

CPA 2007 Fringe

*Formal Modeling gives a Network Centric
Real-Time Operating System
in less than 2K Bytes as a generic base
for MP-SoC and
Process Oriented Programming*

www.OpenLicenseSociety.org
www.melexis.com

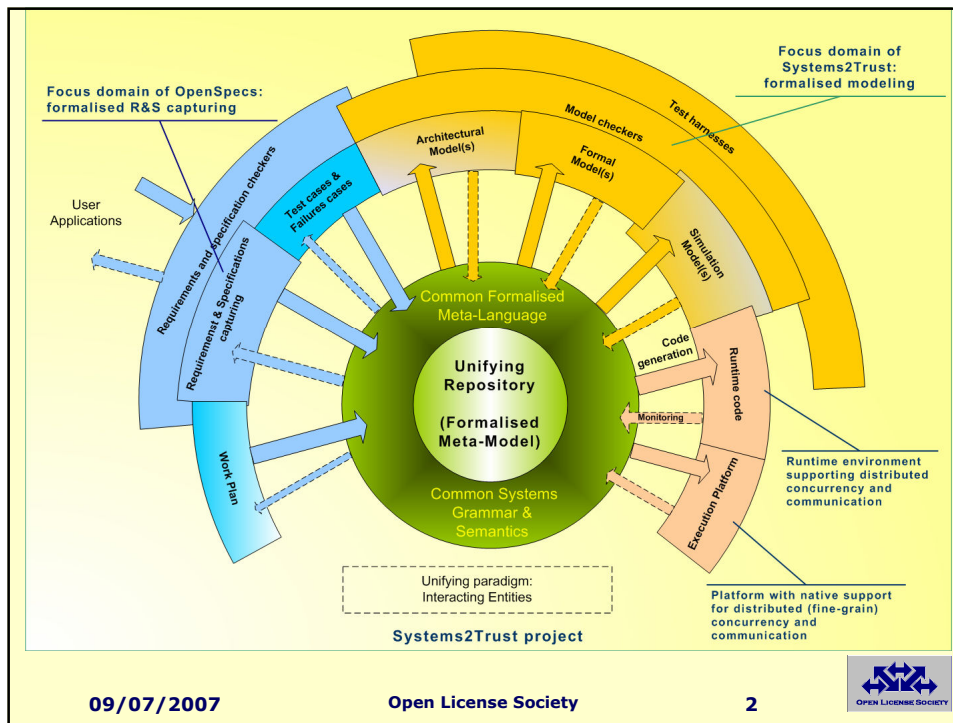
Unifying and systematic system development methodologies
with trustworthy embedded components

Eric.Verhulst@OpenLicenseSociety.org

09/07/2007

Open License Society

1



Some keywords

- **Unified semantics:**
 - For all "views" (from requirements to platform)
- **Meta-modeling**
 - Raising the level of abstraction first
- **System's grammar**
 - Defining a formalised language
 - Interface definition (is more than syntax)
 - "protocols" rather than messages
- **Interacting Entities**
 - Then define the architectural level=
 - Entities and Interactions
 - Applies to almost any system domain

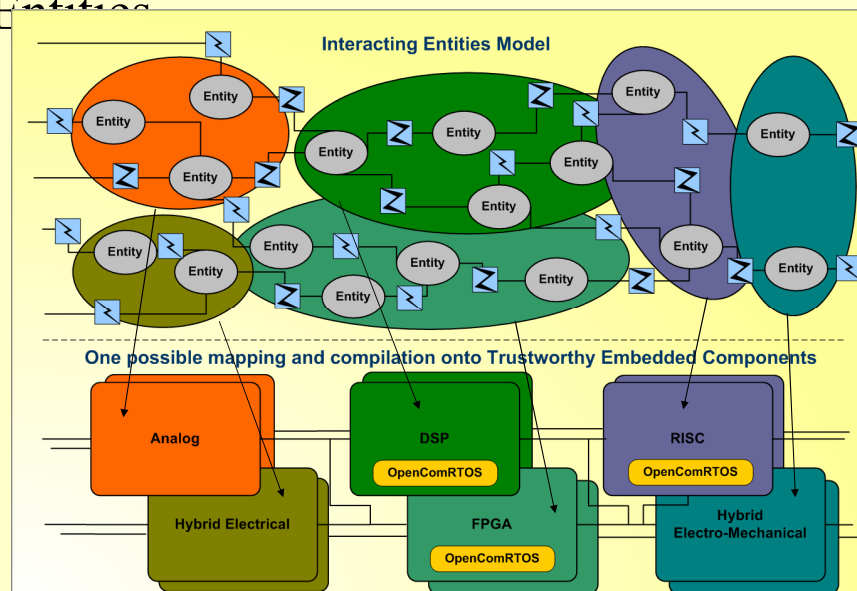
09/07/2007

Open License Society

3



Unifying paradigm (1): Interacting Entities



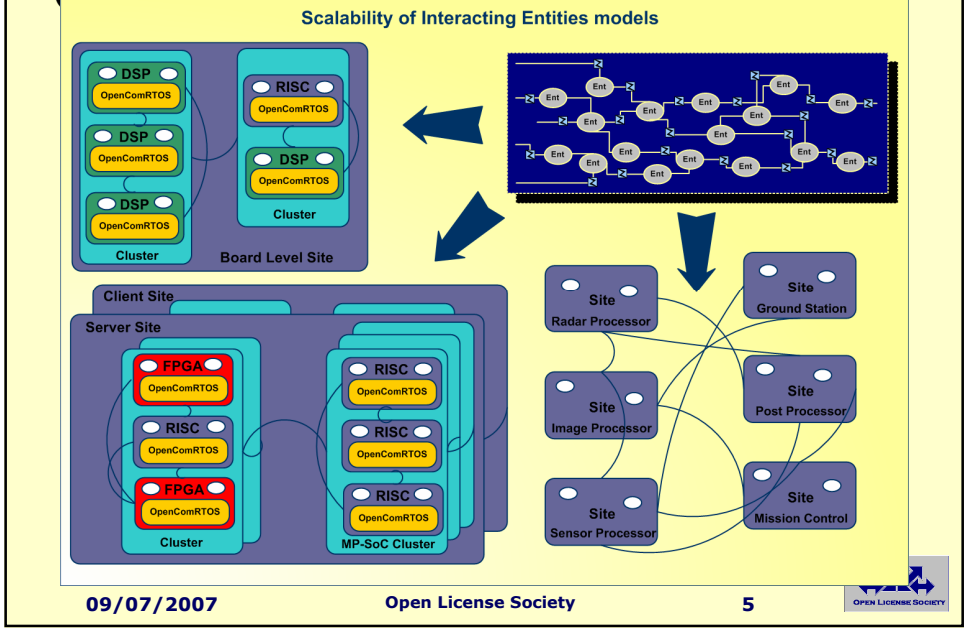
09/07/2007

Open License Society

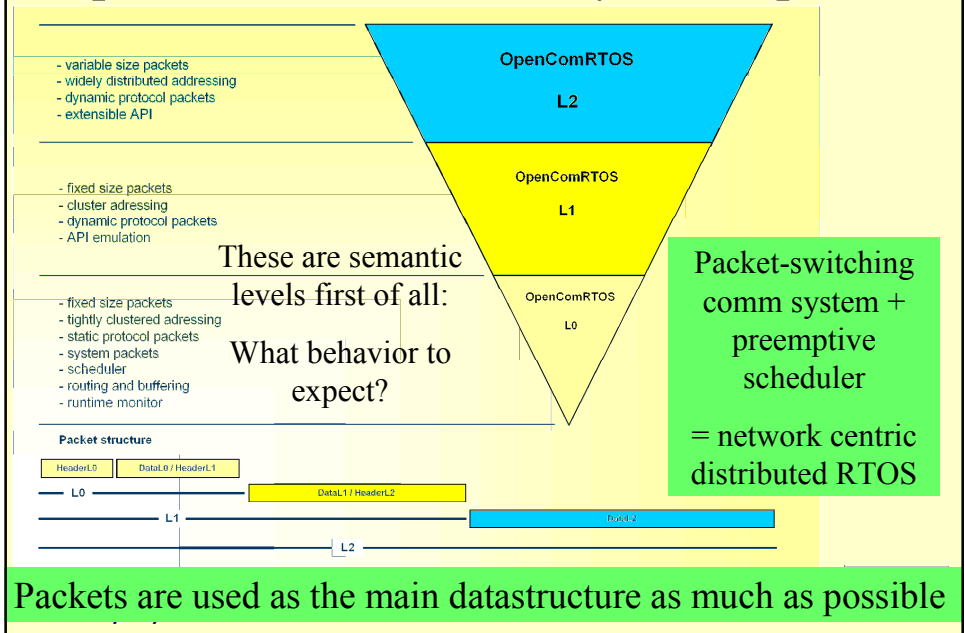
4



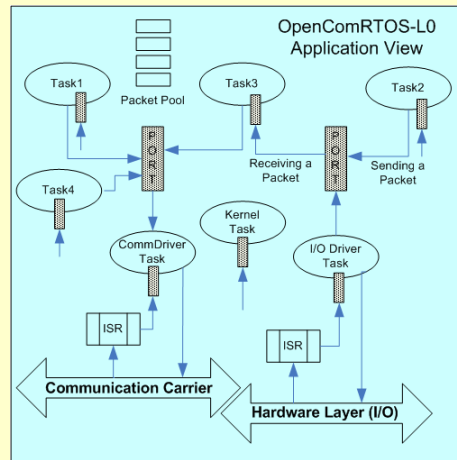
Unifying paradigm (2): Scalable Communication



OpenComRTOS: formally developed



Generic Open-Comm-RTOS



Based on (scalable) "packet switching" at all levels
Tasks (entities) and interactions decoupled

09/07/2007

Open License Society

7



Can we trust our mind ?

- How many « F » did you find ?

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) ?

Who found errors in the definition of occam-2 ?

09/07/2007

Open License Society

8



Formally modeled in TLA: example

```

TypeInvariant == /\ ppool \in [Adr-> Packet \union {NoData}]
                 /\ PQ \in [ FIFO : [Port -> Seq(Adr)],
                             WL  : [Port -> Seq(Adr)]]
                 /\ chan \in [ val: [HLink -> Packet \union {NoData}],
                             stt: [HLink -> {"free", "busy"}]]
                 /\ TxQ \in [TxChan -> Seq(Packet)]
                 \*   /\ tstate \in [UTask -> {"running", "ready", "wait4anS", "wait4anR"}]
    
```

```

67 TypeInvariant  $\triangleq$   $\wedge$  ppool  $\in$  [Adr  $\rightarrow$  Packet  $\cup$  {NoData}]
69
70    $\wedge$  PQ  $\in$  [FIFO : [Port  $\rightarrow$  Seq(Adr)],
71              WL  : [Port  $\rightarrow$  Seq(Adr)]]
72
73    $\wedge$  chan  $\in$  [ val: [HLink  $\rightarrow$  Packet  $\cup$  {NoData}], stt: [HLink  $\rightarrow$  {"free", "busy"}]]
74
75    $\wedge$  TxQ  $\in$  [TxChan  $\rightarrow$  Seq(Packet)]
76
77    $\wedge$  tstate  $\in$  [UTask  $\rightarrow$  {"running", "ready", "wait4anS", "wait4anR"}]
    
```

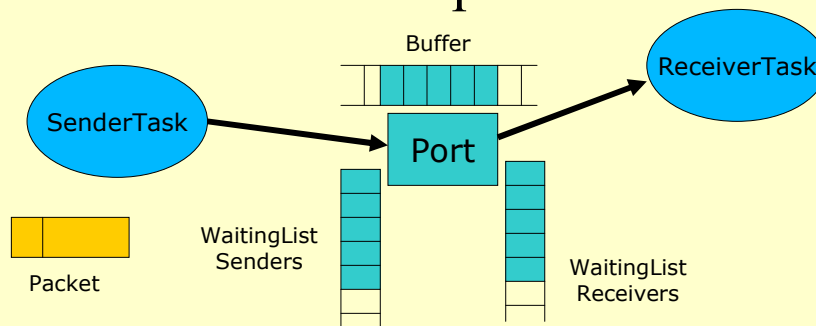
09/07/2007

Open License Society

9



One result as example



- Need for either FIFO Buffer or WaitingList
 - Both (abstract) models are the same
 - Natural language is imprecise, semantics are context driven
- Benefits:
 - Infinite buffering until no more memory (for Packets)
 - Overflow-free buffering

09/07/2007

Open License Society

10



L1 entities

L1 Entity	Semantics
Event	Synchronisation on Boolean value. Waiting list on both sides.
Counting Semaphore	Synchronisation with counter allowing asynchronous signaling.
Port	Synchronisation with exchange of a Packet.
FIFO queue	Buffered communication of Packets. Synchronisation when queue is full or empty.
Resource	Event used to create a logical critical section. Resources have an owner Task when locked
Memory Pool	Linked list of memory blocks protected with a resource
Mailbox	Synchronising entity with matching filter on Task ID. Communication happens as side-effect.
Channel	Asynchronous communication between Tasks with buffering using memory pools. Communication as a side-effect.

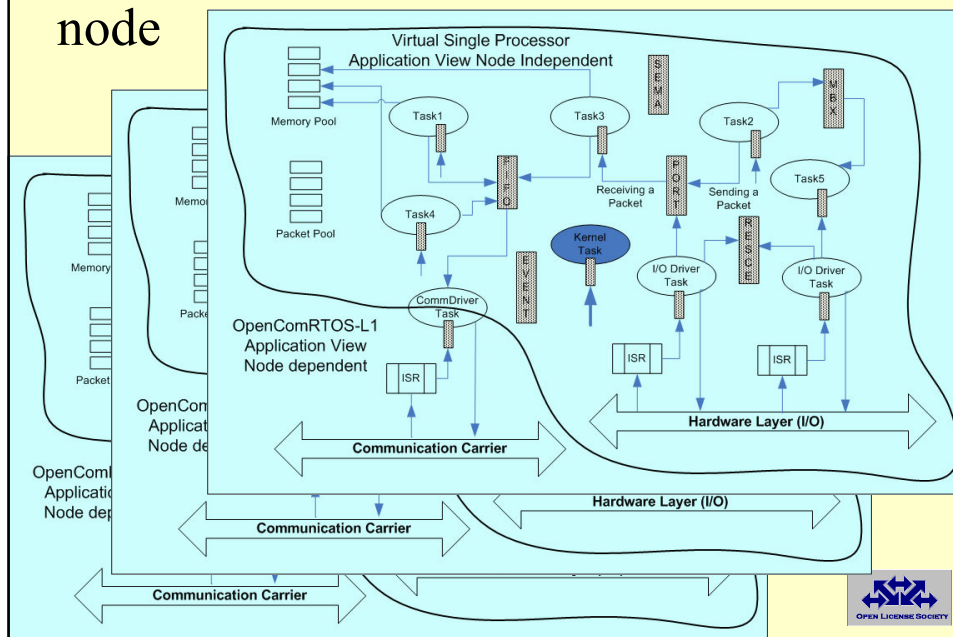
09/07/2007

Open License Society

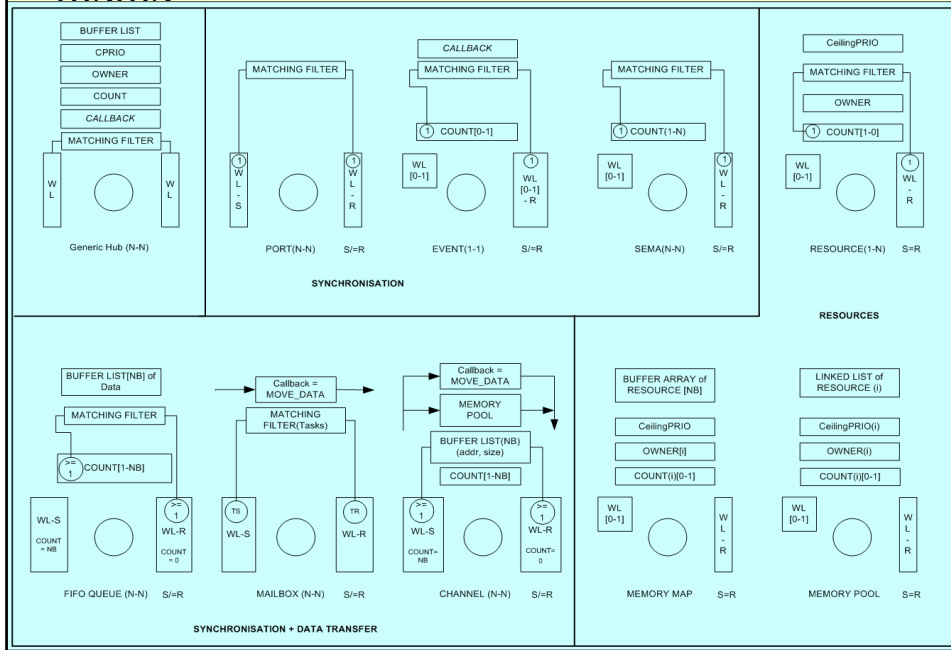
11



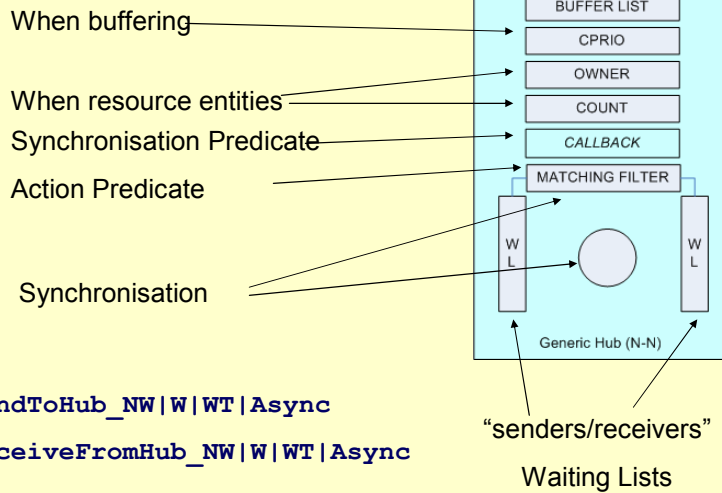
any entity can be mapped onto any node



All (typical) RTOS Entities: variations on a theme



Generic hub: => define your own entities and interactions



API:

L1_SendToHub_NW|W|WT|Async

L1_ReceiveFromHub_NW|W|WT|Async

Clean architecture gives small code (all in ANSI-C)

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
Total with core kernel	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)

Semantic variations

Services variants	Synchronising Behaviour
"Single-phase" services	
_NW	Non Waiting: when the matching filter fails the Task returns with a RC_Failed
_W	Waiting: when the matching filter fails the Task waits until such events happens.
_WT	Waiting with a time-out. Waiting is limited in time defined by the time-out value.
"Two-phase" services	
_Async	Asynchronous: when the entity is compatible with it, the Task continues independently of success or failure and will resynchronize later on. This class of services is called "two-phase" services.

Classes of services: API

- L0_Start/Stop/Suspend/ResumeTask
- L0_SetPriority
- L1_SendTo/ReceiveFromHub
- L1_Raise/TestForEvent_(N)W(T)_Async
- L1_Signal/TestSemaphore_X
- L1_Send/ReceivePacket_X L1_WaitForAnyPacket_X
- L1_Enqueue/DequeueFIFO_X
- L1_Lock/UnlockResource_X
- L1_Allocate/DeallocatePacket_X
- L1_Get/ReleaseMemoryBlock_X
- L1_MoveData_X
- L1_SendMessageTo/ReceiveMessageFromMailbox_X
- L1_SetEventTimerList
- ... => user can create his own service!

09/07/2007

Open License Society

17



OpenComRTOS Visual Environment (beta)

The screenshot displays the OpenComRTOS Visual Environment interface. It includes a task tree on the left, a configuration table for 'Node_C' in the center, a code editor on the right showing C code for a task, and a process tracer at the bottom showing a timeline of task execution and system calls.

Name	Value
Node	Node_C
Priority	6
EntryPoint	app_task
Name	app_task0
Arguments	NULL
StackSize	100
Status	LO_STARTED

```

Configuration
LO_Timer LO_TimerPool[LO_NODE_NUMBER_OF_TIMERS];
#ifdef /* APTM_SERVICES */
#include <lo_api.h>
#include <lo_dbg_api.h>
#endif
int main ()
{
#ifdef LO_DEBUG
    otftracefile = stdout;
    tracefile = stdout;
#endif /* LO_DEBUG */
return LO_RunOpenComRTOS (LO_NODE_NUMBER_OF_TASKS,
    LO_NODE_NUMBER_OF_PACKETS,
    LO_NODE_NUMBER_OF_SOURCE_PACKETS,
    LO_NODE_NUMBER_OF_TIMERS
);
}
    
```

Results

- Break-through results in well-known domain
 - 100's of RTOS with such support
 - 15 years of experience, 3 generations of distributed RTOS design (Virtuoso RTOS – Eonic Systems)
 - 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
 - Needs a few 100 bytes of data RAM
 - Fully in ANSI-C, MISRA-C compliant
 - Runs on MelexCM (16 bit) and Windows, ports underway (Cell, Sparc, uBlaze, ARM, PCI-Express) using porting kit
 - User can add his own application specific services
 - Improved scheduling algorithm reducing worst-case rescheduling latency and blocking time
 - All RTOS Entities are variations of a generic « hub » object
 - => less but faster code: **5 KBytes vs. 50 KBytes before**
 - **RT performance @ 5 Mips, needed 50 Mips before**

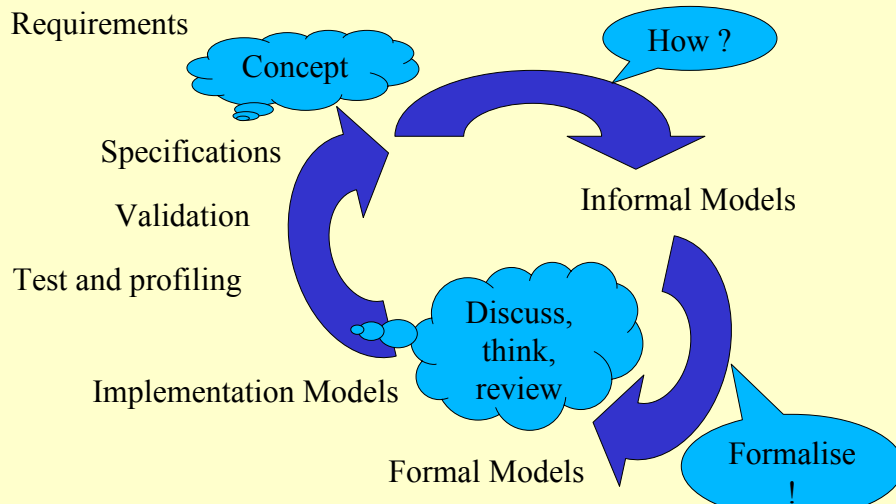
09/07/2007

Open License Society

19



How it really works: teamwork



09/07/2007

Open License Society

20

