

**“FM 2008”**

***An industrial case:  
pitfalls and benefits of applying formal  
methods to the development of a  
network-centric RTOS***

**Eric.Verhulst @ OpenLicenseSociety.org**

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

1



## Who is Open License Society?

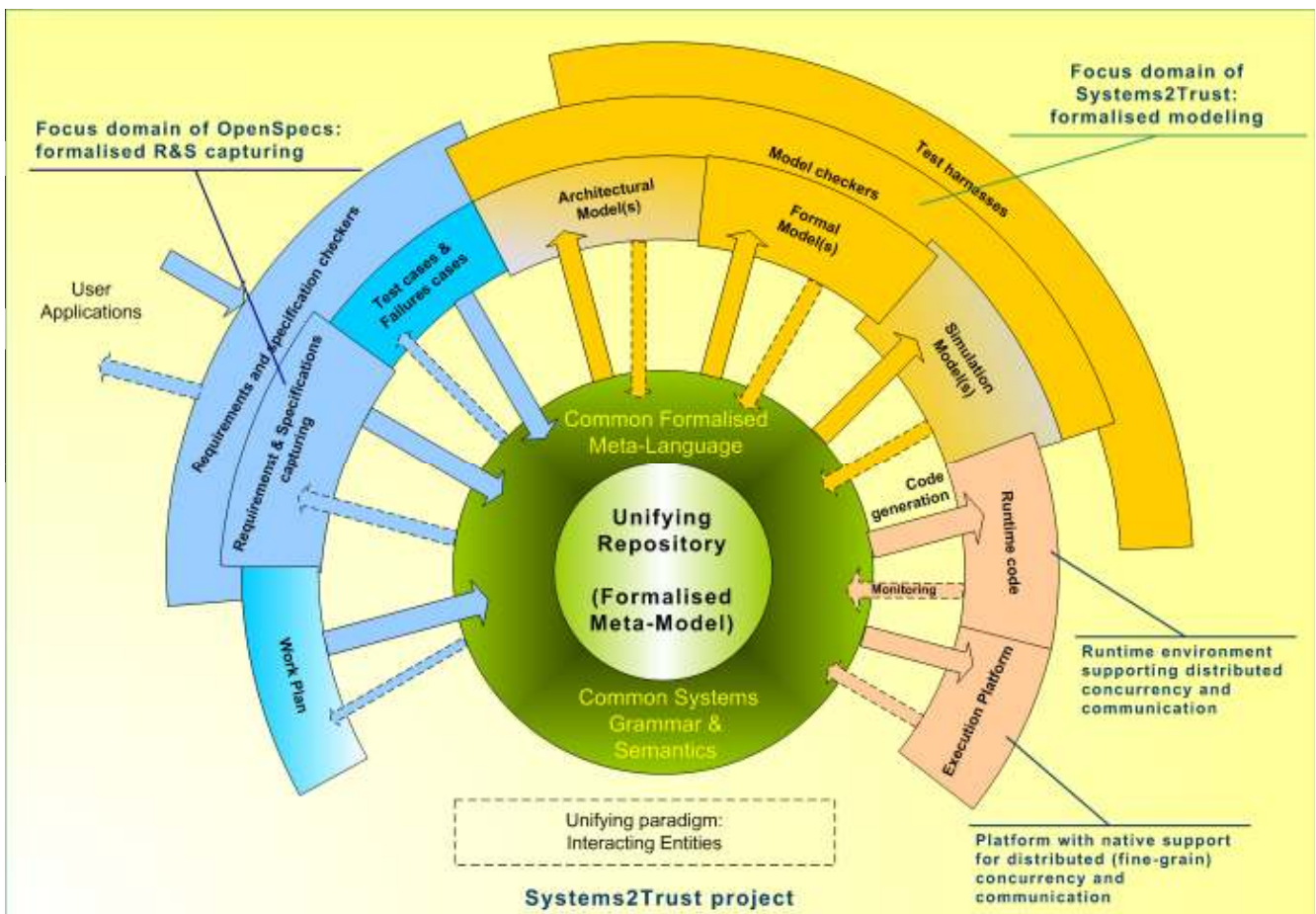
- Beyond “Open Source”
- Privately funded R&D institute
  - Leuven (BE), Berdyansk (UA)
  - Industrial sponsors
  - IWT project funding for OpenComRTOS
  - Results now commercialised via Altreonic Co.
- Why: 70 % of all SE projects do not deliver
- Objectives
  - Systematic & Unified Systems Engineering Methodology
  - ‘Interacting Entities’ paradigm at all levels:
    - OpenComRTOS as runtime environment (formally developed)
  - Implies ‘Trustworthy Components’
    - => Open License (source code + all design, test, .... docs)
- Focus:
  - Embedded Systems:
    - Constraints driven development
    - Real-time, distributed, hardware & software, ...

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

2





26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

3



Can we trust our mind ?  
That's why we need formal(isation)

- Can you find the 3 letters « F » ?

**FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS**

Did you see the similarity with source code (debugging) ?

26/05/2008

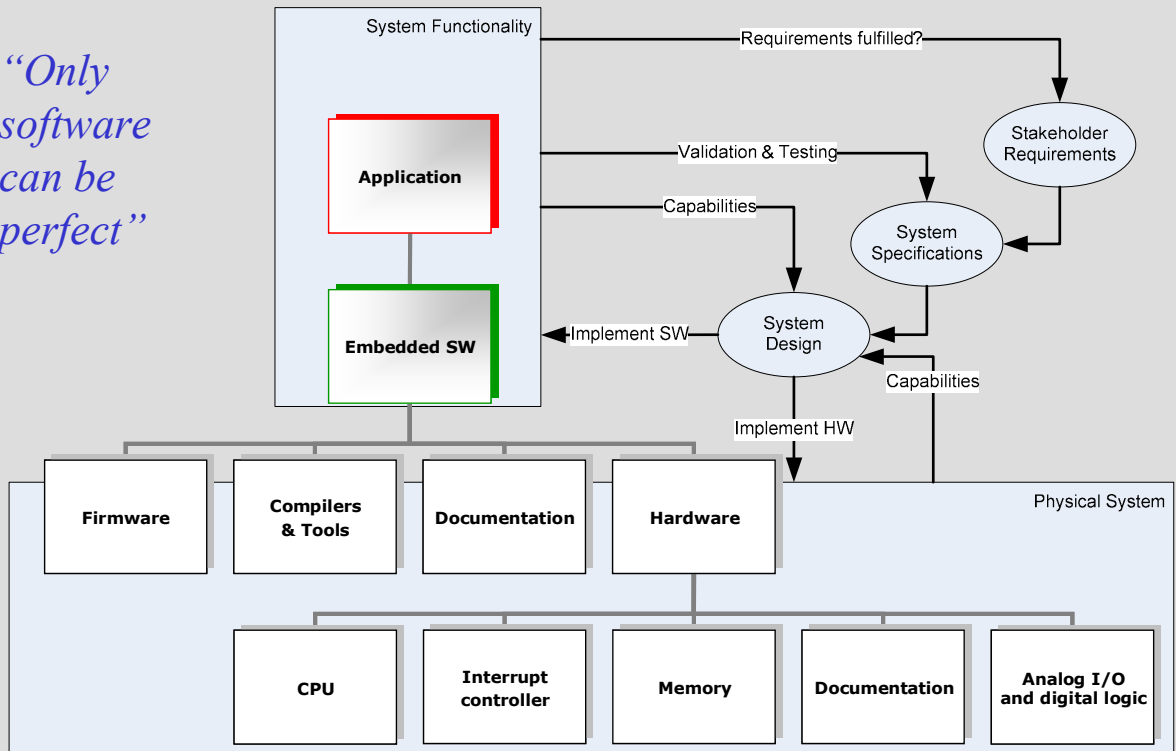
[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

4



# Systems/Software Engineering Process dependency graph

*“Only software can be perfect”*



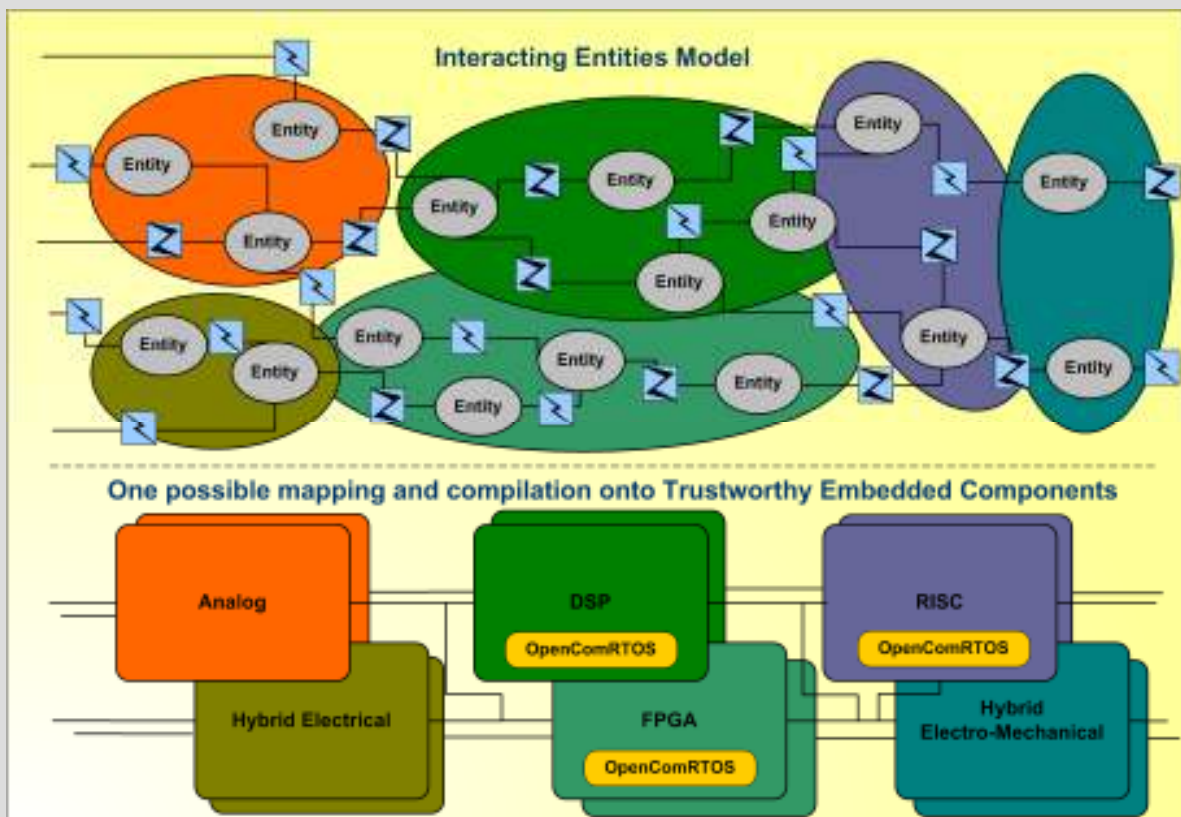
26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

5



## Unifying architectural paradigm: Interacting Entities



26/05/2008

[www.openlicense.org](http://www.openlicense.org)

6



## OpenComRTOS project objectives

- **Funded R&D project (IWT, Flanders)**
  - Lancelot Research: management, commercialisation
  - Open License Society: technology development
  - University Gent (INTEC, Prof. Boute): formal modeling
  - University Berdyansk: tools and formal validation
  - Melexis: co-sponsor and first user (16bit uC)
- **GUI tools:**
  - graphical modeling/development environment
- **Goal:**
  - Develop Trustworthy distributed RTOS
    - Follow OLS SE methodology
    - Formal verification & analysis: formal modelling
  - Scalable distributed RTOS
  - Verify benefits and issues of using Formal Modeling

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

7



## Some requirements

- **Targets:**
  - Single chip, tightly coupled: multi-core
  - Multi-chip, tightly coupled: parallel processors on board
  - Multi-boards, multi-rack: using backplane interconnects
  - Distributed: using LAN and WAN
  - Host node
- **Programming models:**
  - "Interacting Entities"
  - "Virtual Single Processor":
    - transparent for topology
    - Supporting heterogenous targets
    - Write once, run everywhere
  - Distributed real-time
  - Safe, secure
  - Small code size, low latency (=high performance)

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

8



## RTOS Metamodel

- Based on Interacting Entities Paradigm
- Application can be constructed from various entities (*kernel entities*) and interactions between them (*kernel services*).
- The Metamodel allows extensions to different sets of kernel entities and services of other RTOSes.
- Expression of the Metamodel in XML format

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

9



## OpenComRTOS systems grammar

### OpenComRTOS **IS\_DEFINED\_BY**

Configuration (1) // The root node of XML file

Configuration **IS\_DEFINED\_BY** // Nodes of configuration section

Parameters (1) **AND** // *Attributes* of the configuration section of XML file

SystemTasks (4) **AND** // Kernel, Idle, Rx or Tx

ApplicationTasks (1-N) **AND**

Ports (1-N) **AND**

Nodes (1-N) **AND**

Links (1-N)

Configuration **HAS\_ATTRIBUTES** // Parameters

DataSize (1) **AND** // Packet data size (in bytes)

NodeIdSize (1) // Length of Node identifier (in bits)

SystemTask **CAN\_BE** // Type of system task

KernelTask **OR**

IdleTask **OR**

RxTask **OR**

TxTask

Etc.

26/05/2008

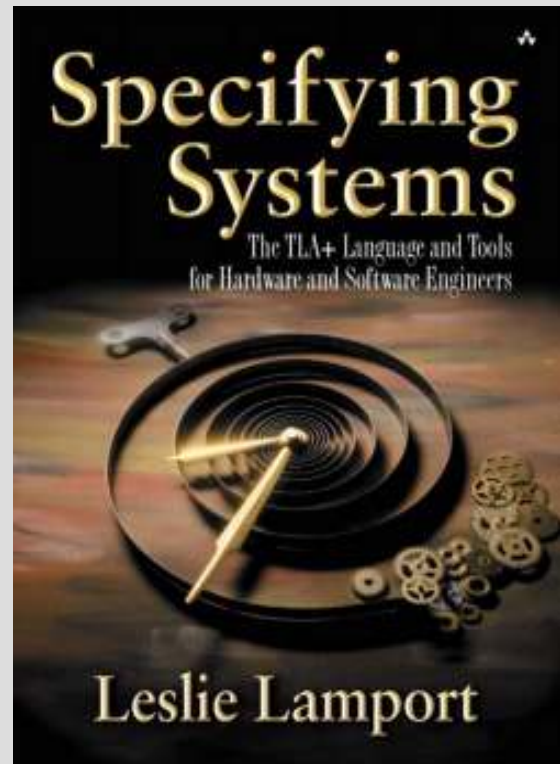
[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

10



# TLA

- **TLA (the Temporal Logic of Actions)** is a logic for specifying and reasoning about concurrent and reactive systems.



26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

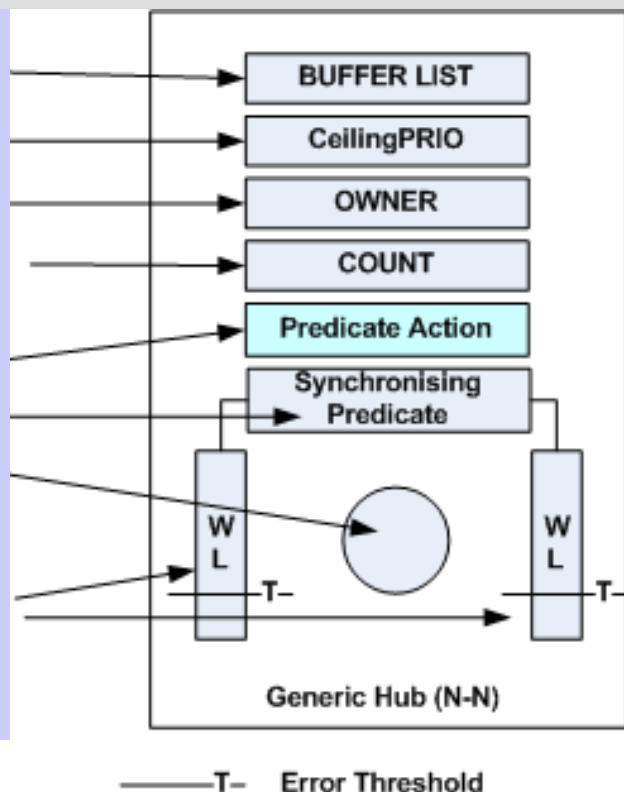
11



## TLA modelling results

We modeled entities of OpenComRTOS:

- Port
- Event
- Semaphore
- Resource
- Packet Pool
- Memory Pool
- FIFO
- Mailbox



26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

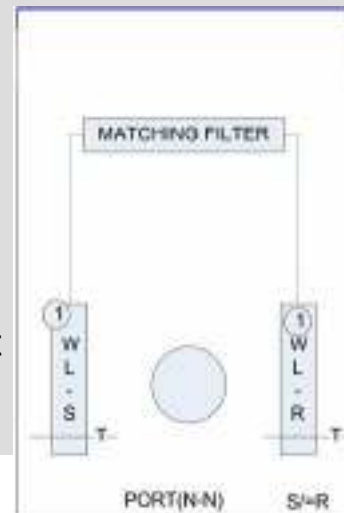
12



## Example: Port

### Verified Properties:

- There are never more Tasks on the ready list than there are Tasks in the system
- There are never more Tasks in the Port's waiting list than there are Tasks in the system
- All Tasks waiting on an Port, either waiting to send a Packet, either waiting to receive a Packet are of the same type in each waiting list



*TypeInvariant*  $\triangleq$

$\wedge \text{Cardinality}(\text{ReadyList}) \leq \text{Cardinality}(\text{TaskId})$

$\wedge \forall p \in \text{PortId} :$

$\text{Len}(\text{PortWL}[p]) \leq \text{Cardinality}(\text{TaskId})$

$\wedge \forall p \in \text{PortId} :$

$\forall i, j \in 1 \dots \text{Len}(\text{PortWL}[p]) :$

$\text{PreallocatedPacket}[\text{PortWL}[p][i]].\text{type} = \text{PreallocatedPacket}[\text{PortWL}[p][j]].\text{type}$

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

13



## The OpencomRTOS “HUB”

- Result of formal modeling
- 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

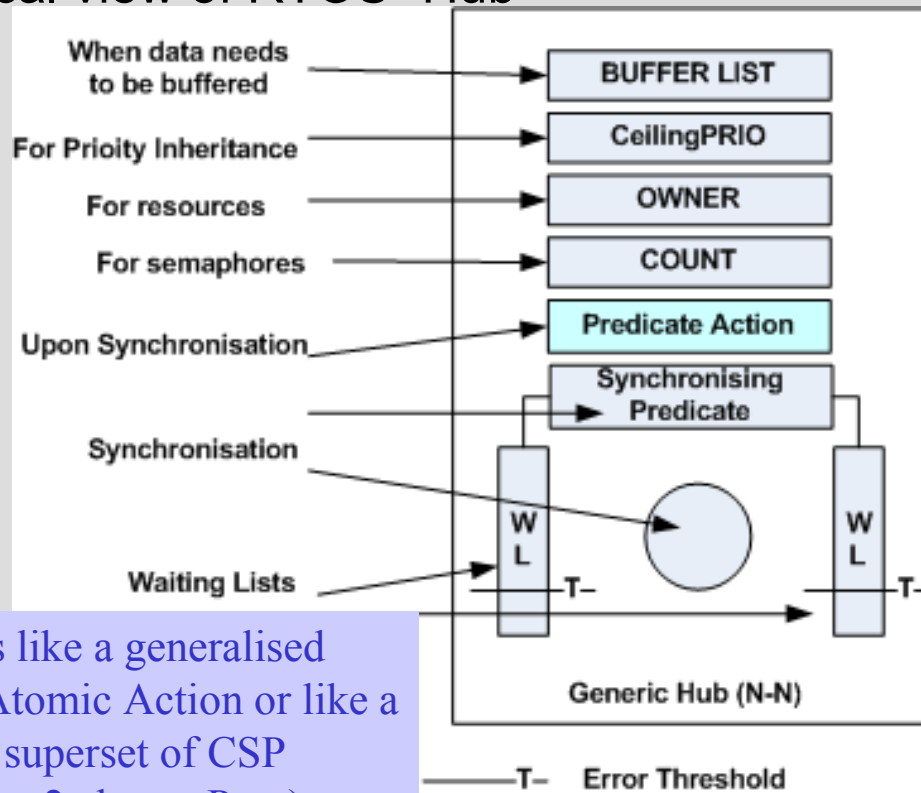
26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

14



# Graphical view of RTOS "Hub"



A "hub" is like a generalised Guarded Atomic Action or like a pragmatic superset of CSP channels (= 2 chan + Proc)

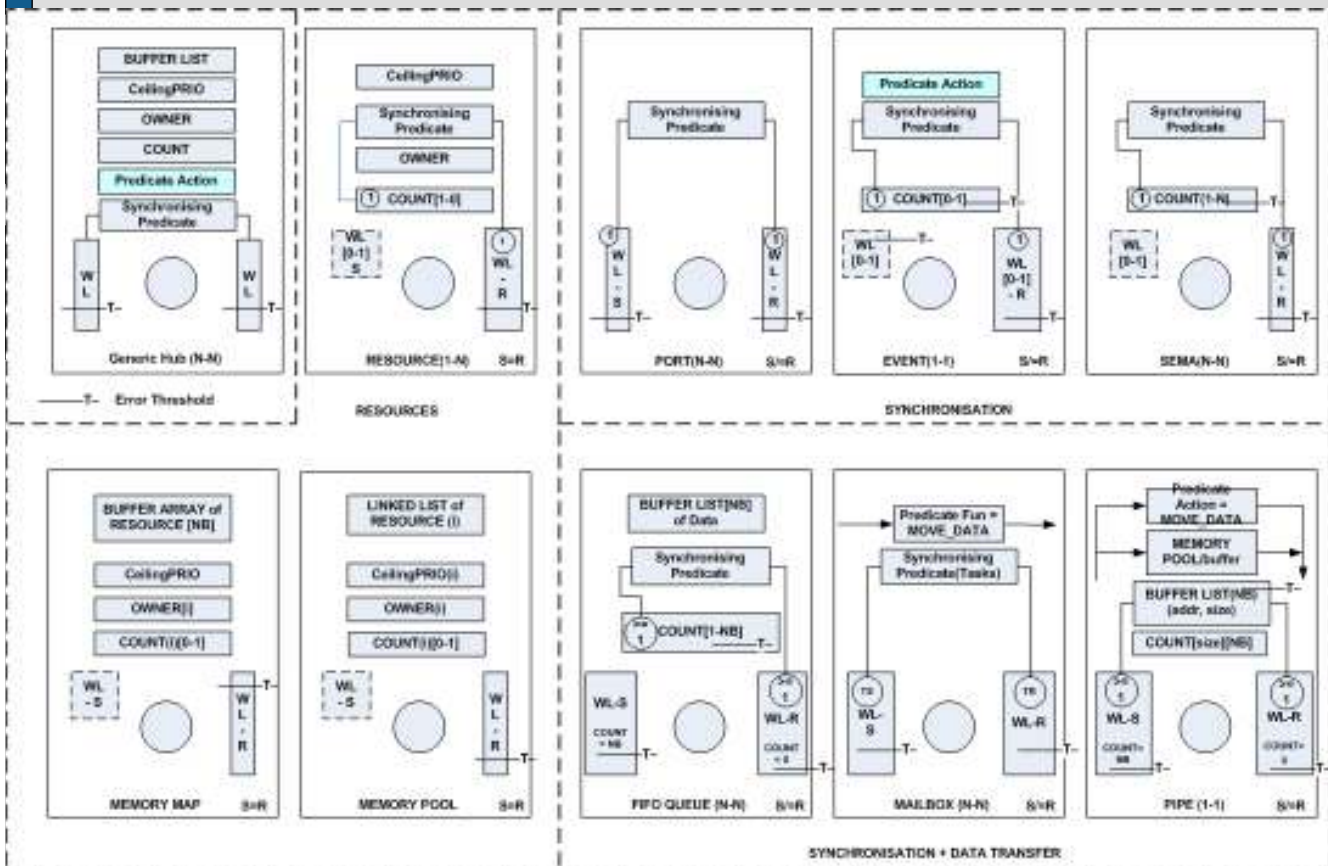
26/05/2008

www.OpenLicenseSociety.org

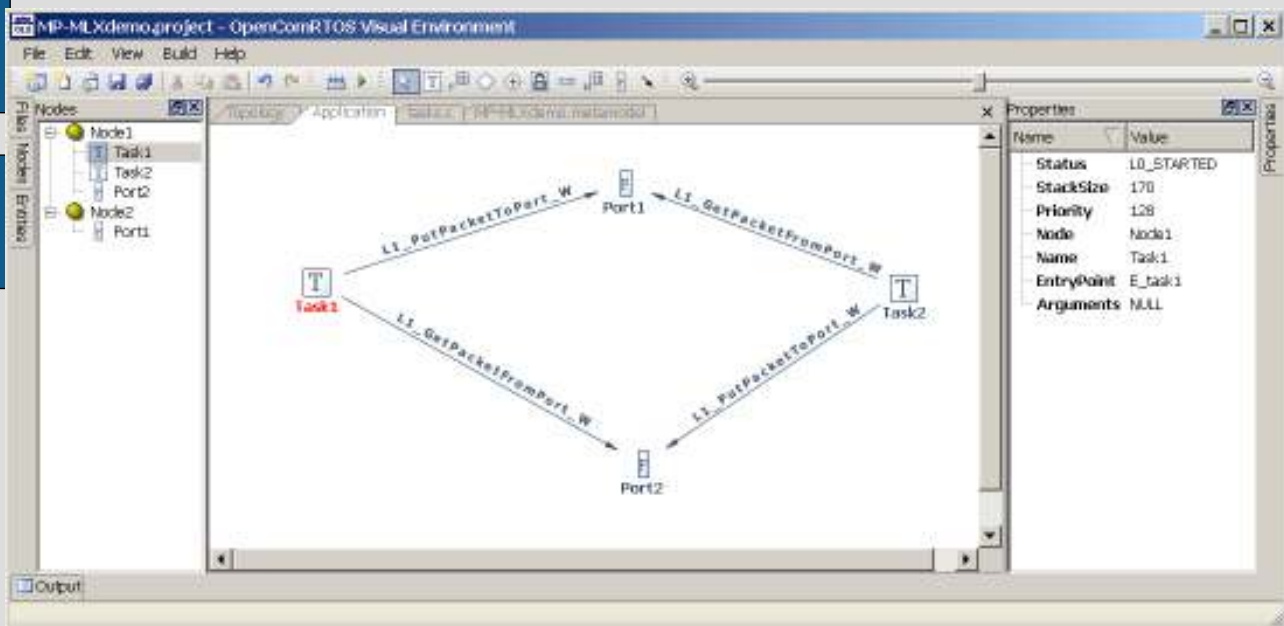
15



# All RTOS entities are "HUBs"



# Application's view



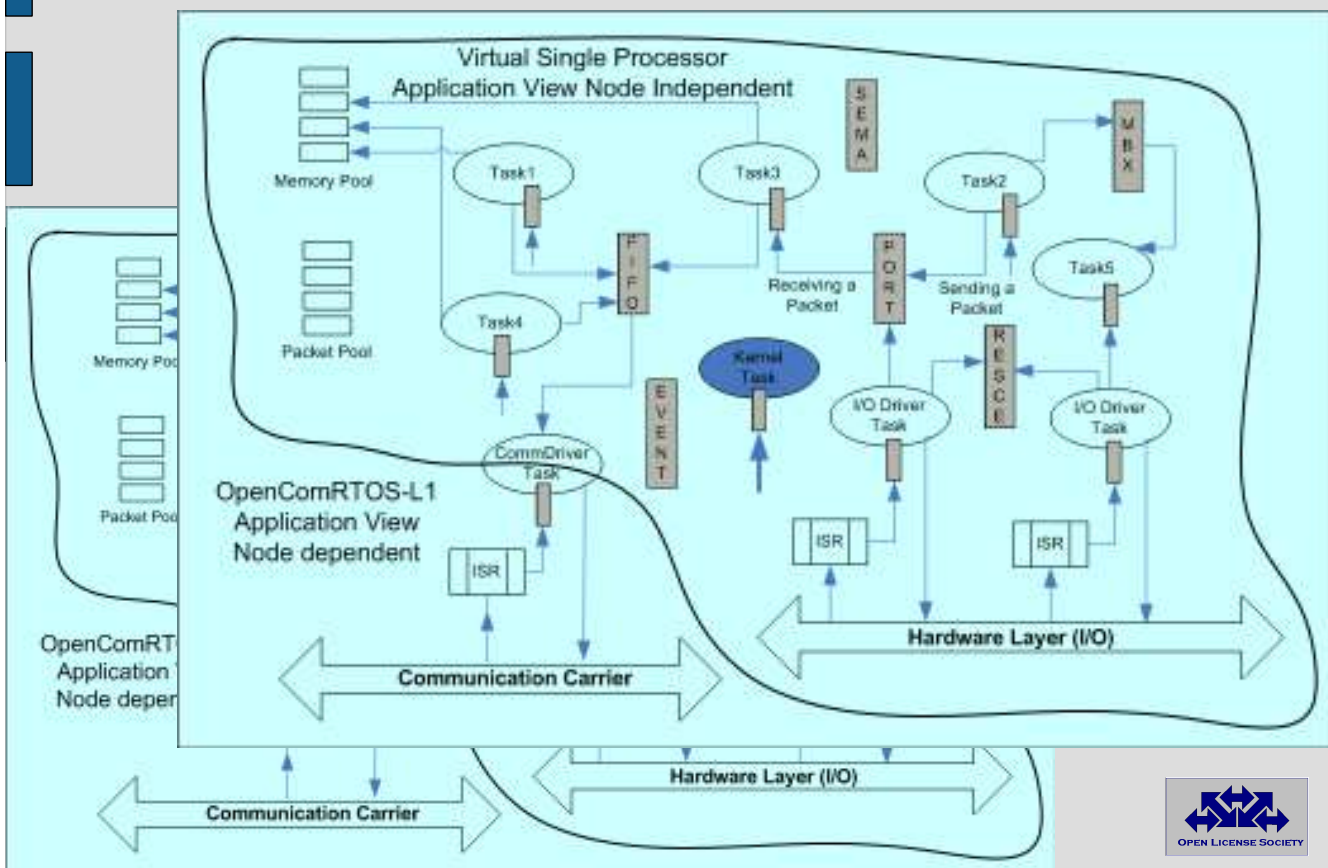
26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

17



L1 application view:  
any entity can be mapped onto any node



## Clean architecture gives small code

| OpenComRTOS L1 code size figures (MLX16) |         |      |          |      |
|--|---------|------|----------|------|
|  | MP FULL |      | SP SMALL |      |
|  | L0      | L1   | L0       | L1   |
| L0 Port                                  | 162     |      | 132      |      |
| L1 Hub shared                            |         | 574  |          | 400  |
| L1 Port                                  |         | 4    |          | 4    |
| L1 Event                                 |         | 68   |          | 70   |
| L1 Semaphore                             |         | 54   |          | 54   |
| L1 Resource                              |         | 104  |          | 104  |
| L1 FIFO                                  |         | 232  |          | 232  |
| L1 Resource List                         |         | 184  |          | 184  |
| Total L1 services                        |         | 1220 |          | 1048 |
| Grand Total                              | 3150    | 4532 | 996      | 2104 |

Smallest application: 1048 bytes program code and 198 bytes RAM (data) (SP, 2 tasks with 2 Ports sending/receiving Packets in a loop, ANSI-C)  
Number of instructions : 605 instructions for one loop (= 2 x context switches, 2 x L0\_SendPacket\_W, 2 x L0\_ReceivePacket\_W)

## Results

- Break-through results in well-known domain
  - 100's of RTOS with such support
  - 15 years of experience, 3 generations of RTOS design
  - Typically CPU dependent, use of assembler and async operation
- Small, scalable, distributed and maintainable code
  - SP(L0): < 1000 machine instructions
  - MP(L1): < 2000 - 5000 machine instructions
  - Yet all typical RTOS services ( \_NW|W|WT|Async )
  - Needs a few 100 bytes of data RAM
  - Fully in ANSI-C, MISRA-C compliant
  - Runs on MelexCM (16 bit), AVR, Win32, Linux, (Sparc, uBlaze, ...)
  - Scheduling algorithm can be improved to reduce worst-case rescheduling latency and blocking time
  - All RTOS Entities are variations of a generic « hub » entity
    - => less but faster code: 5 KBytes vs. 50 KBytes before

## How it really works: teamwork

Requirements

Specifications

Validation

Test and profiling

Implementation Models

Formal Models

How ?

Concept

Informal Models

Discuss,  
think,  
review

Formalise  
!

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

21



## From theoretical concept to products



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

26/05/2008

[www.OpenLicenseSociety.org](http://www.OpenLicenseSociety.org)

22

