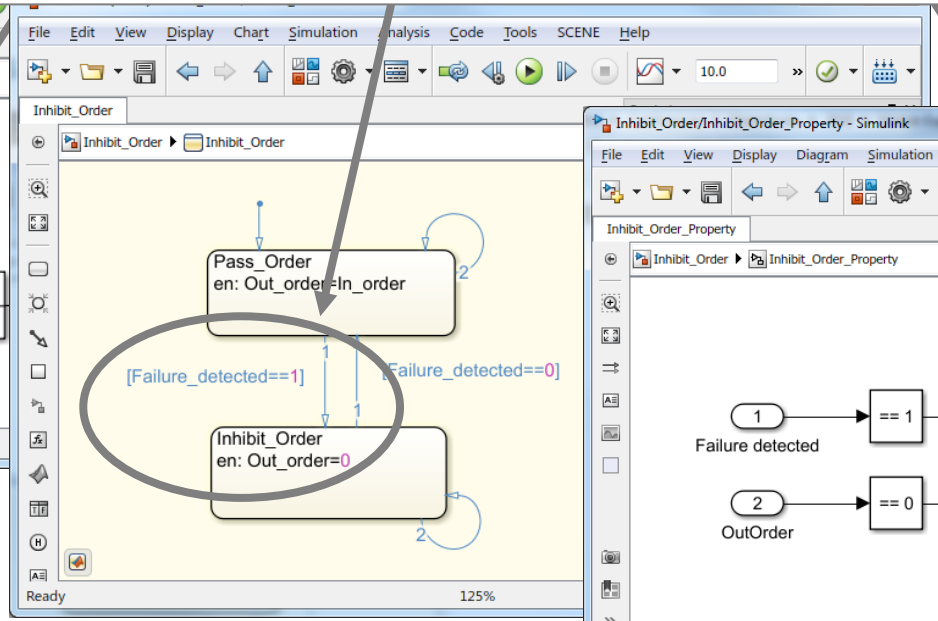
Identifier

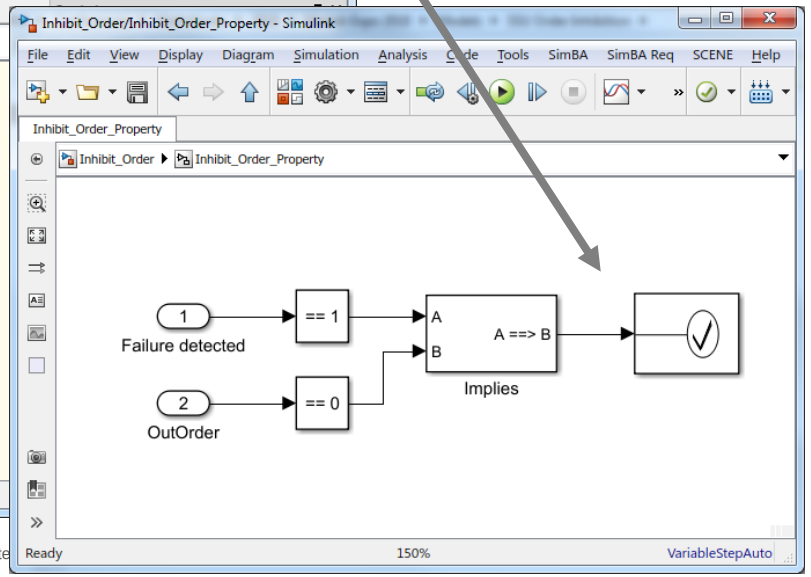
Statement : The function shall inhibit the order when a failure is detected.

Rationale : To avoid potential impact of failures on the aircraft motion.

Additional information : …

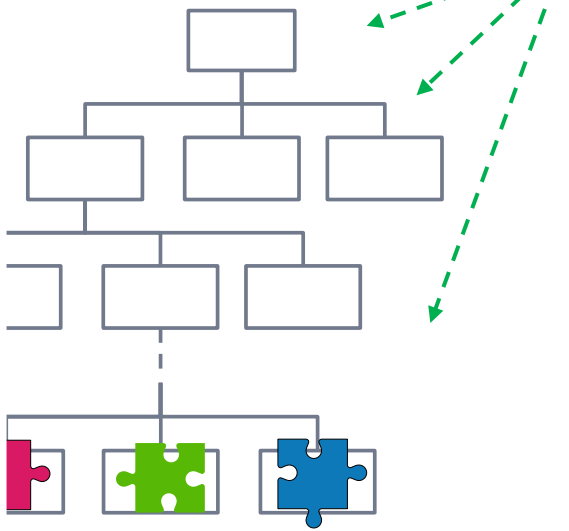






How to address Complexity ?

Allow to well manage expectations cascading, but not to master the design.



Formalizing expectations at each level.

AIRBUS

How to address Complexity ?

How to get deep & full understanding of the overall system ?

3 Use of **Systemic approach.**

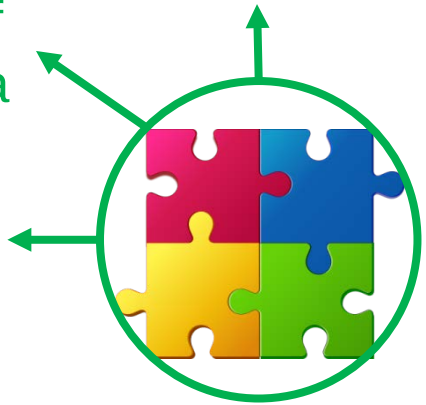
Considering the **system as a whole**.

Focusing on specific aspects the overall system, thanks to :

Abstraction

Extraction of relevant data

Usage of adapted view points.



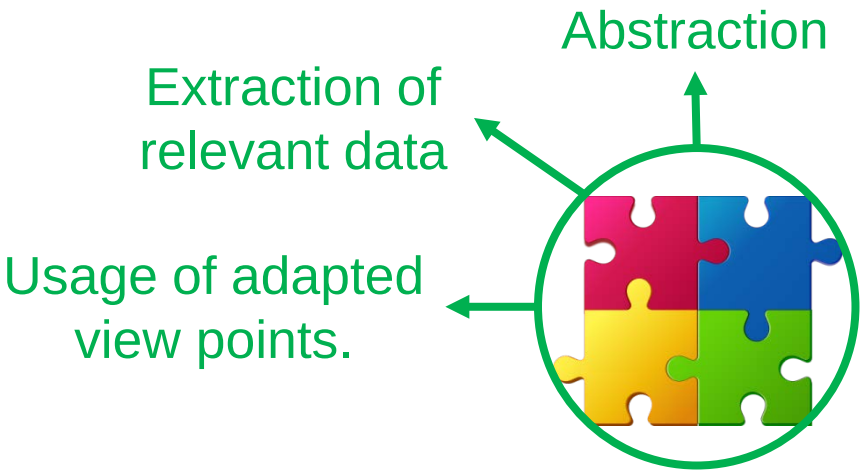
AIRBUS

How to address Complexity ?

Global Model is a set of consistent diagrams or models.

to address

- different scope,
- different abstraction levels,
- different point of views.



How ? ➔ Adapted Graphical representation 3 ➔ Systemic approach.

The use of multiple views allows to

efficiently address each concern.

Data Centric approach:

Considering models as means to

- **visualize, edit and analyze design data.**
- **To simulate associated design behavior**

Conclusion

- **Reduce model complexity**
- **Better consider expectations**
- **Better address the information**

1

Optimize model structure by

Factorizing Model elements.

- Make modeling of re-usable elements simple & easy.
- Promote genericity of model elements by better managing specificities and variability.

2

Improve design quality & capitalization, by

Formalizing Design Expectations

into the model.

- Cascade design expectations into the model structure.
- Use model elements to formalize Requirements.
- Ensure full traceability between requirements and with associated model elements

3

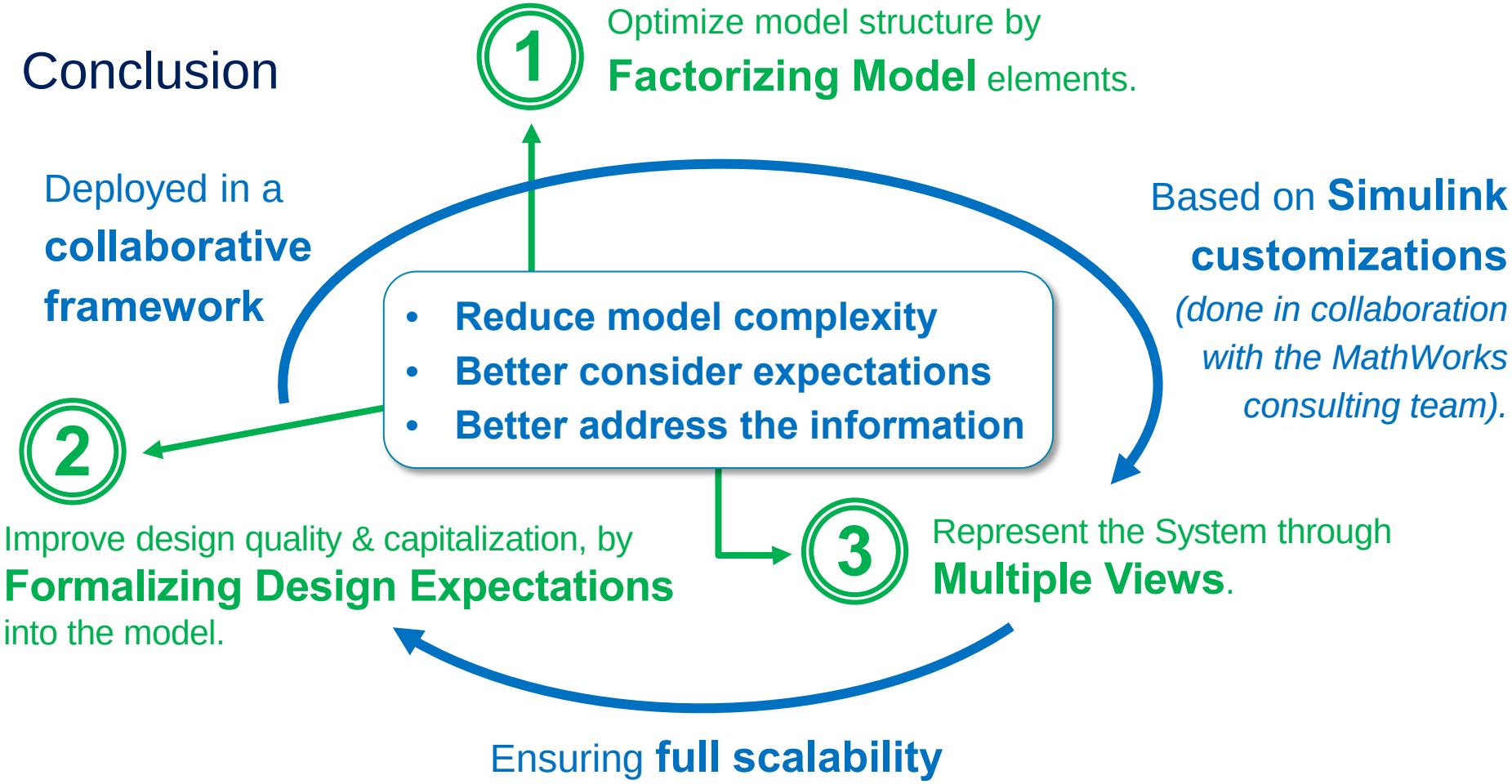
Represent the System through

Multiple Views.

Using most adapted graphical representations to address each concerns in the most efficient way.

- Using several abstraction levels
- Considering only relevant information
- Using the more adapted description formalism

Conclusion



AIRBUS

Conclusion

Presented example:

12 models.
17 model blocks.
37 instances.

Maximum interface decomposition:
6 “sub-interfaces” through 2 levels.

Typical real example:

> 50 models.
> 100 model blocks.
> 500 instances.
Through ~10 levels.

Maximum interface decomposition:
> 4 levels.
> 25 sub-interfaces.

Expected global size:

> 3 000 models.
> 6 000 model blocks.
> 30 000 instances.
Through >10 levels.

> 10 000 requirements.

> 10 parallel active users editing same scope.

Ensuring **full scalability**

Perspectives

1 Optimize model structure by **Factorizing Model** elements.

Initiate the functional analysis and structure the **deriving functional description**.

Allow **checking the design** definition towards expected operational deployment.

Use scenarios to **support testing** and automatic testing.

4 Use **Scenarios modeling** to model operational use cases.

Ease the **capture** of functional **expectations**.

Additional viewpoint to address the system from a procedural point of view.

2 Improve design quality & capitalization, by **Formalizing Design Expectations** into the model.

3 Represent the System through **Multiple Views**.

Thank you

AIRBUS