The Software Factory concept

© hiberus.com

Andreas Jung
Jean-Loup Terraillon
ESA TEC-SWE

European Space Agency

ESA UNCLASSIFIED – For Official Use

Importance of software in the system

- Software implements (more and more of) the **system behaviour**
- System **complexity** increases ➔ software size increases
- **Software schedule** is squeezed within the system schedule
- Software is the last **flexibility** of the system at the end of the life cycle
- Software is a candidate for **subcontracting** policies
- Software touches many parts of the system. It has **interface** everywhere (ground – hardware – avionics – payloads – sensors – actuators – egse – security)
- Software uses a **lot of data** from various system functional chains (centre of gravity, temperature, health status, voltage)
- Software has several **users** (system – AIT – operation)

User needs from Savoir-Faire

FASTER (increase productivity)

- Shorter software development time
- Reduce Verification and Validation effort
- Reduce recurring developments (don’t redevelop recurring software: about 50% of platform software)
- Increase cost-efficiency (more requirements same cost)
- Quality of the product (at least same quality)

LATER (increase reactivity)

- Mitigate the impact of late requirement definition or change
- Optimize flight maintenance
- Simplification and harmonization of FDIR

SOFTER (increase flexibility)

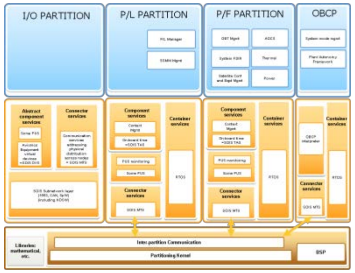
- Support for various system integration strategies (customer-supplier)
- Industrial policy support
- Role of software suppliers (multi-vendor policy)
- Dissemination activities (concept usable by system engineers)
- Future needs

Needs	→ Solutions
– Productivity	➔ Automation (automatic generation, continuous build, automatic regression)
– Complexity	➔ Rely on process Assess feasibility early, verify behaviour
– Reactivity – Flexibility	➔ Architecture (reference architecture, product lines)
– Consistency – of interface, – of data flows, – of use	➔ Configuration (data driven, parameters, missionisation) → system database

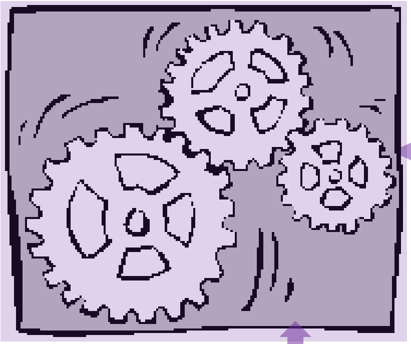
→ *Automation, production line, process, configuration, build*, is the vocabulary of a **FACTORY**

Software factory context

PRODUCT LINE



FLIGHT SOFTWARE

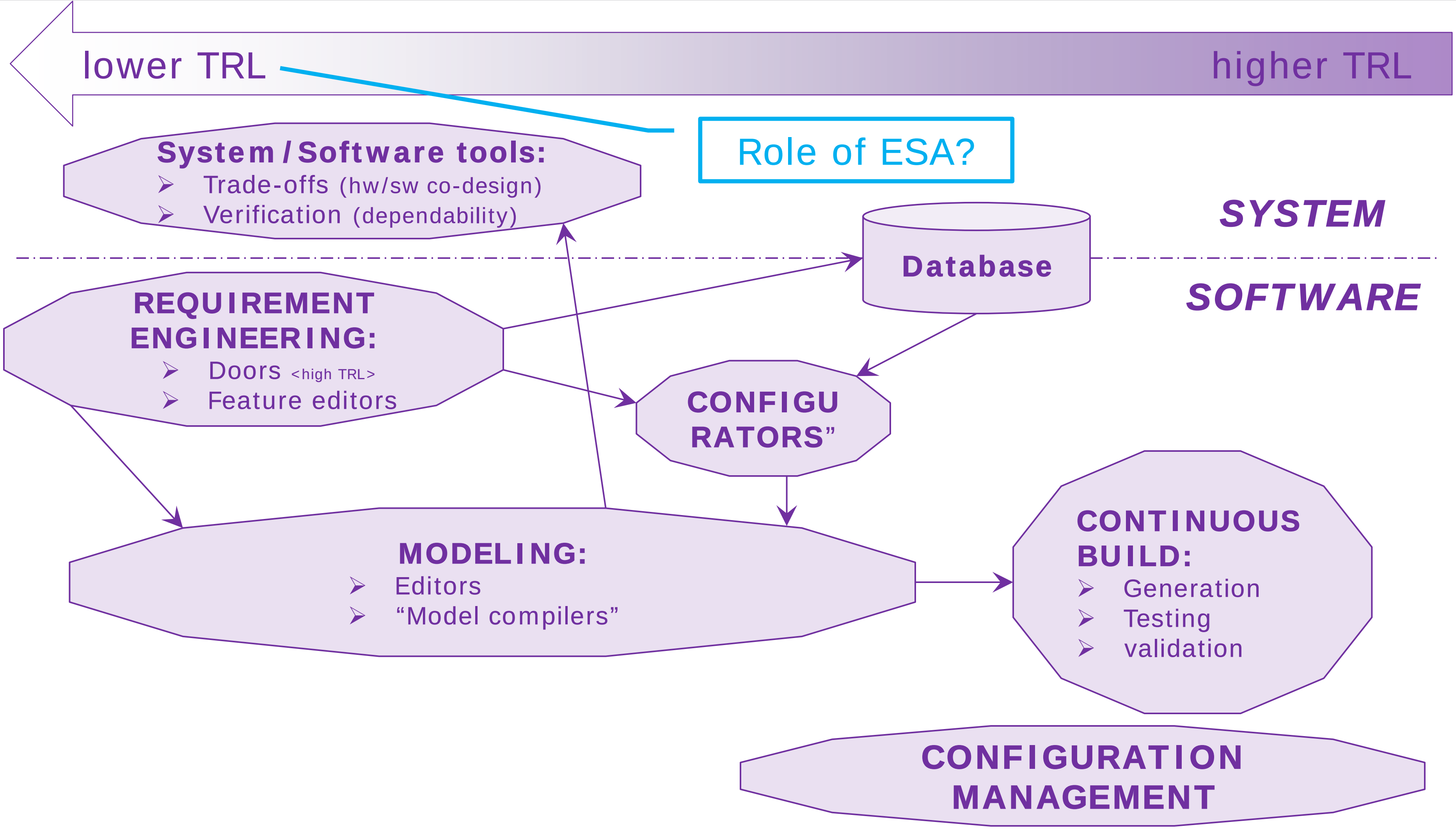


MISSION SPECIFIC configuration, missionisation

DATABASE

FUNCTIONAL VERIFICATION

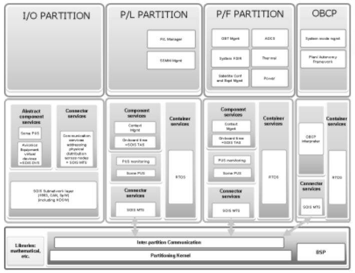
Software factory content



The presentations

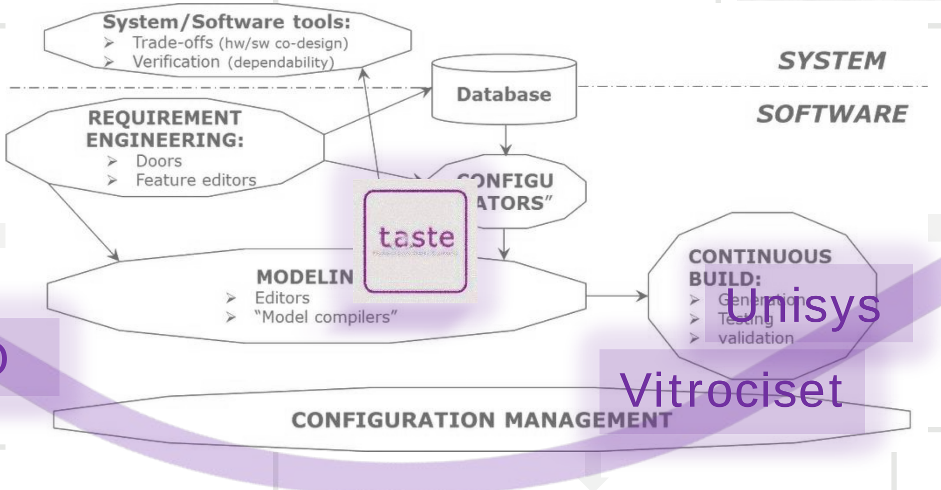
PRODUCT LINE

LERO



AST – TAS - OHB

FLIGHT SOFTWARE



System/Software tools:
- Trade-offs (hw/sw co-design)
- Verification (dependability)

SYSTEM

SOFTWARE

Database

REQUIREMENT ENGINEERING:
- Doors
- Feature editors

CONFIGU ATORS”

taste

MODELIN
- Editors
- “Model compilers”

CONTINUOUS BUILD:
- Generation
- Testing
- validation

CONFIGURATION MANAGEMENT

OBEO

Unisys

Vitrociset

MISSION SPECIFIC
configuration,
missionisation

DATABASE

FUNCTIONAL VERIFICATION

Questions for the round table

- Why should we **automate** software engineering in software factories?

- What are the **preconditions**, the **obstacles** and the **limits** of **automation**?

- Is there a **process model** or life cycle, which is more favourable?

- Is there a **business context** more favourable? Relationship automation/product line.

- What is the **tool support organisation** of software factories?

- Should the **customer** do something to make software factories more efficient?