A TASTE for OVERTURE to keep SLIM

Marcel Verhoef, Maxime Perrotin
European Space Agency,
ESTEC, TEC-SWE, Noordwijk (NL)
Marcel.Verhoef@esa.int

The Future of Overture (as I see it)

1. Within the next year: loose weight!

2. Within the next five years: get more attractive!

3. Within the next decade: lead by example!

Agenda

- European Space Agency in a nutshell (intro)
- Challenges in spacecraft system design
- Model-based system and software engineering (MBSSE)
- The TASTE toolset for software development
- Comparing TASTE to Overture (*caveat: Overture assumed as "known"*)
- A roadmap (mash-up plan) for integrating TASTE and Overture
- Outlook and discussion

European Space Agency

Our mission: *To provide for and promote, for exclusively peaceful purposes, cooperation among European states in space research and technology and their space applications*.

- Space science
- Human spaceflight
- Exploration
- Earth observation
- Launchers
- Navigation
- Telecommunications
- Technology
- Operations

European Space Agency

Austria, Belgium, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Luxembourg, The Netherlands, Norway, Poland, Portugal, Romania, Spain, Sweden, Switzerland, United Kingdom

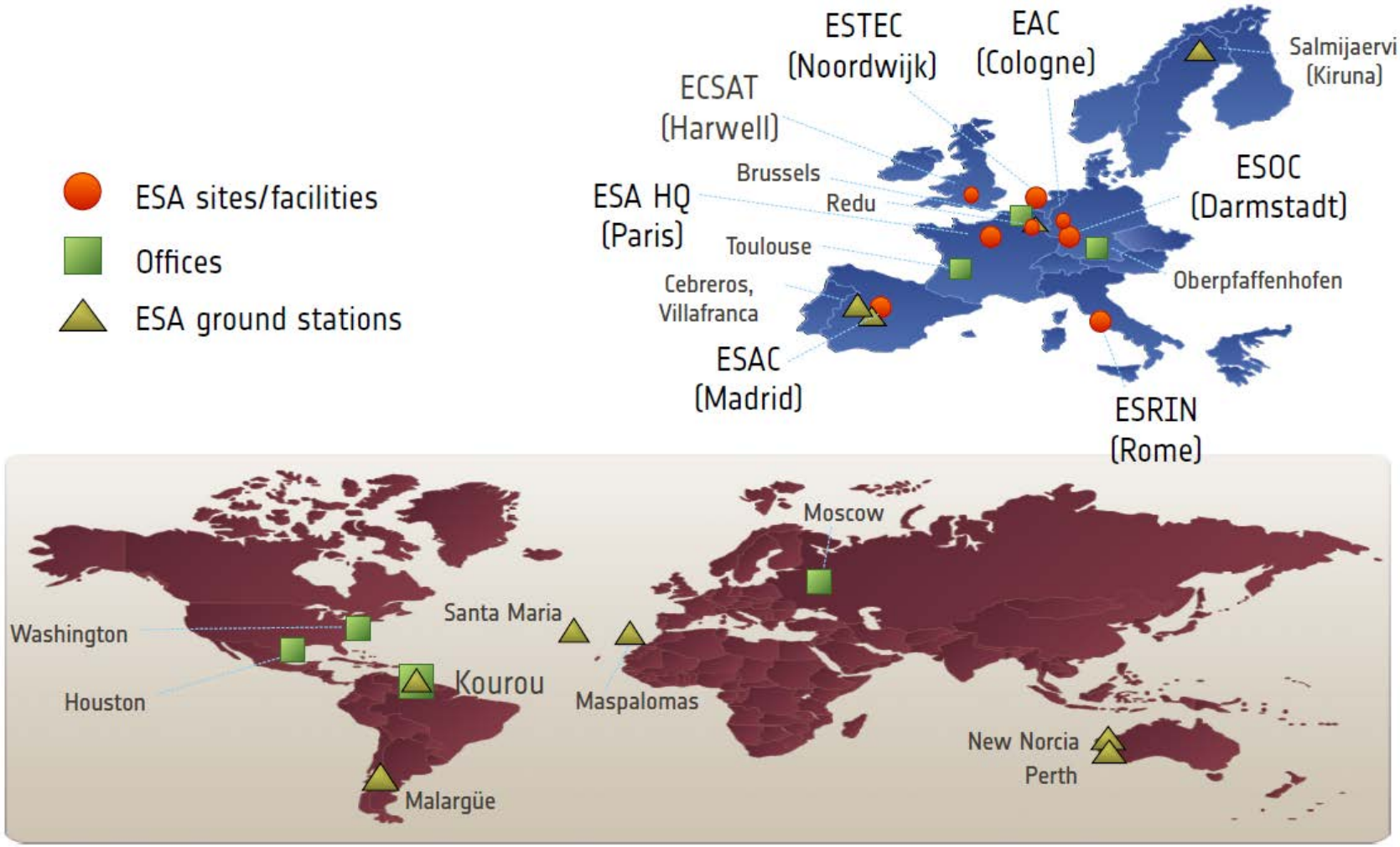
+

Canada

+

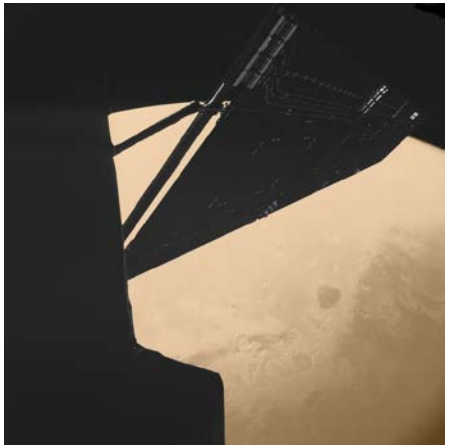
Bulgaria, Cyprus, Latvia, Lithuania, Malta, Slovakia, Slovenia, Croatia

European Space Agency



European Space Agency

- Over 80 satellites designed, tested and operated in flight
- Over 20 scientific satellites currently in operation
- Six types of launchers developed, 200+ launches
- 2015 budget: 4.4 billion Euro
- 85% spend on contracts with industry (“geographic return”)
- 35000 jobs in Europe
- 2200 staff @ESA
- 105 MEu technology support



Meanwhile last night in Kourou (1)

Meanwhile last night in Kourou (2)

Lift-off of VEGA mission VV-05 with Sentinel-2

ESTEC (Noordwijk, NL)

European Space Technology and Research Centre (ESTEC) is the largest technical centre of the European Space Agency, employing 2000 people

- Concurrent Design Facility (mission definition and feasibility analysis)
- Engineering laboratories (materials, radiation, software verification)
- Test center (thermal [solar], acoustic, shock, vibration, EMC)

The space engineering challenge (1)

- Inherent complexity of scientific space missions
- Computers are used to support a wide variety of mission objectives:
    a. Attitude and orbit control
    b. Payload and data handling
    c. Thermal control and health assessment
- Spacecraft are always operated under ground control, but …
- Spacecraft visibility is not continuous and communication is delayed
- Spacecraft operations through telecommand (TC) and telemetry (TM)
- Spacecraft is exposed to extreme environmental conditions
- Mission design and mission duration might span several decades
- Spacecraft maintenance is typically not feasible, but …
- *Software maintenance is (technically) possible!*

The space engineering challenge (2)

- Resilience to faults is a dominant design driver (FDIR, autonomy)
- Dependability *must* ensure that mission objectives are met
- Affects all design artifacts and all life-cycle stages and processes
- Major contributing factor to the cost of space missions
- Analysis is typically performed late in the life-cycle
- Incompatible (and usually informal) domain specific techniques are used
- “fear-based” analysis (and there is *no limit to fear*)
- Acceptable bounds to residual risk are very hard to quantify (ALARP)
- *Complexity converges in the on-board software design and V&V*

Approach promoted by ESA:

- Establish dependability as a system-level discipline (methodology)
- The use of *model-based system and software engineering*

MBSSE at ESA

1. the use of ontologies and formal verification techniques to create a *catalogue of system and software properties*, which forms the basis for correct-by-construction software synthesis and re-use of requirements
2. the use of architecture description languages to *explore system resilience by analyzing explicit fault models* using model checking
3. improve the production of flight software by *integrating well-founded formal technologies* in those parts of the engineering chain where their benefit is clear and the gain is significant
4. the use of time and space partitioning kernels to implement mixed criticality applications on multi-core processors, supported by formal analysis techniques to study *deterministic schedulability in the presence of (WCET) uncertainty*

The TASTE toolset (1)

- consolidated result from (and continued development of) the ESA-led EU-FP6 ASSERT project: **T**he **A**SSERT **S**et of **T**ools for **E**ngineering
  - open-source tool suite for rigorous software engineering
  - aimed at development of heterogeneous embedded systems
  - focus on (but not limited to) space on-board software
  - based on mature (formal) notations with long term support
  - model-centric development with high levels of automation
  - seamless interoperability offers DSL-like approach
  - model synthesis towards wide range of target platforms
  - robust tools maintained by active (but small) community

For more information see http://taste.tuxfamily.org/

The TASTE toolset (2)

The main elements of TASTE are:

1. *Abstract Syntax Notation One* (ASN.1, ITU X.680-X.693)
    - used to describe (abstract) data types (i.e. TC and TM)
    - orthogonal encoding rules for physical representation
    - code generation and run-time support for C and Ada
    - generation of interface documentation and test sets

2. *Architecture Analysis and Design Language* (AADL, SAE AS 5506B)
    - extensible formal textual and graphical notation, used to describe the system logical and physical architecture
    - used to capture avionics hardware components, their communication interfaces and deployment of software artifacts
    - generation of high-integrity (SPARK) Ada code

The TASTE toolset (3)

The main elements of TASTE are (continued):

3. *Specification and Description Language* (SDL, ITU-T Rec. Z.100)
    - formal language to describe *state machines*
    - graphical and textual notation, native support for ASN.1 types
    - model evolution visualized as *message sequence chart* (Z.150)
    - record and playback useful for analysis and testing
    - code generation to (SPARK) Ada

4. *build automation and automatic target deployment*
    - Linux and SMP2 simulation environments
    - RTEMS and Xenomai on (virtualized) QEMU or TSIM
    - RTEMS or Ada-Ravenscar run-time on target hardware

(caveat: also support for Simulink, SCADE, VHDL, Ada, C)

The TASTE toolset (4)

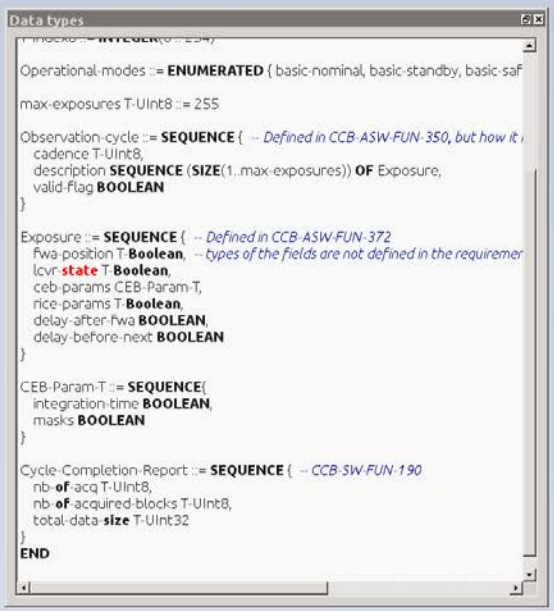
The TASTE development process consist of the following steps:

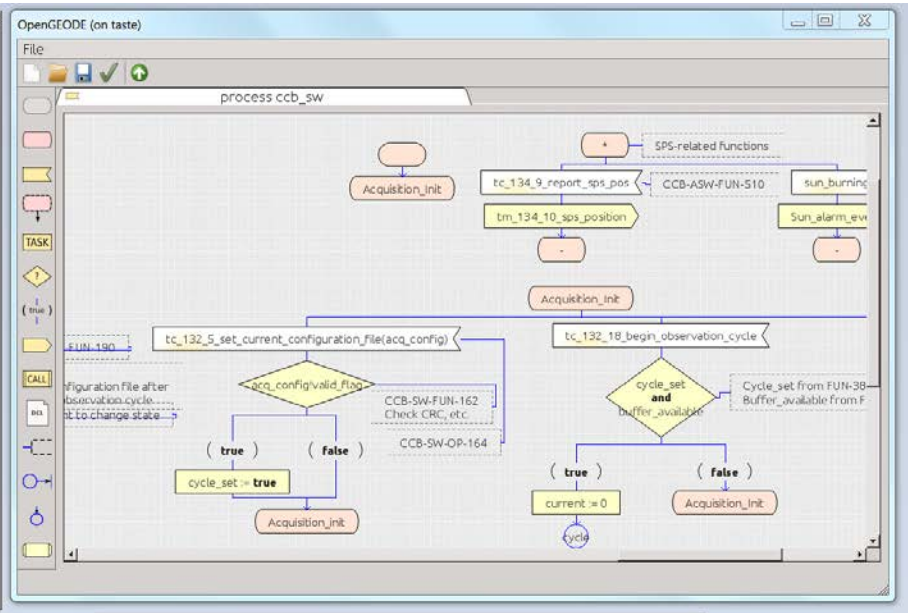
1. describe the system logical architecture (AADL) and interfaces (ASN.1)
2. describe the system behavior (SDL)
3. describe the deployment of functionality on the avionics (AADL)
4. generate code, build the system and download on simulator or target
5. monitor and interact with the system at run-time (test execution)

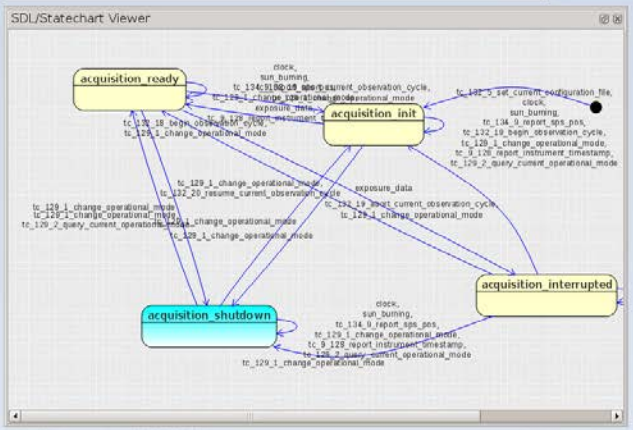
TASTE allows complementary analysis (re-)using the AADL model

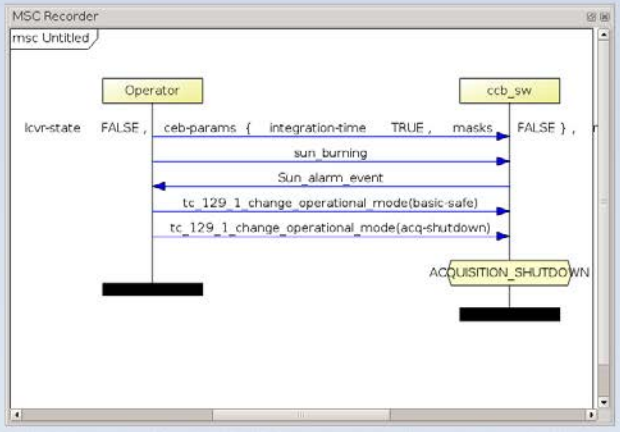
- Schedulability analysis using MAST and CHEDDAR tools
- Use AADL extension capability to specify explicit fault behavior using *System-Level Integrated Modelling* language (SLIM) which can be verified using the (TASTE compatible) FAME tools (nuSMV)

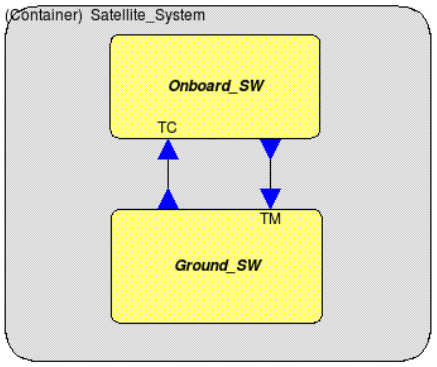
The TASTE toolset (5)





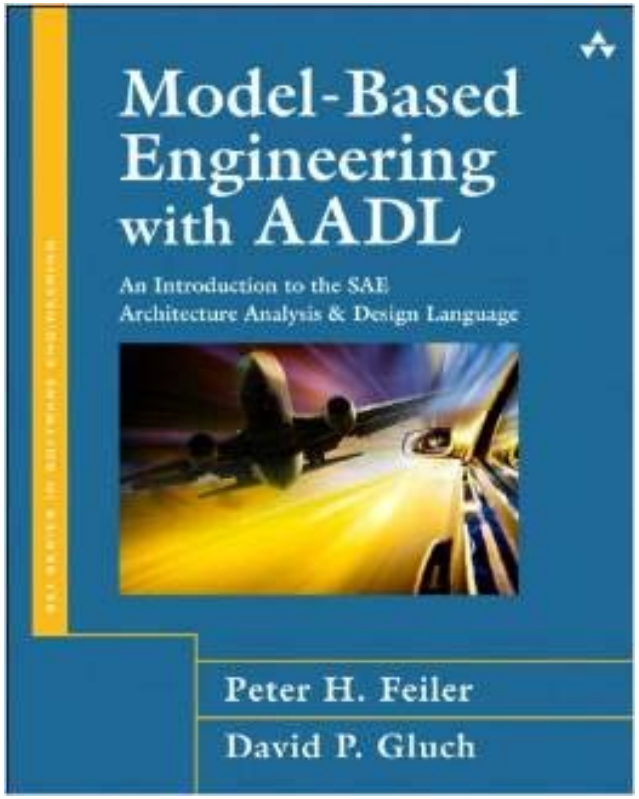






Some words on AADL

- Roots in avionics: META-H (Honeywell [Vestal])
- Now coordinated by CMU-SEI
- Standard sanctioned by SAE
- Used by ARINC
- Well-defined formal language (stronger than SysML, UML-MARTE)
- Extensible
- Active (research) community
- Tool support (Eclipse: OSATE)
- http://www.aadl.info



Comparing TASTE to Overture

TASTE	Overture
1. open-source	1. open-source
2. robust tool set (quality control)	2. robust tool set (quality control)
3. research platform (explore new ideas)	3. research platform (explore new ideas)
4. (co-)simulation to support validation	4. (co-)simulation to support validation
5. focus on rigorous analysis and testing	5. focus on rigorous analysis and testing
6. small but active community	6. small but active community
COMMONALITIES (STRENGTHS)	
7. improving quality of code artifacts	7. early design validation
8. goal is to extend scope towards modeling	8. goal is to extend scope towards synthesis
9. built-in support for state machines	9. state machines require framework
10. weak support for data transformations	10. strong support for data transformations
COMPLEMENTARY (OPPORTUNITIES)	
11. extensible architecture model	11. restricted built-in architecture model
12. implemented in python / QT on Linux	12. Eclipse based (multi platform)
DIFFERENCES (WEAKNESS)	

TASTE – Overture mash-up plan

1. Investigate bi-directional translation between ASN.1 and VDM types and values
    - Linking pin between VDM and SDL worlds

2. Couple OpenGEODE to Overture (using SDL external call paradigm)
    - Allowing complex data transformations in VDM but have the state machine model in SDL (best of both worlds)

3. Exploit the same scheme to link to Crescendo models
    - providing co-simulations with a rich notion of state machines

4. Support SPARK-ADA or MISRA-C code generation from Overture
    - reuse TASTE build management infrastructure to deploy

The Future of Overture (as I see it)

1. Within the next year: loose weight!
   - *Replace VDM-RT system class by a full blown AADL model*
   - *Fix the broken OO mechanisms in VDM++*
   - *Dare I ask: how about some core semantics?*
2. Within the next five years: get more attractive!
   - *Find MSc / PhD students to implement mash-up plan*
   - *Overture as a platform for integrating non-VDM techniques*
   - *Adopt / include those cooperating communities (join forces)*
   - *Provide (interactive) on-line teaching: MOOC? Raspberry PI?*
3. Within the next decade: lead by example!
   - *Increase our visibility: active outreach (social media, videos)*
   - *Overture (generated and qualified) code flying in space*