# Model Based Syst & Sw Engineering

## Wrap up

# Why not?

Need specific **skills**:

- “normal engineers” don’t know modelling languages
- [MB] system engineering not systematically part of education

Big effort to make model; no clear proof of **return on investment**

No clear way to prove that the model is **correct**

Complexity is high so models are too big, the tools **explode**

Tools ergonomy and performance. No evaluation of tools

➔ Not very good reasons…

# Why?

Improve **communication** across all system dimensions. Model is reference for discussion and convergence. Increase confidence.

Justify, trace, assess change impact, generate doc, manage complexity, explore solutions, detect error and overcost early. (Euclid)

Support **integration** (with EDS, of emulators, of physical models)

Support **reuse** (inlc. sw), link engineering to production [functional to product lines], improve products quality.

Must adapt to different needs (flexibility, availability, real-time, rams, embedded sw)

BUT:

Modelling must **solve a problem**. Models must be linked and traced

Move from “textual documents” towards “**documenting in models**”

There are a lot of existing initiatives for many different purposes :

➔ we need “**digital continuity**” (avionics, fluidic, ground, V&V, Rams), remove duplication (e.g. of editors)

# Why?

**Software**:

- Interface models (data model (Proba3), architecture model)
- Dynamics model (behavior and algorithms)

"**Correct by construction**" (model checking, schedulability analysis)

Improve **productivity** (automatic code and test generation, or framework instantiation) and dependability

Ensure **consistency** all along the implementation

Support :

- reference architecture and reuse,
- document generation
- reuse of pre-qualification
- rapid prototyping

Faster, Later, Softer

Dynamic loading of FPGA (**HW/SW co-design**)

Link with **formal methods** (e.g. VDM)

Why not: ½ slide
Why : 2 slides

# Process view

**System process**: functional analysis, refinement, allocation hw-sw, then discipline processes (hw; sw) [A6]

Or 3 parallel processes : system&sw; dependability; FDIR [TAS]

**V&V** must also be covered

Define new **roles** (make clear that we don't steal people's work)

Define models as **deliverables**. No single universal model

Define **reference architecture** (safety/depend/fdir from ASRA, OSRA) with viewpoints, sw architecture with **adaptation points**

➔ Need to define MBSE process within the Standard's frame

➔ Adapt the standards (E10; E40: merge RB & TS?)

# Tool view

Some tools are now **scaling up**

**SysML**: good tool support but lack guidance (too flexible)

- One tool for architecture, others for behaviour and analysers (Fdir)
- One tooling for system, another for software/implementation (A6, Fraunhofer, etc)
- Several tools within software (architecture, data, behavior, model checking)

**Compass** roadmap, compass star, compass integration with others tool incl. matlab

Use tools that “normal engineers” know

Be able to **share** models. Should we have unique metamodel for **interoperability**?

Agile perspective: initial sprint used to agree on methodology, what to exchange, modeling guidelines…

Relate tools to standards, make spin in from outside space

For reuse, **model adaptation mechanisms** must be part of the semantic

Prefer to qualify generated code than tool itself

Domain Specific languages shift domain knowledge into tools

➔ **A lot to do in tooling!**

# Towards harmonisation?

**Too many tools** with various quality level, most unusable industrially

Missing **case studies** to have reference, missing **users** to consolidate & mature

**Open source** can help harmonizing (many sw commercial tools are dead), but long term maintenance?

We need to share: no one can succeed alone!

Harmonisation can rationalize investment: ESA has a role: harmonisation?

➔ **But harmonize at which level, what are we prepared to harmonize?**

- The same (part of ) processes more detailed that E-10/E40?
- The same modelling toolset architecture (e.g. with ModelBus)?
- The same model editors?
- Or only the way we exchange models?
- Do we want to exchange? What?