Concurrent systems composing in a reliable and efficient way

Eric Verhulst

www.altreonic.com

Push Button High Reliability

HALL 6 - 615

Altreonic's mission

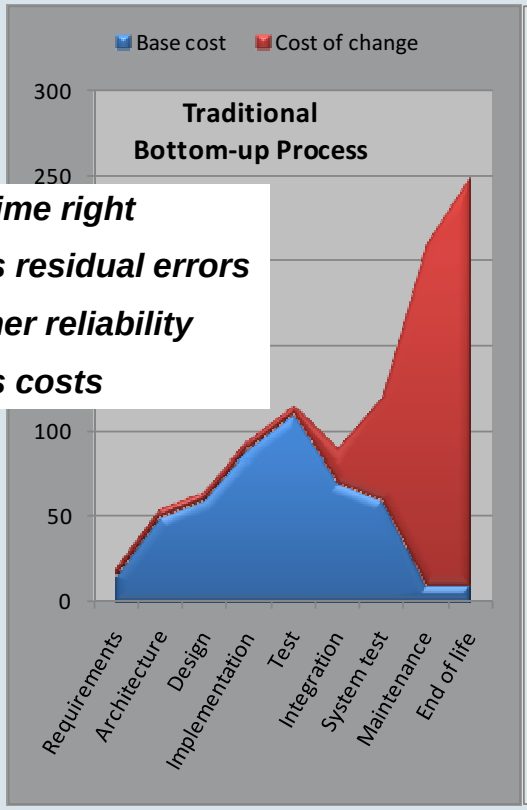
- **"To provide a unified, yet streamlined methodology with supporting tools and products to make high reliability and scalable performance cost-efficient"**
- Focus is on **high reliability embedded** markets
- More performance and trust utilising less resources
- Application domains:
  - Ultra low power embedded devices
  - Distributed sensing and control
  - Many/multicore devices
  - Parallel supercomputing
  - Fault tolerant/ safety critical systems

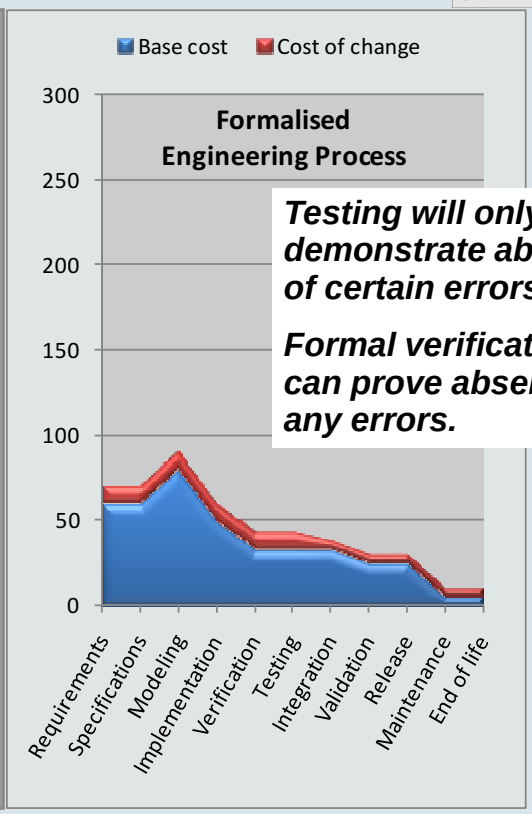
How is this possible?

- **Formalisation**
  - Helps to deeply understand the problem domain
  - Helps to find better, leaner and cleaner solutions
  - Helps to find better architectures
  - Helps to improve reuse
  - Helps to get it right the first time

- Our methods:
  - **Unified semantics**
    - Speak the same language from early requirements capturing till final product / system is put to use
  - **Interacting Entities**
    - A common, yet very scalable and modular architectural model

Why FORMALISATION works





First time right

= Less residual errors

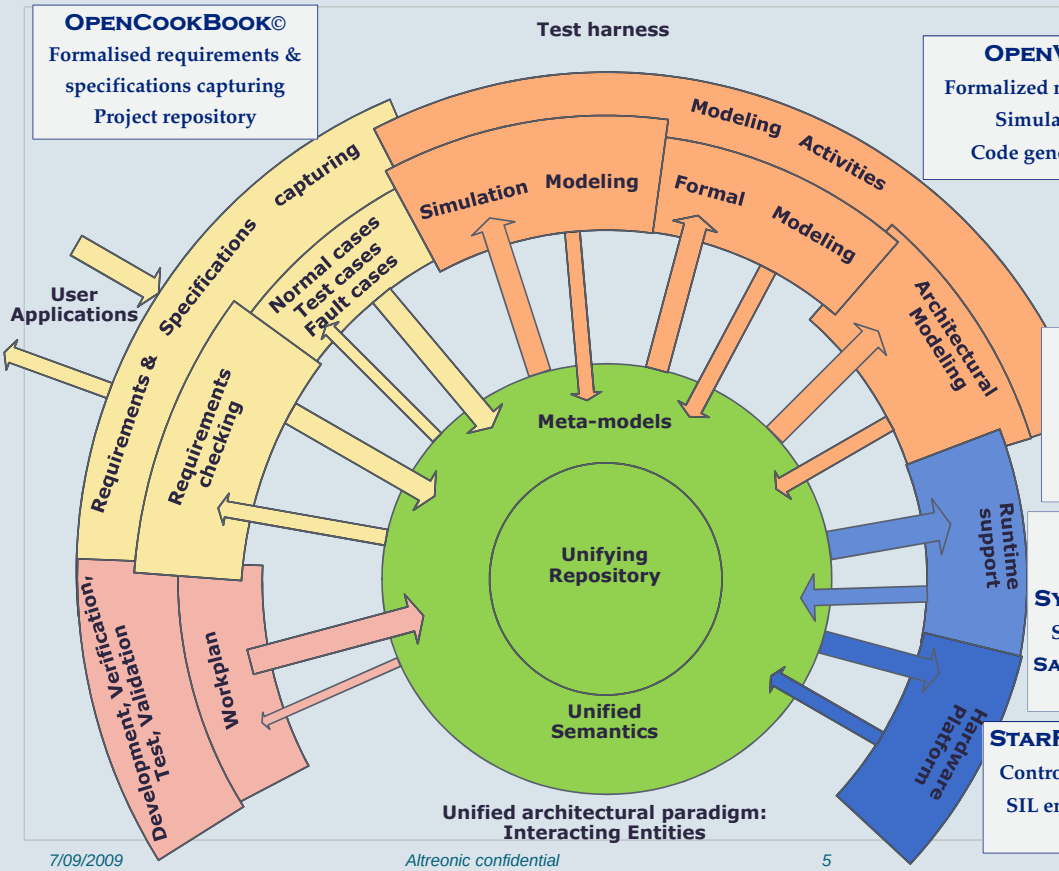
= Higher reliability

= Less costs

Testing will only demonstrate absence of certain errors.

Formal verification can prove absence of any errors.

Concurrent Systems Composer



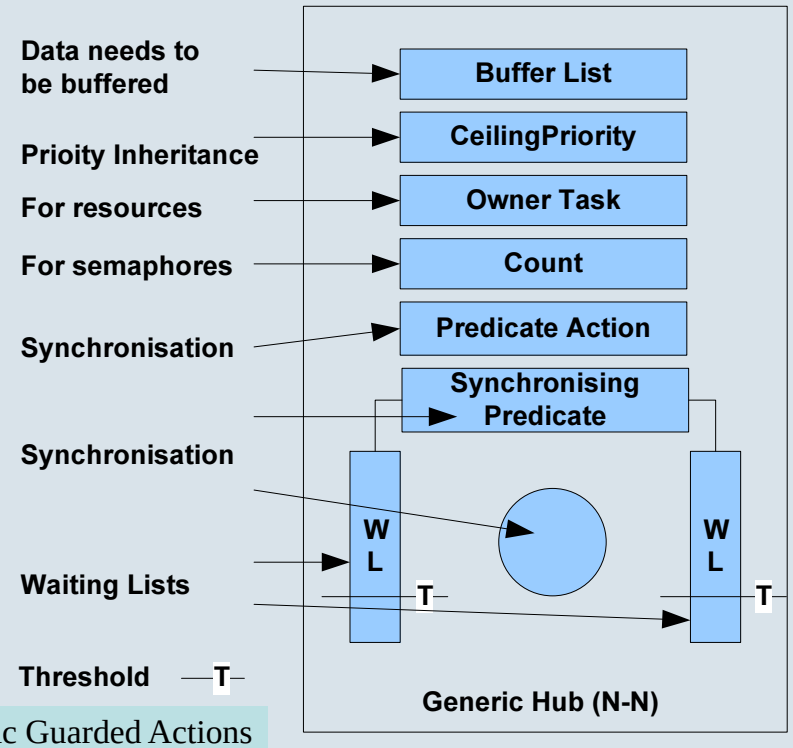
Unique software technology

- Formalised but straightforward approach
- Full integration of tools from requirements to final applications is unique
- OpenComRTOS is a **unique** programming system, a unique network-centric RTOS, quasi-universal
  - Formally developed and verified
  - Scalable yet very small: typically 2 to 5 kiB/node
  - Real-time communication support
  - Heterogeneous target support
  - OpenComRTOS **nominated embedded award**
- Capable of fault-tolerance
  - (at affordable cost)

The OpencomRTOS “HUB”

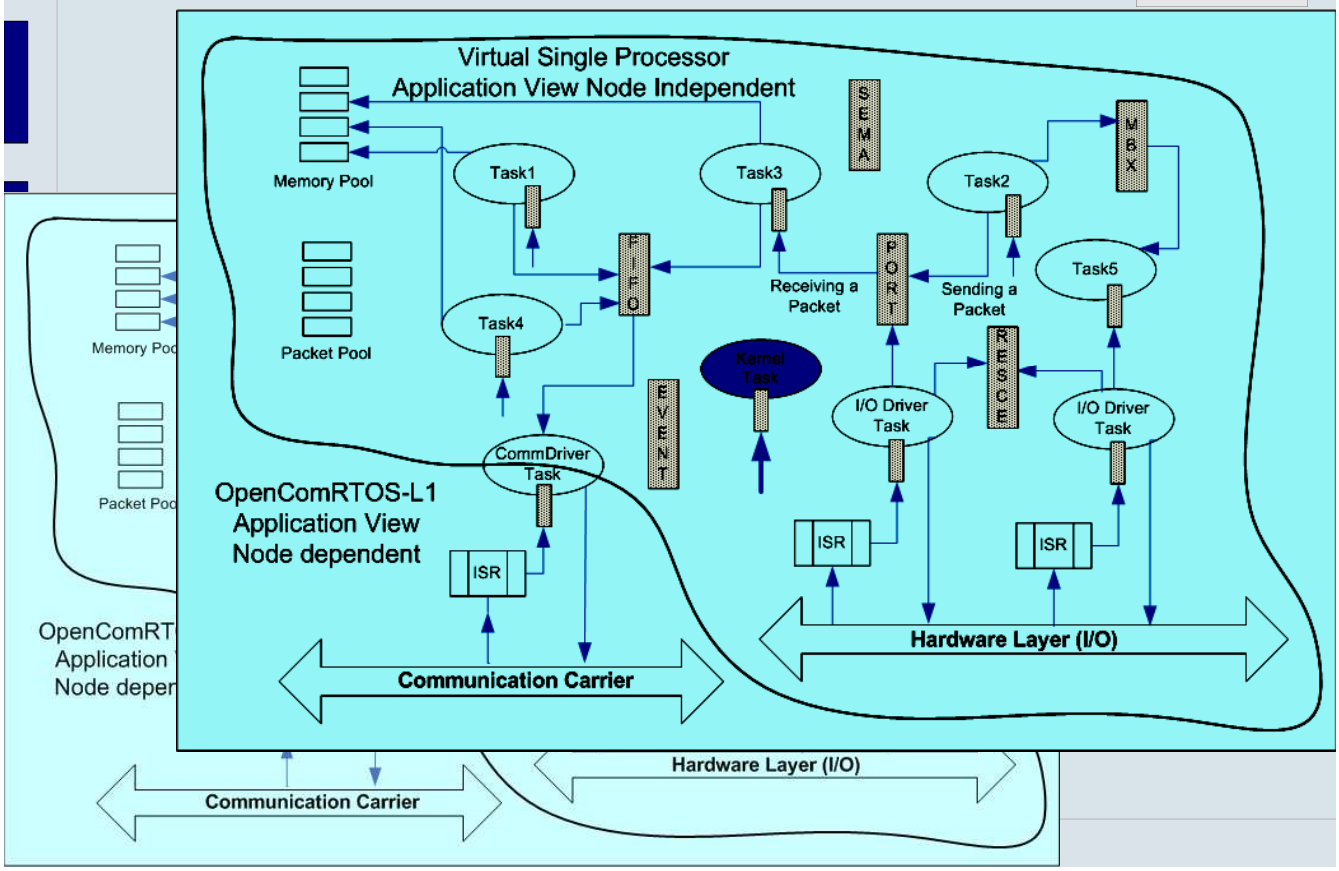
- Result of formal modeling (TLA+)
- Events, semaphores, FIFOs, Ports, resources, mailbox, memory pools, etc. are all variants of a generic HUB
- A HUB has 4 functional parts:
  - Synchronisation point between Tasks
  - Stores task’s waiting state if needed
  - Predicate function: defines synchronisation conditions and lifts waiting state of tasks
  - 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!
  - => a new concurrent programming model

The generic hub as metamodel



Similar to Atomic Guarded Actions
Or
A pragmatic superset of CSP

Resulting programming model



Codesize Figures

- Up to 10x smaller than traditional design (thanks to formal development)
- Less power, less memory, easier to verify, scalable ...

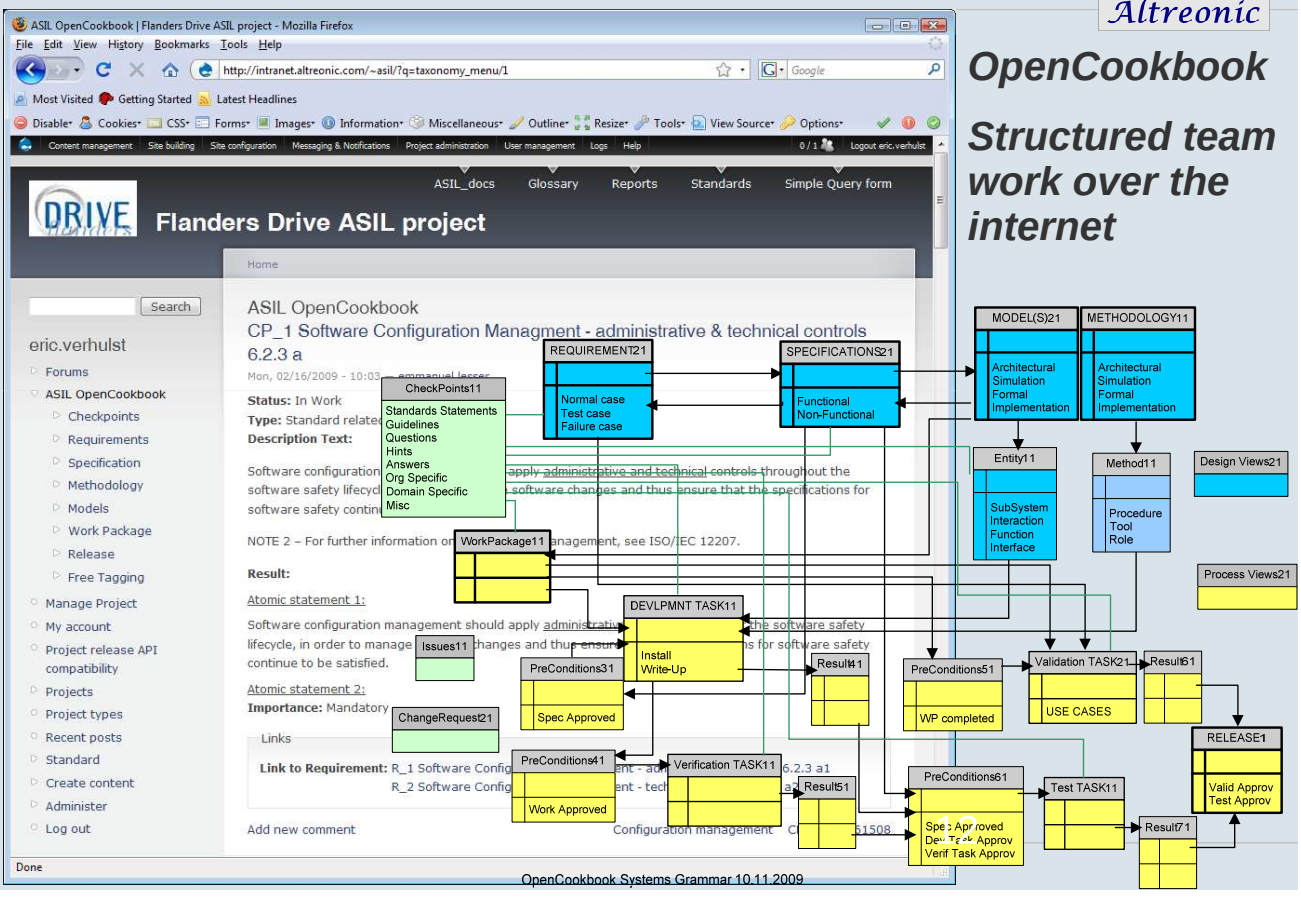
Service	MLX-16	MicroBlaze	Leon3	ARM	XMOS
L1 Hub shared	400	4756	4904	2192	4854
L1 Port	4	8	8	4	4
L1 Event	70	88	72	36	54
L1 Semaphore	54	92	96	40	64
L1 Resource	104	96	76	40	50
L1 FIFO	232	356	332	140	222
L1 PacketPool	NA	296	268	120	166
Total L1 Services	1048	5692	5756	2572	5414

Code size figures (in Bytes) obtained for our different ports, compiled with Optimisation Os

Applications potential

- Ultra low power:
  - SoC, 2K instructions on CoolFlux DSP of NXP
  - E.g. hearing aids
- Sensor and actuator networks
  - Small code size
  - Power saving modes, wake up by interrupt
  - System wide routing
- Distributed control
  - Network support is built in
  - Easy to integrate redundancy
  - Easy to distribute control and I/O
  - No more binding glue, no more middleware layers
- Parallel “supercomputing”
  - Parallel heterogeneous DSP networks
  - Intel 48 core SCC chip

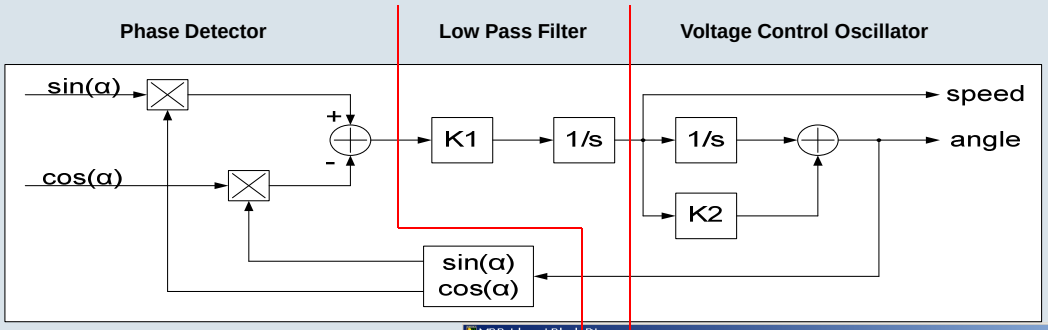
Step1: Requirements & Specifications



OpenCookbook

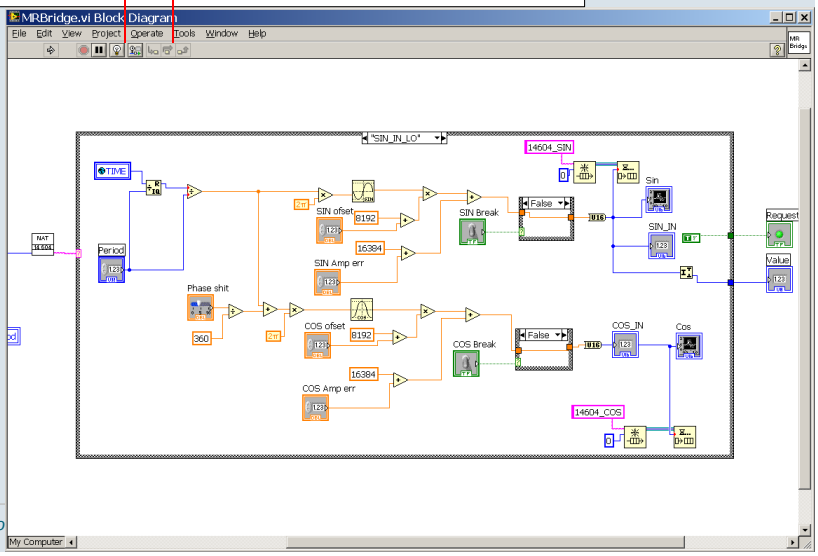
Structured team work over the internet

Step2a: simulation and formal models

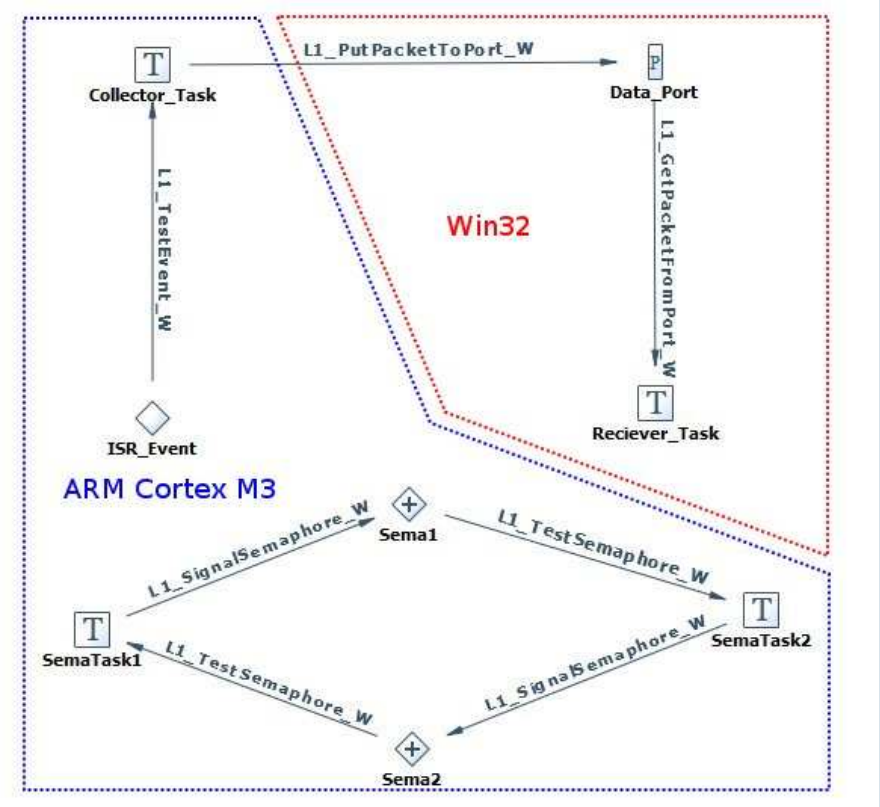


(third party tools)

Simulating the algorithm in a PC doesn't cost much, but allows to find the issues early on



Step2b: Implementation Modeling



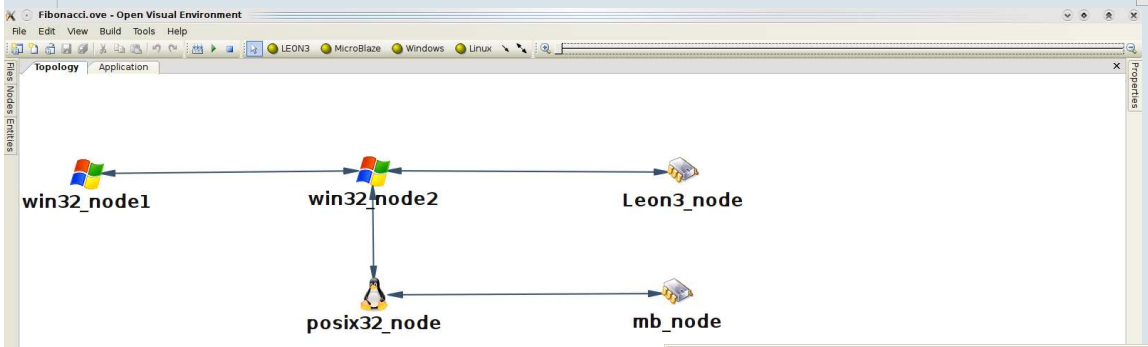
After simulation and model checking, select the application architecture and start development

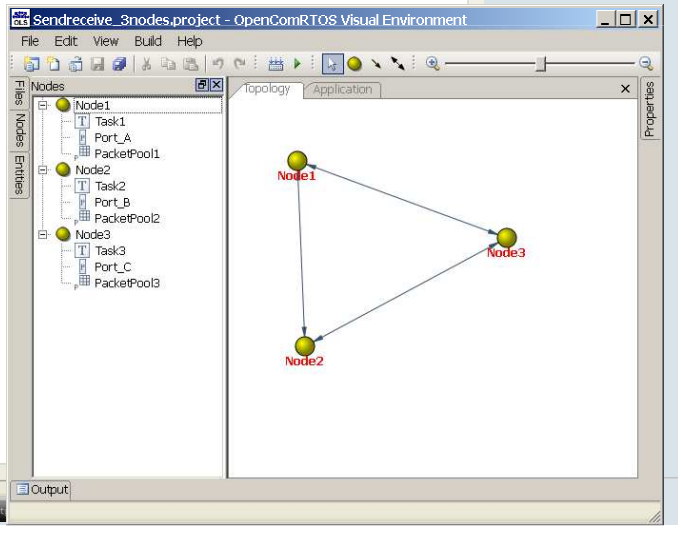
Step3a: Select processing modules

- ***Networked control modules do the real work.***
- ***Added value from high reliability and high performance algorithms***
- ***Fault tolerance is a configuration option***

Altreonic Inside

Step3b: target topology

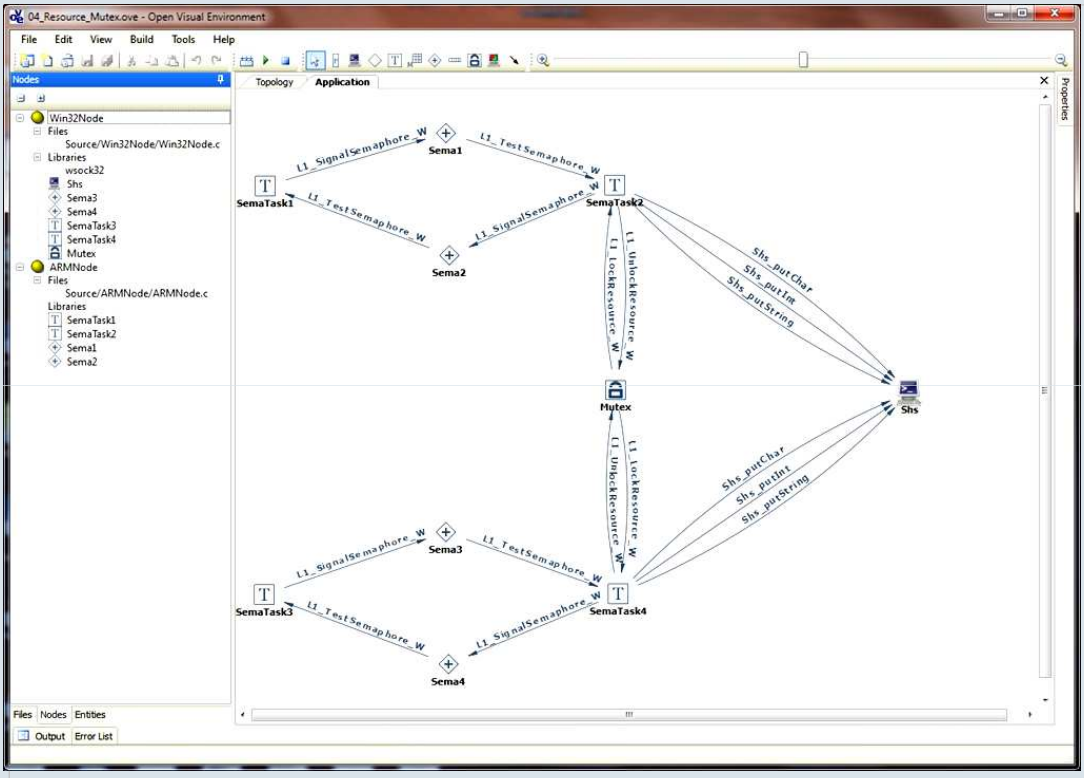




OpenVE:

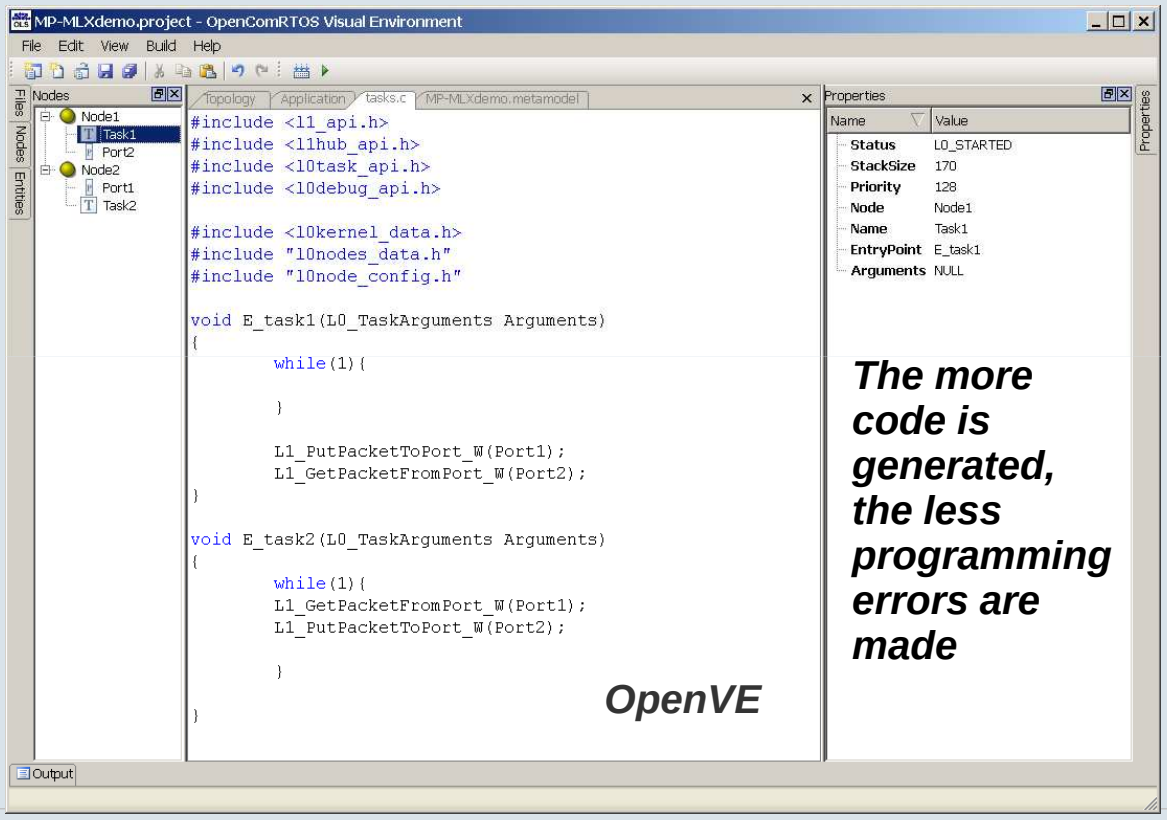
How are processors connected ?

Step 4a. Application model



OpenVE ***How is the application structured ?***

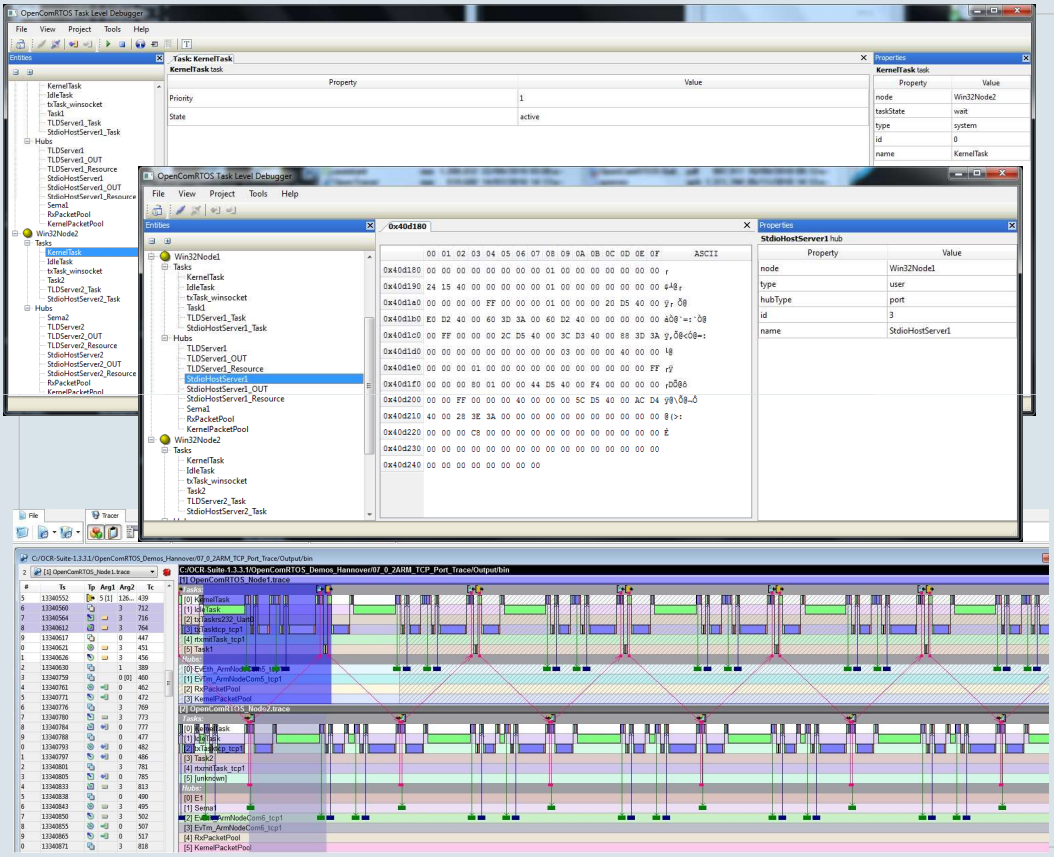
Step4b: generate C for OpenComRTOS



OpenVE

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

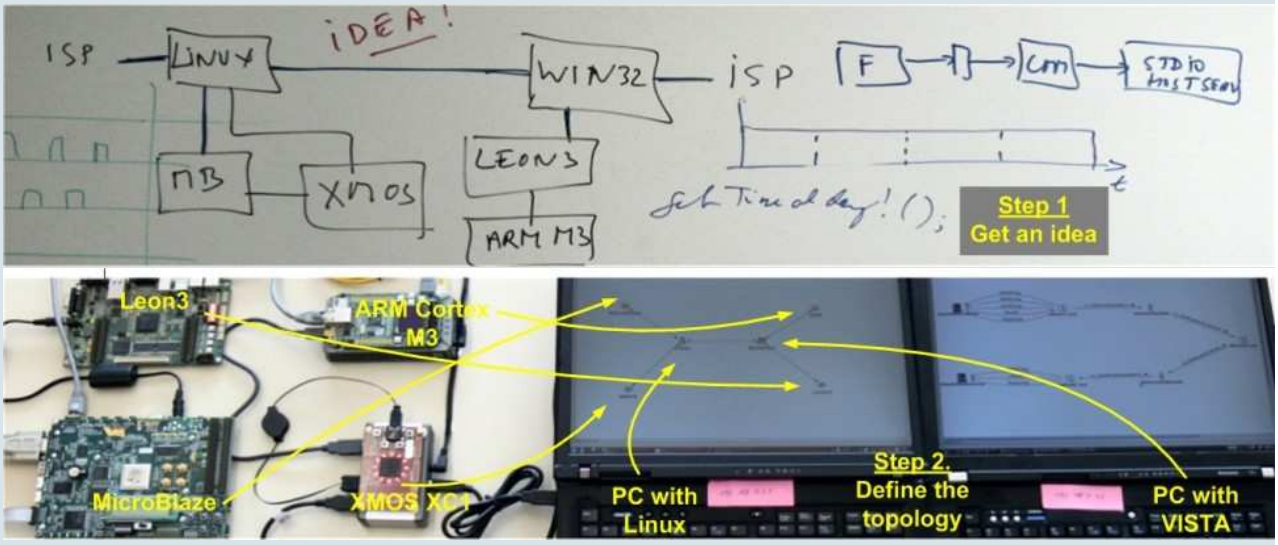
Step5: Run and verify



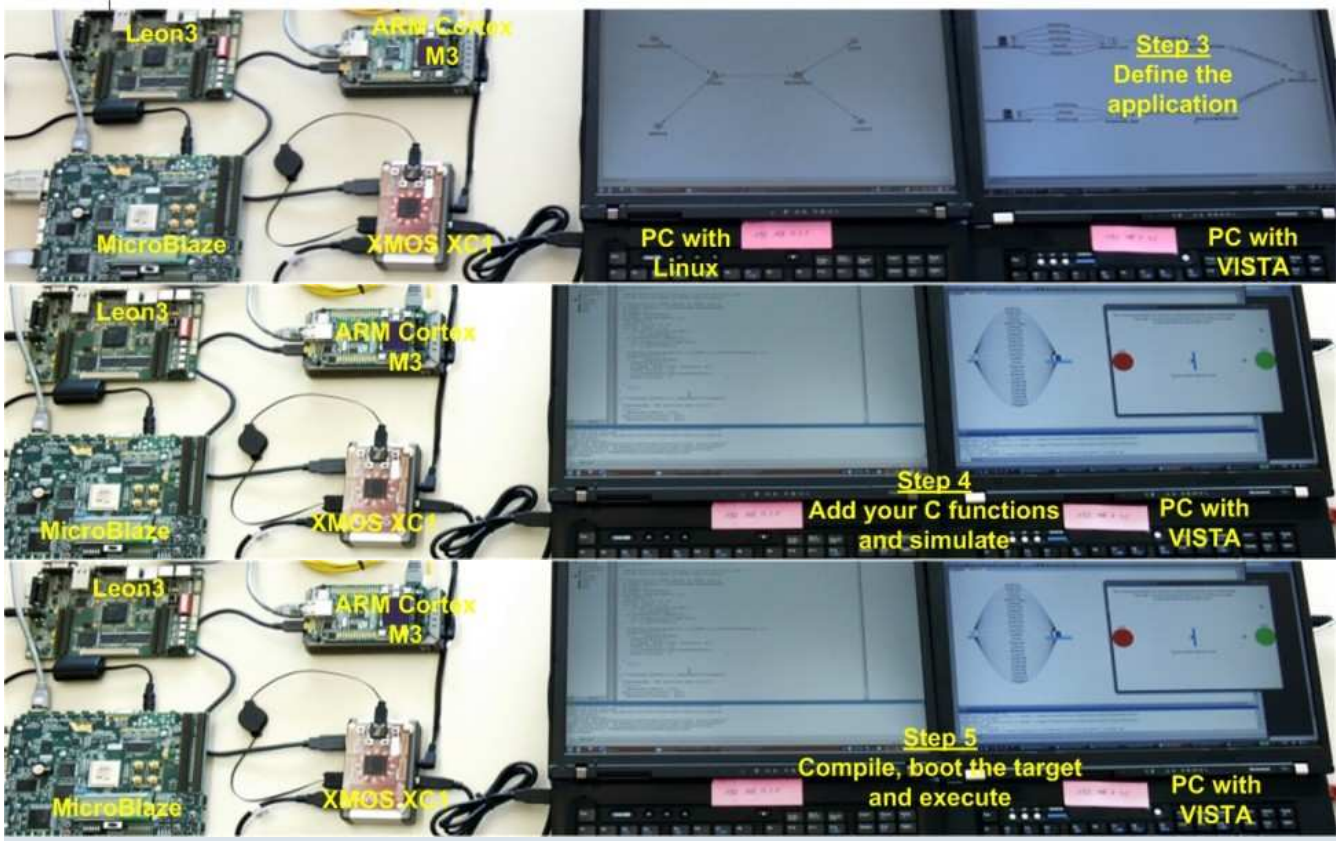
Verification and testing is needed to confirm the work was well done

What happened?

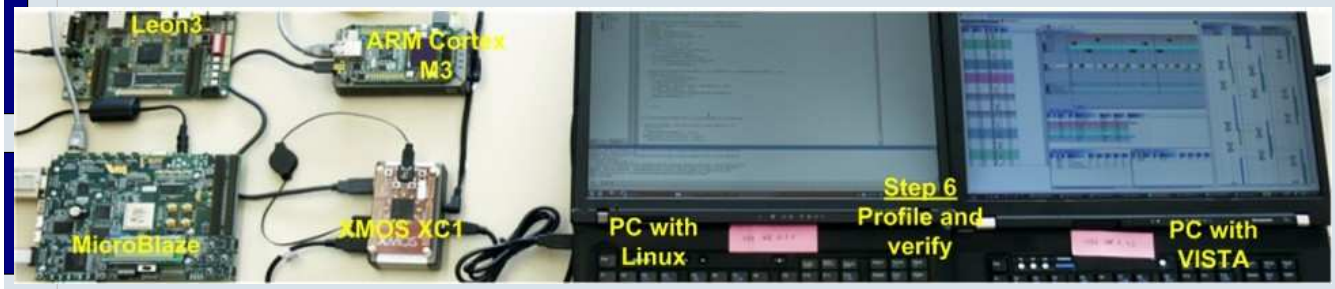
From idea to prototype in a seamlessly integrated and controlled process



Demo set-up



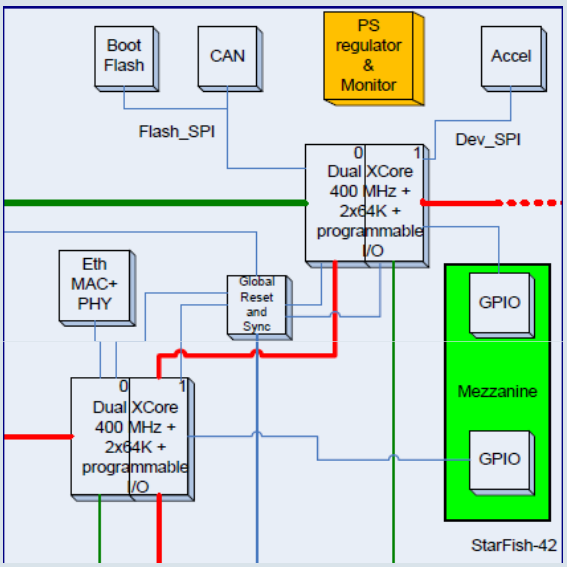
Transparent and processor independent!

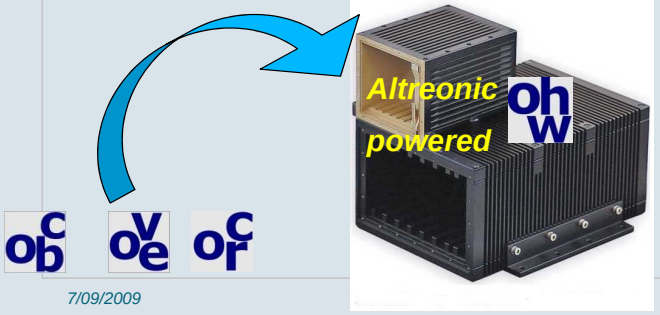


- OpenComRTOS supports heterogeneous networked and many-core processor systems:
  - Remapping tasks or RTOS entities requires no source code changes
- Timings will differ but logic application remains
- Meta-models hide complexity for user

StarFish customizable controllers

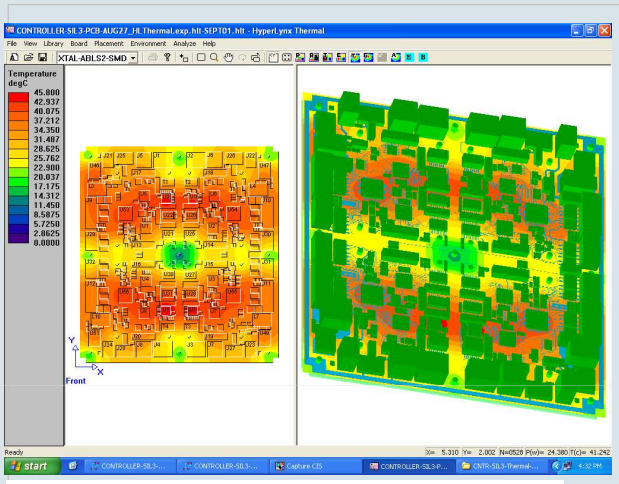
- **Key characteristics :**
  - Scalable performance
  - High Reliability (SIL3)
  - Fault Tolerance (SIL4)
- **Target market :**
  - Robotics, Automotive,
  - Transport, Aerospace,
  - Machine Control.

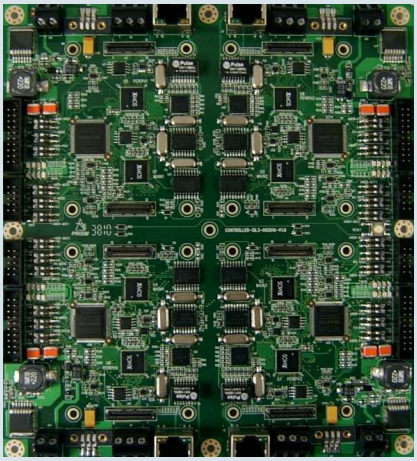


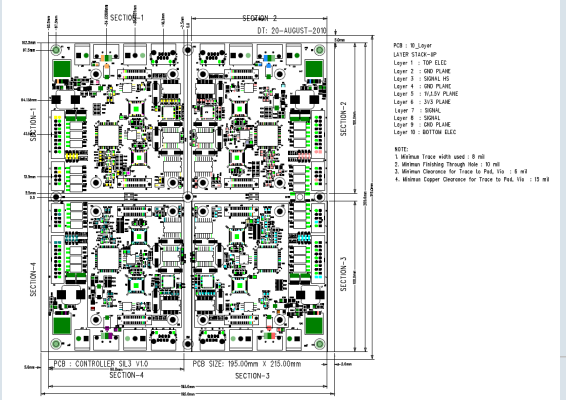


(Status: engineering systems Q4)

StarFish engineering system

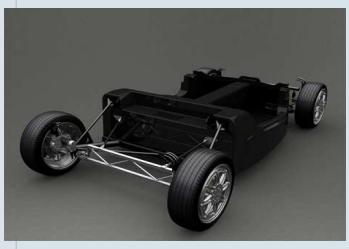






- Allows full access
- Fully closed enclosure (IP64 or higher)
- Power consumption rated at 7.5 W when using all quadrants @ > 3200 Mips
- Application specific mezzanines
- Production version will be compact and stacked or use one quadrant as unit

Use case 1 : Drive-by-wire e-wheel

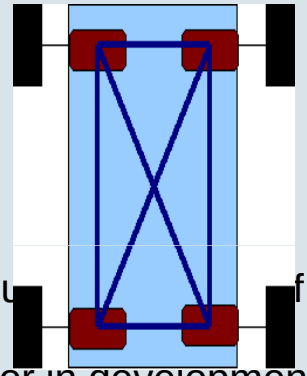


Key characteristics :

➢High Reliability (SIL3) → Fault Tolerance (SIL4)

All-in:

- Traction
- Braking
- Anti-slip
- Stability control
- Active suspension



➢ Exploits transparent distribu f **OpenComRTOS**

➢ Own controllers and e-motor in development

➢ Software and Hardware redundancy enables **fault-tolerant controllers** in 1-, 2-, 3-, 4-, n-wheel platforms

=> StarFish was designed with such topology in mind

Binary, Source and Open Licenses

- Innovative no-risk **open licensing scheme** as well as binary and source code licenses
- Binary
  - Single seat/single site
  - No runtime royalties
- Source code
  - kernel and code gens
- Open Technology license
  - Formal models, design doc, source code, test suites, … of RTOS + GUI tools
  - Right to generate extra binary licenses
    - Small royalty
  - For all Software and all Hardware products

Conclusion

- Altreonic's value is based on a long experience and track record

- We have several **unique** products
- Focus is **High Reliability with less resources**
- **High added value** for customers
- Open Licensing scheme = no risk

www.altreonic.com

HALL 6 - 615