“PI – Workshop”

MFES 2013/2014

5th December 2013

Eric.Verhulst@altreonic.com

Leonel.Braga@altreonic.com

“From Deep Space to Deep Sea”

www.altreonic.com

Altreonic

Content

- **Who I am?**
- **Who's Altreonic?**
  - Formalised Systems and Software Engineering
- **The OpenComRTOS approach**
  - Formally modeled
  - Hubs and packet switching
  - Small code size
  - Virtual Single Processor model
  - Scalability, portability, …
  - Visual Programming
- **The GoedelWorks**
- **Project Proposals**
  - Event B Support
  - Alloy support
  - Frama-C integration

Who I am?

- Former student of Universidade do Minho
- Education
  - 2006 – 2010: Licenciatura em Engenheria Informática
  - 2010 – 2012 : Mestrado em Engenharia Informática
    - Métodos Formais em Engenharia de Software
    - Criptografia e Segurança de Sistemas de Informação
- Living and Working in Leuven (Belgium) since January
- Software Engineer at ***Altreonic***
- Developing ***GoedelWorks***

Content

- Who I am?
- **Who's Altreonic?**
  - Formalised Systems and Software Engineering
- **The OpenComRTOS approach**
  - Formally modeled
  - Hubs and packet switching
  - Small code size
  - Virtual Single Processor model
  - Scalability, portability, …
  - Visual Programming
- **The GoedelWorks**
- **Project Proposals**
  - Event B Support
  - Alloy support
  - Frama-C integration

Who's Altreonic?

- **Eonic Systems (Eric Verhulst) : 1989 – 2001**
  - Parallel RTOS Virtuoso (=> Wind River Systems)
  - Formally rooted in CSP process algebra (Hoare):
    - "pragmatic superset of CSP"
  - Used on targets with up to 12000 nodes

- **Open License Society: 2004 – now:**
  - R&D on systematic Systems and Software Engineering
  - Unified Semantics & Interacting Entities
  - *Formally developed OpenComRTOS*

- ***Altreonic: 2008 – now***
  - *Commercialises and develops OLS results*
  - *Application of tools in products*
  - *EU R@D projects*

Past ESA experience

- **Virtuoso (ex-Eonic Systems):**
  - Distributed Virtuoso RTOS for multi-21020RT boards
    - Using SpW links on Moasic-20
  - Used in several ESA missions
  - Acquired by Wind River Systems in 2001
- **ESA Market research to find alternative:**
  - No real alternative
  - Serious certification issues as well as for COTS as for Open Source RTOS => “Open License” adopted
- **Feasibility study for reconfigurable telecom satellite**
  - proposed 900 Xilinx-2 FPGA for digital payload

Altreonic's System Engineering

- **Systems/software engineering requires common language in all stages for all activities**

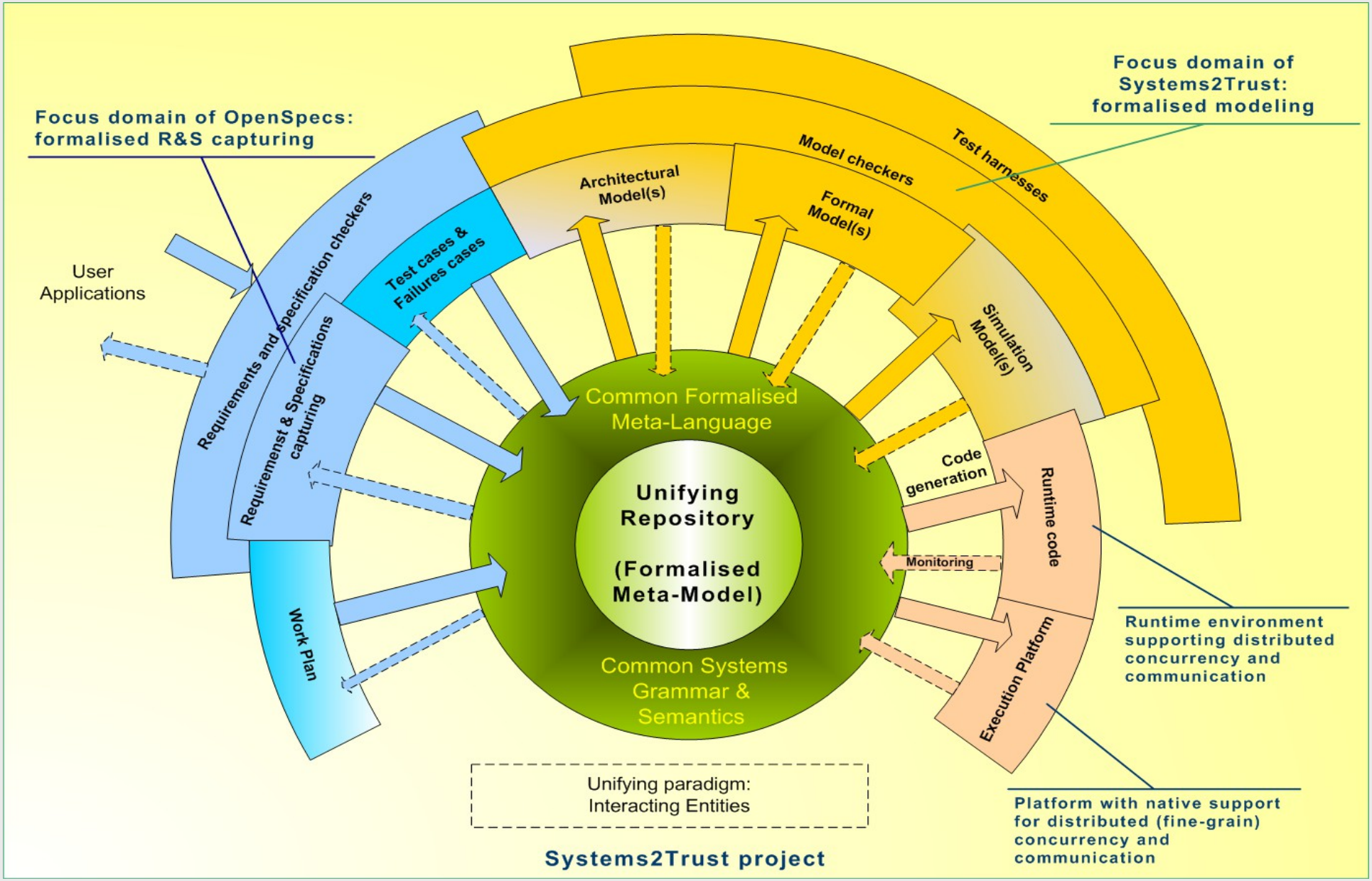
  "Unified semantics"

- **Conquer complexity:**

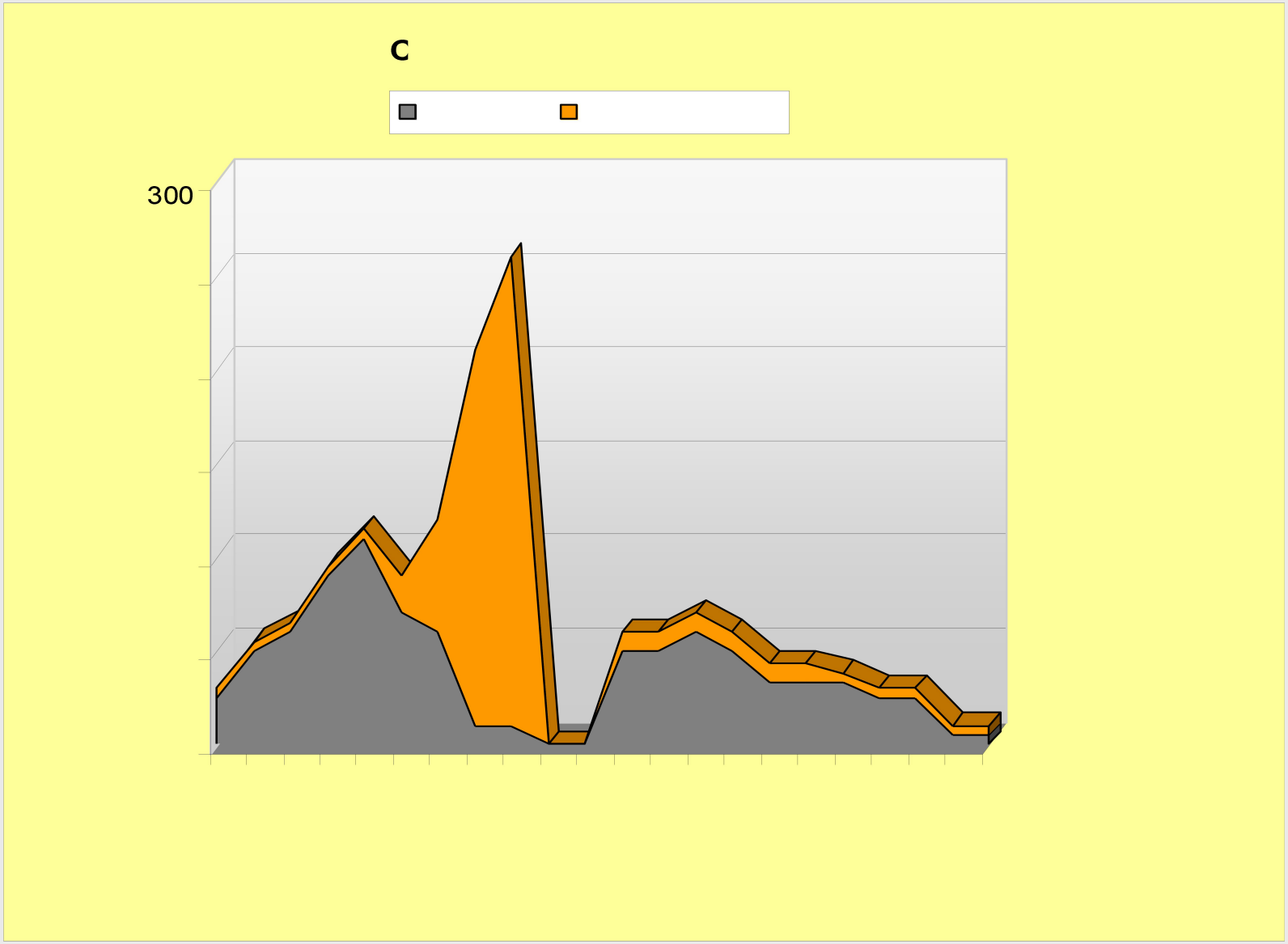
  "Interacting Entities"

- **Different views on same system, same semantics:**
  - Requirements, specifications
  - Checkpoints, issues, change request
  - Modeling & Simulation
  - Failure view, testing view
  - Verification view, validation view
  - Workplan view

Unified Systems/Software engineering



Why FORMAL (ISED) ?



“Formal” gives more for less

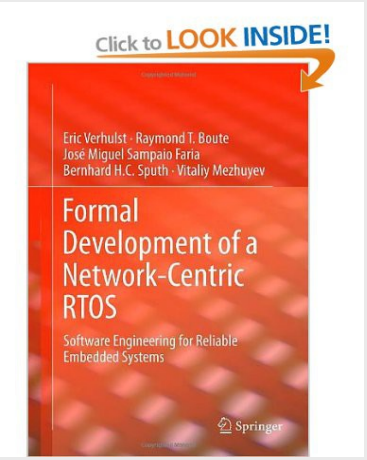
- Formal techniques are just tools in a formalised systems/software engineering process
- Purpose: raising the level of abstraction
- Benefits:
  - Shorter development times
  - Less residual errors, preventing run-away costs
  - Products are:
    - Higher safety, security, usability
    - Cleaner and scalable
    - Smaller => less memory, less power consumption
- Contrary to popular misconception, safety critical designs are also more efficient
- Safer and greener go hand in hand

Content

- Who I am?
- Who's Altreonic?
  - Formalised Systems and Software Engineering
- **The OpenComRTOS approach**
  - Formally modeled
  - Hubs and packet switching
  - Small code size
  - Virtual Single Processor model
  - Scalability, portability, …
  - Visual Programming
- The GoedelWorks
- Project Proposals
  - Event B Support
  - Alloy support
  - Frama-C integration

The OpenComRTOS approach

- **Derived from a unified systems engineering methodology**
- **Two keywords:**
  - **Unified Semantics**
    - use of common “systems grammar”
    - covers requirements, specifications, architecture, runtime, ...
  - **Interacting Entities (models almost any system)**
- **RTOS and embedded systems:**
  - Map very well on “interacting entities”
  - Time and architecture mostly orthogonal
  - Logical model is not communication but “interaction”
- **We have a book:**
  - *“Formal Development of a Network-Centric RTOS: Software Engineering for Reliable Embedded Systems”*
  - ISBN-10: 1441997350
  - ISBN-13: 978-1441997357



The OpenComRTOS project

- **Target systems:**
  - Manycore, multicore, parallel processors, networked systems, including "legacy" processing **nodes** running old (RT)OS
- **Methodology:**
  - Formal modeling and formal verification
- **Architecture:**
  - Target is multi-node, hence communication is system-level issue, not a programmer's concern
  - Scheduling is orthogonal issue
  - An application function = a "task" or a set of "tasks"
    - Composed of sequential "segments"
    - Tasks synchronise and pass data ("interaction")

The OpencomRTOS “HUB”

- Result of formal modeling
- Events, semaphores, FIFOs, Ports, resources, mailbox, memory pools, etc.
  - all variants of a generic HUB
- A HUB has two main functions:
  - Predicate function
    - defines synchronisation conditions
  - Synchronisation function:
    - functional behavior after synchronisation
    - can be anything, including passing data
- All HUBs operate system-wide, but transparently:
  - Virtual Single Processor programming model
- Possibility to create application specific hubs & services
  - New concurrent programming model

Universal packet switching

- **Another new architectural concept in OpenComRTOS is the use of "packets":**
  - Used at all levels
  - Replace service calls, system wide
  - Easy to manipulate in data structures
  - Packet Pools replace memory management

- **Some benefits:**
  - Safety and security
  - No buffer overflow possible
  - Self-throttling
  - Less code, less copying

Unexpected: RTOS 5-10x smaller

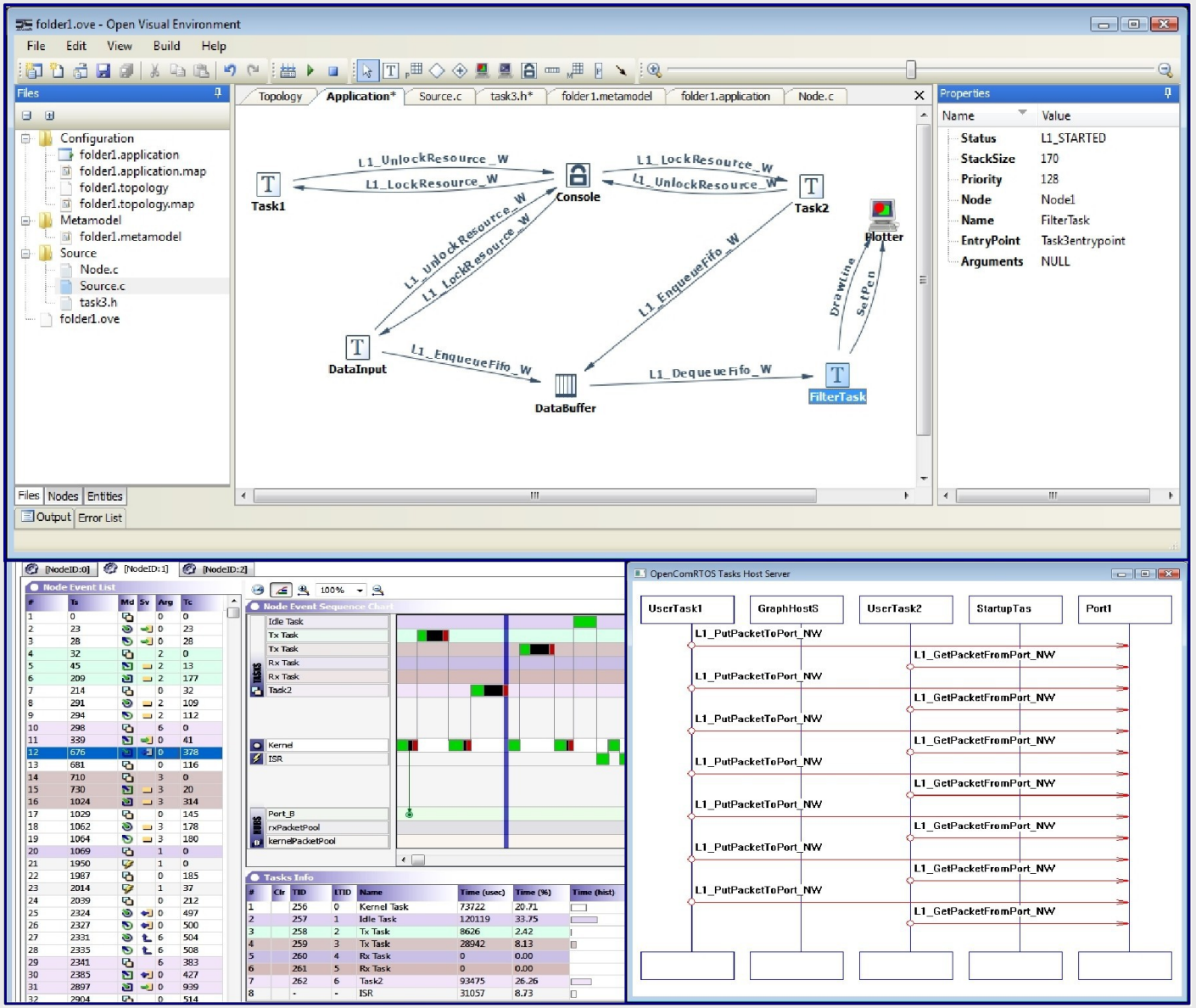
- **Reference is Virtuoso RTOS (ex-Eonic Systems)**

- **New architectures benefits:**
  - Much easier to port
  - Same functionilaty (and more) in 10x less code
  - Smallest size SP: 1 KByte program, 200 bytes of RAM
  - Smallest size MP: 2 KBytes
  - Full version MP: 5 to 10 KBytes

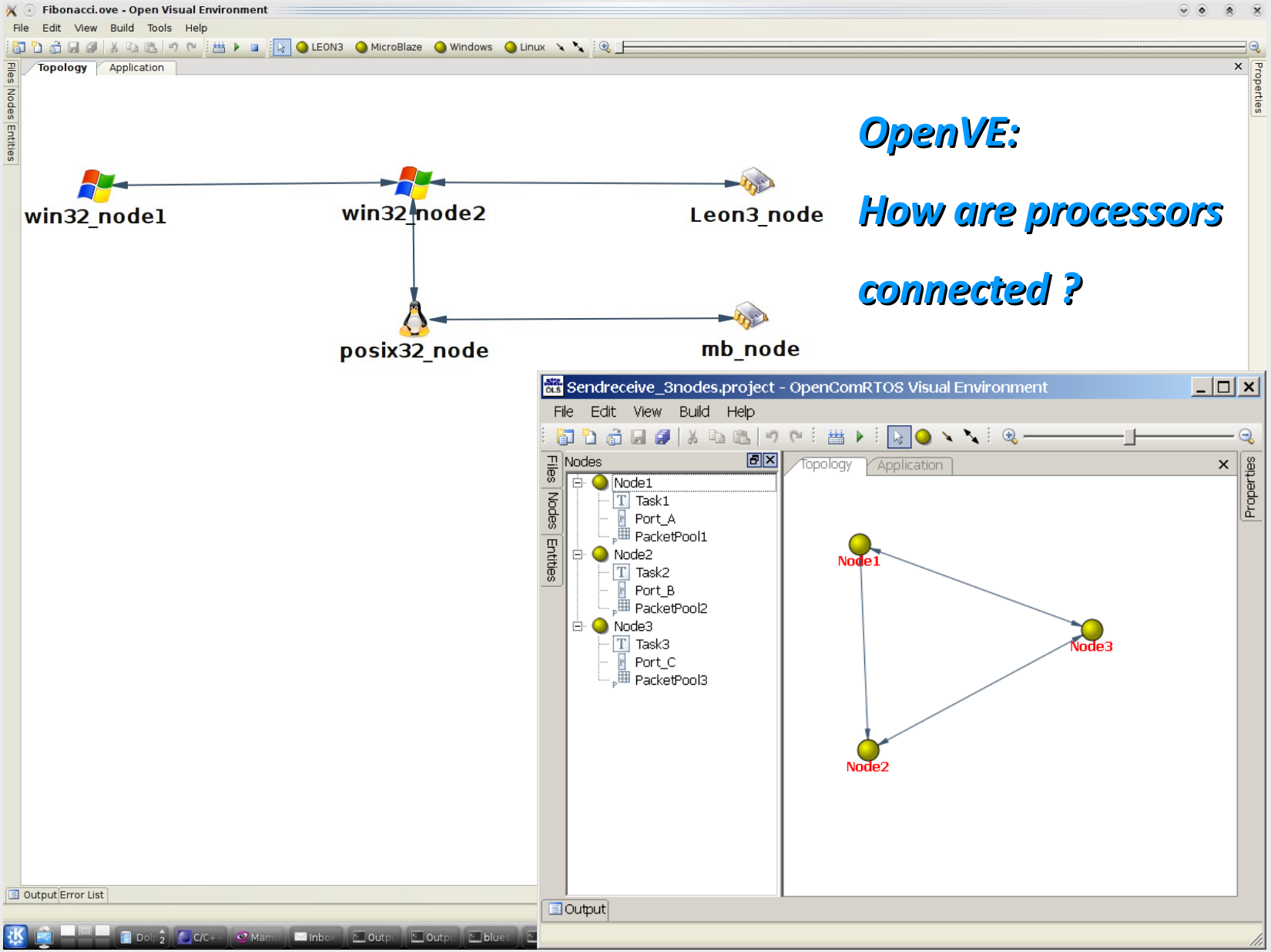
- **Why is small better ?**
  - Much better performance (less instructions)
  - Frees up more fast internal memory
  - Easier to verify and modify

- **Architecture allows new services without changing the RTOS kernel task!**

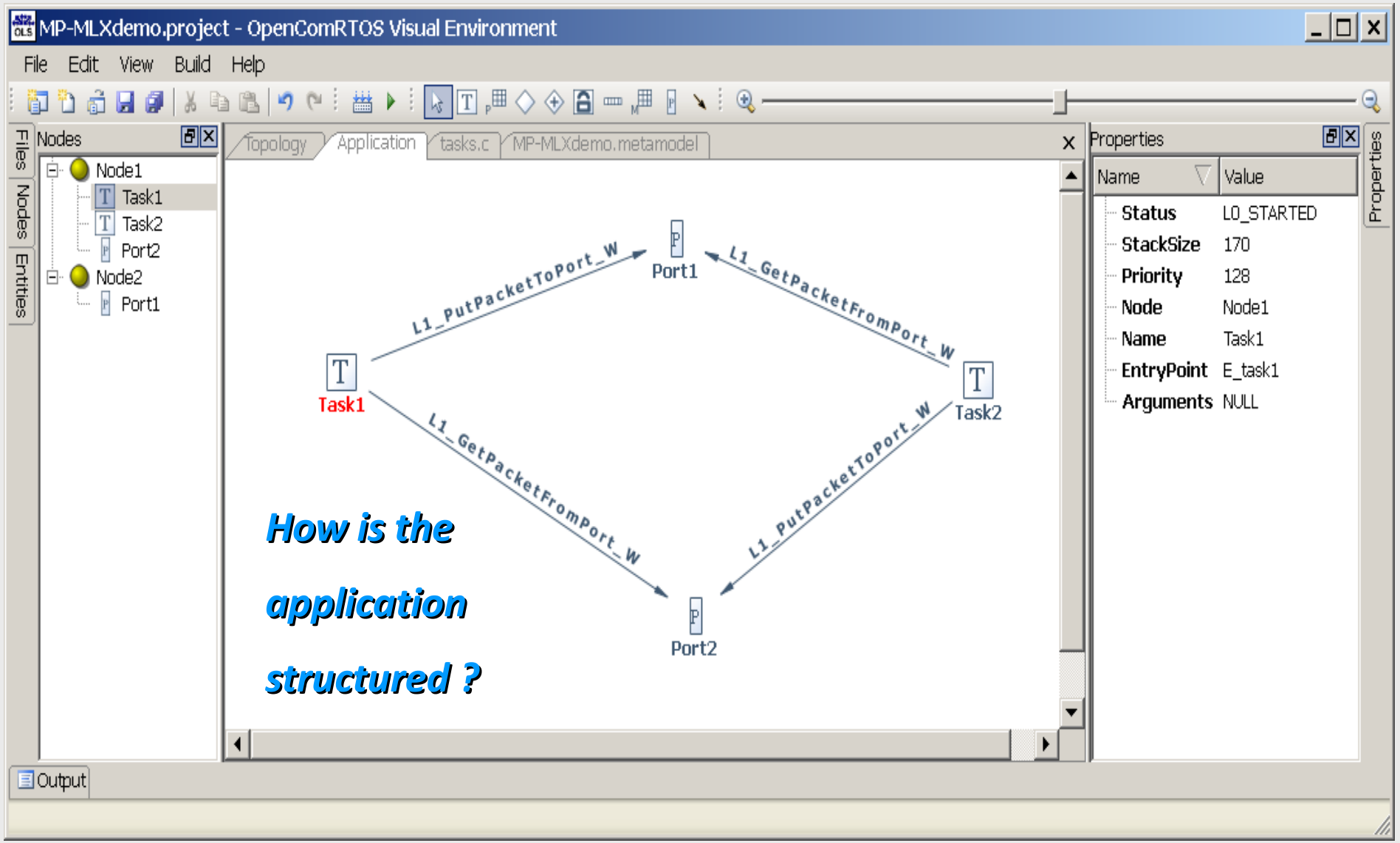
The OpenComRTOS Tools



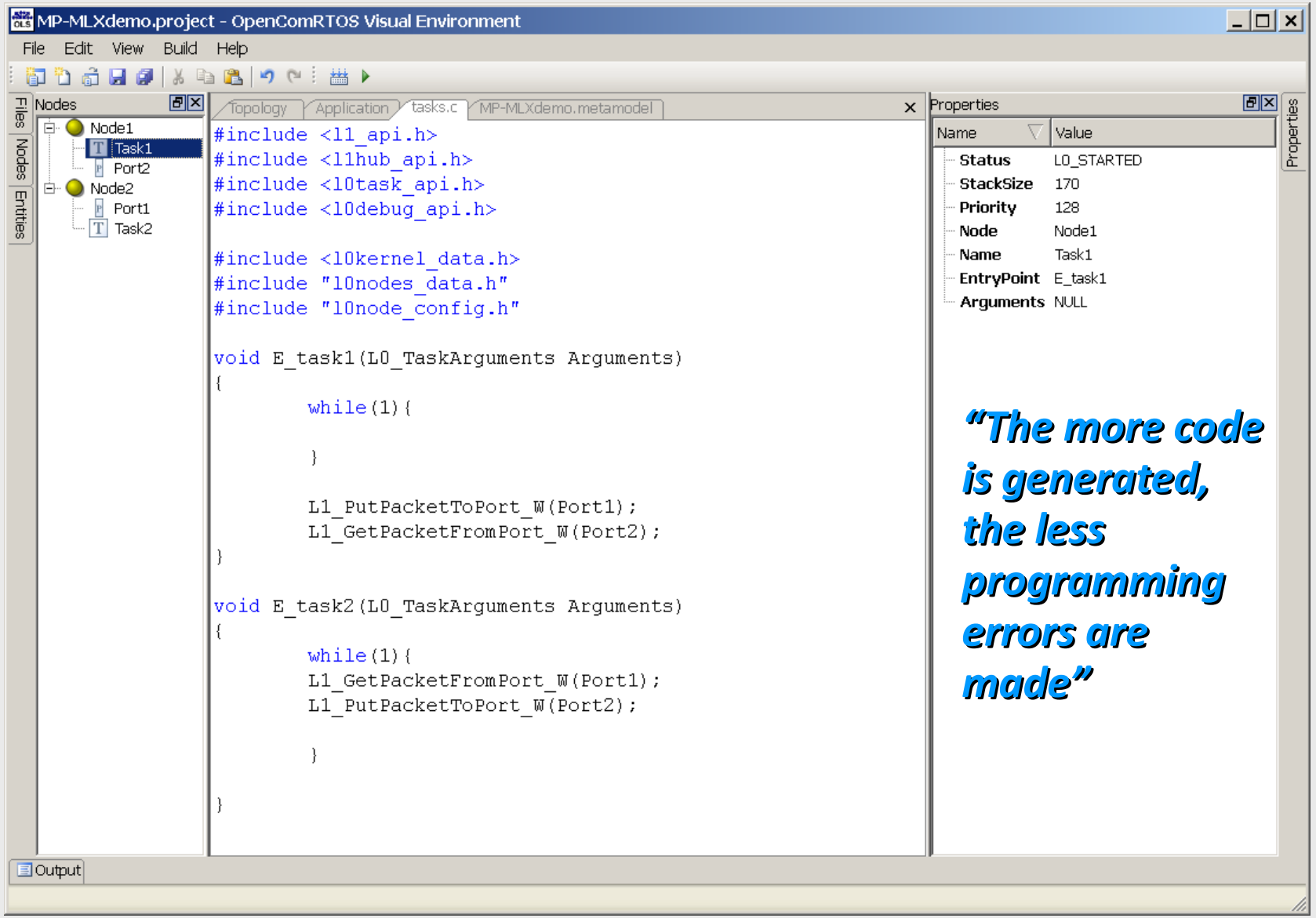
OpenVE: Defining the topology



OpenVE: Application model

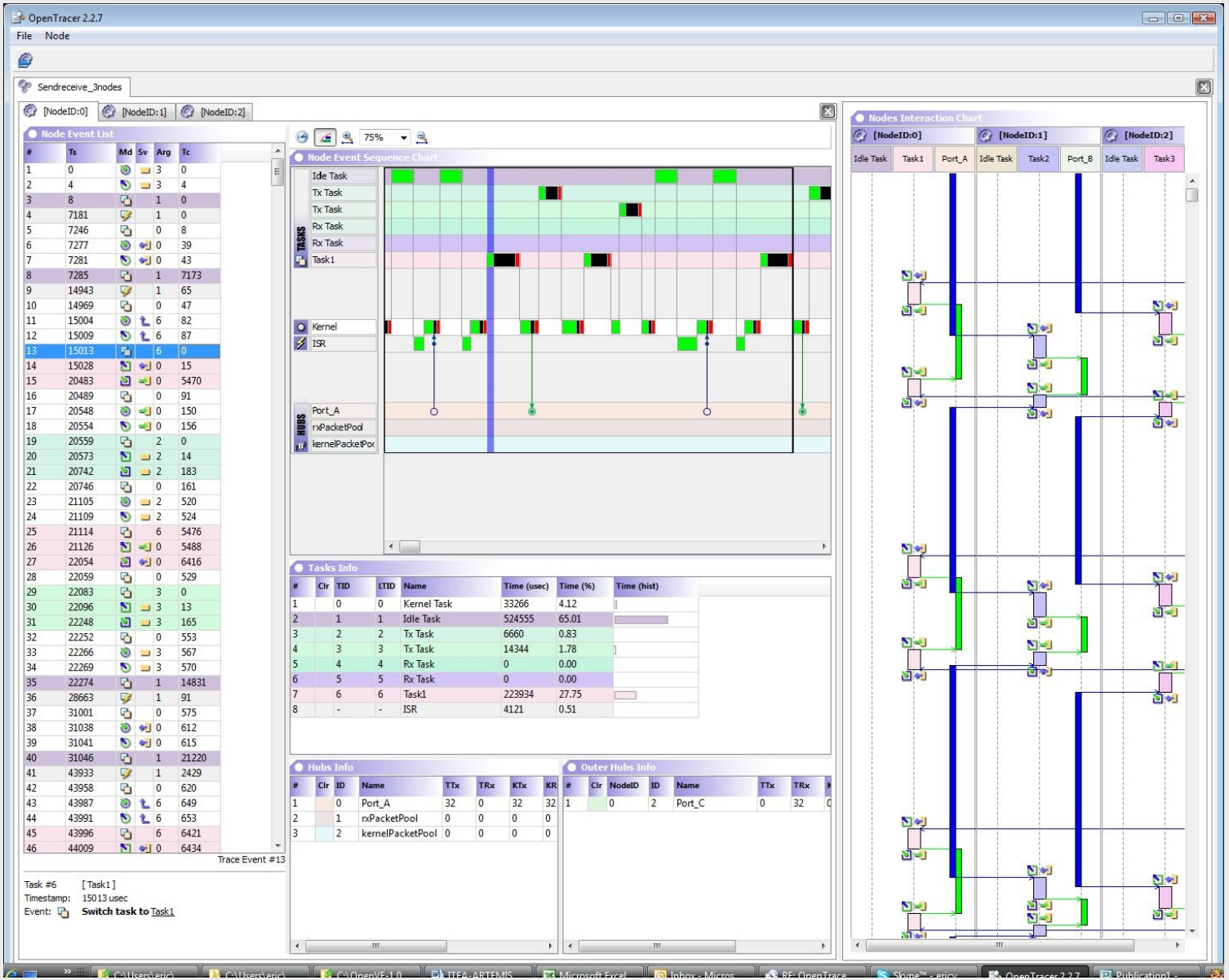


OpenVE: generate C for OpenComRTOS

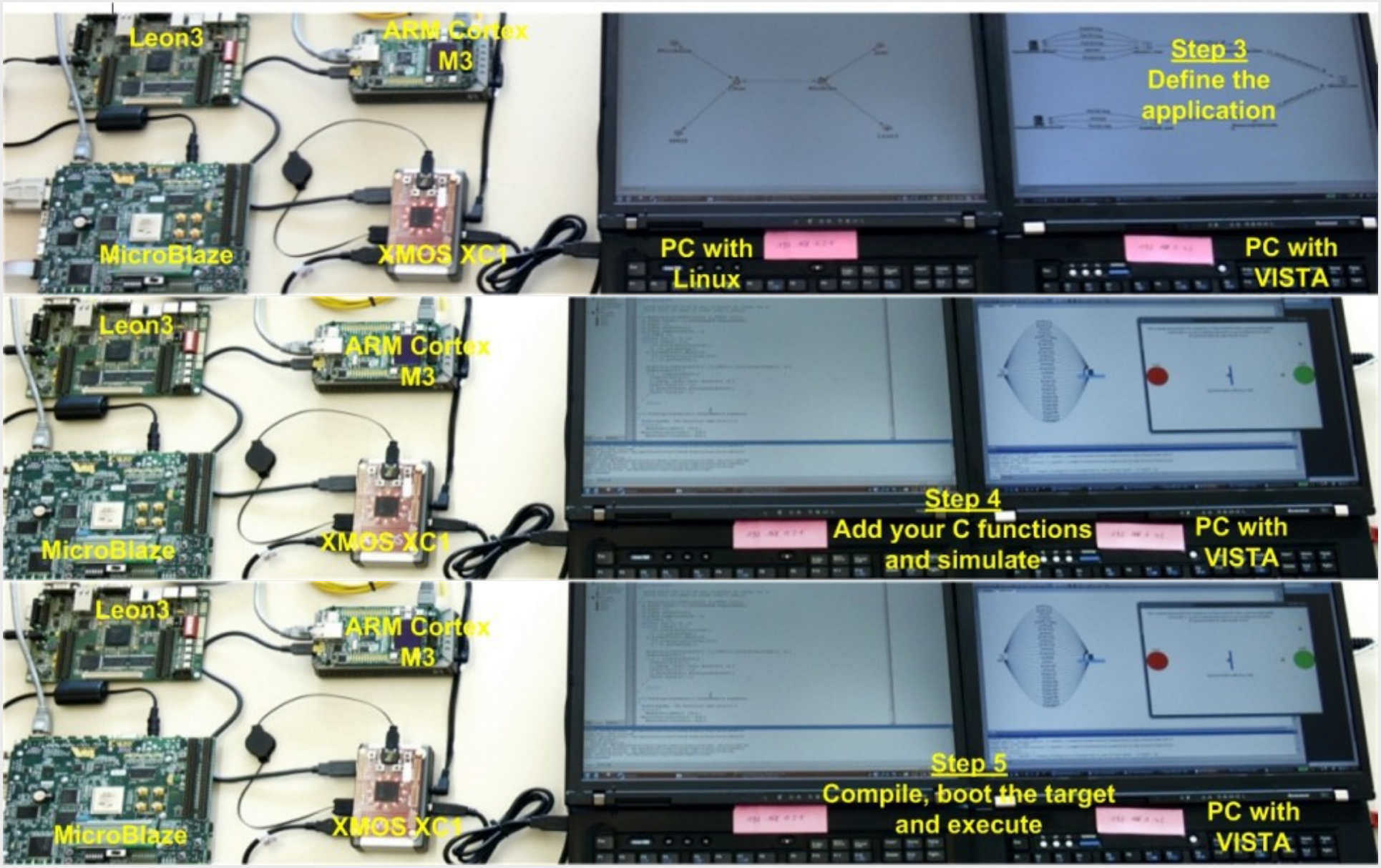


"The more code is generated, the less programming errors are made"

Tracer: Run and verify



Demo set-up


Leon3
ARM Cortex M3
MicroBlaze
XMOS XC1
PC with Linux
Step 3
Define the application
PC with VISTA
Leon3
ARM Cortex M3
MicroBlaze
XMOS XC1
Step 4
Add your C functions and simulate
PC with VISTA
Leon3
ARM Cortex M3
MicroBlaze
XMOS XC1
Step 5
Compile, boot the target and execute
PC with VISTA

Result:

“Transparent and processor independent!”

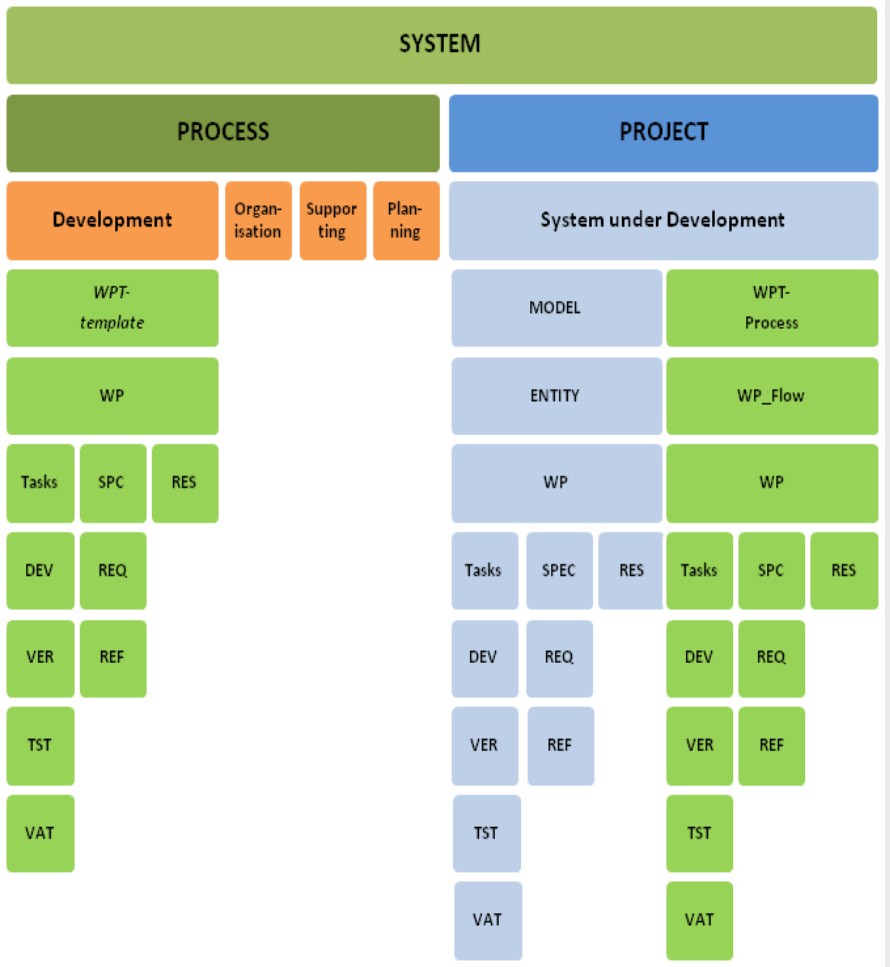
- OpenComRTOS supports heterogeneous networked and many-core processor systems:
  - Remapping tasks or RTOS entities requires no source code changes
- Timings will differ but logic application remains
- Meta-models hide complexity for user

Content

- **Who I am?**
- **Who's Altreonic?**
  - Formalised Systems and Software Engineering
- **The OpenComRTOS approach**
  - Formally modeled
  - Hubs and packet switching
  - Small code size
  - Virtual Single Processor model
  - Scalability, portability, …
  - Visual Programming
- **The GoedelWorks**
- **Project Proposals**
  - Event B Support
  - Alloy support
  - Frama-C integration

“GoedelWorks”

- **An internet based portal for safety and systems engineering**
- **It help teams to:**
  - Collaborate
  - Reduce risks.
  - Automate
  - Analyze
  - Leverage their knowledge
- **Simple and clean metamodel that allows users to:**
  - Import any process: from company-specific processes to standard-imposed.
  - Formalise their development
  - Comply to the steps imposed by the safety standards
- **Full traceability from early requirements to work products**
- **It is agile: Bottom-up and top-down approaches are allowed!**



Simplified of the GoedelWorks Metamodel

Content

- **Who I am?**
- **Who's Altreonic?**
  - Formalised Systems and Software Engineering
- **The OpenComRTOS approach**
  - Formally modeled
  - Hubs and packet switching
  - Small code size
  - Virtual Single Processor model
  - Scalability, portability, …
  - Visual Programming
- **The GoedelWorks**
- **Project Proposals**
  - Event B Support
  - Alloy support
  - Frama-C integration

Project Proposals

- **Key points:**
  - "Formal or specification focused front-end for RTOS applications"
  - OpenComRTOS Designer
    - develop fully trustworthy systems and applications
  - Applications are still developed using error-prone programming specification
  - oriented front-end tools that allows formal verification as well as code generation

Project 1: “Event B Support”

- **Description:**
  - Specify the behaviour of the sequential parts of the application Tasks
    - Event-B and Event-B model checkers like Pro-B and Atelier B
  - C-source-code generation from the models

- **Tasks:**
  - Studying Event-B and the supporting tools
  - Study the C-code generation and its constraints
  - Implement a proof of concept implementation

Project 2: “Alloy support”

- **Description:**
  - Specify the behaviour of the sequential parts of the application Tasks
  - C-source-code generation from the models

- **Tasks:**
  - Studying Alloy and the supporting tools
  - Study the C-code generation and its constraints
  - Implement a proof of concept implementation

Project 3: “Frama-C integration”

- **Description:**
  - increase the assurance in the application and kernel code written in C
  - Frama-C is used to verify and proof the C-code of the applications that are created using the OpenComRTOS design tools

- **Tasks:**
  - Studying Frama-C and its applicability
  - Developing use-case scenarios as a function of Frama-C’s capabilities
  - Integrating with the OpenComRTOS design process
  - Developing proof of concept demonstrations

Thank you for your attention

"If it doesn't work, it must be art.
If it does, it was real engineering"

For more information:

- Leonel Braga: leonel.braga@altreonic.com
- Eric Verhulst: eric.verhulst@altreonic.com

www.altreonic.com