



Open License Society

www.OpenLicenseSociety.org

Unifying and systematic system development methodologies
with trustworthy embedded components

Eric.Verhulst@OpenLicenseSociety.org

01/09/2007

Open License Society

1



Interacting Entities as a Unified Systems Engineering paradigm

and

its application using formal modeling in the
development of a distributed embedded RTOS

01/09/2007

Open License Society

2



Who is Open License Society?

- Privately funded R&D institute
 - Leuven (BE), Berdyansk (UA)
 - Industrial sponsors
 - IWT project funding for OpenComRTOS
- 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, ...

01/09/2007

Open License Society

3



Part 1

The Wall of Complexity challenge:

Because it is all connected

01/09/2007

Open License Society

4



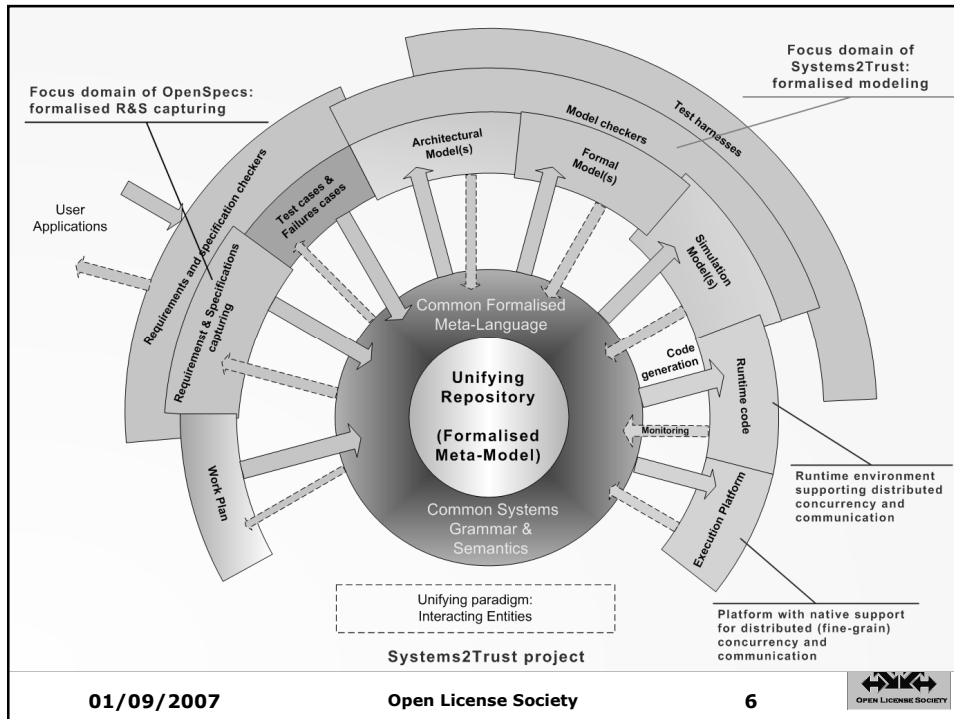
The Wall of Complexity challenge

- Products become systems
 - Smart systems with 10's of distributed processors
 - Embedded systems are essential and life-critical
 - High quality, high reliability, scalability and security
 - High level of safety, fault-tolerance, graceful degradation
- Murphy's law
 - things going wrong is a matter of probability
 - the odds are getting worse
- Business context
 - Cost-efficient
 - Competitive
 - Upgradeable
- Trustworthy Components
 - Formally developed and validated SW & IP
 - Engineering vs crafting
 - Rid of unnecessary complexity and legacy
 - Higher quality, less life-cycle costs, higher added-value

01/09/2007

Open License Society

5



01/09/2007

Open License Society

6



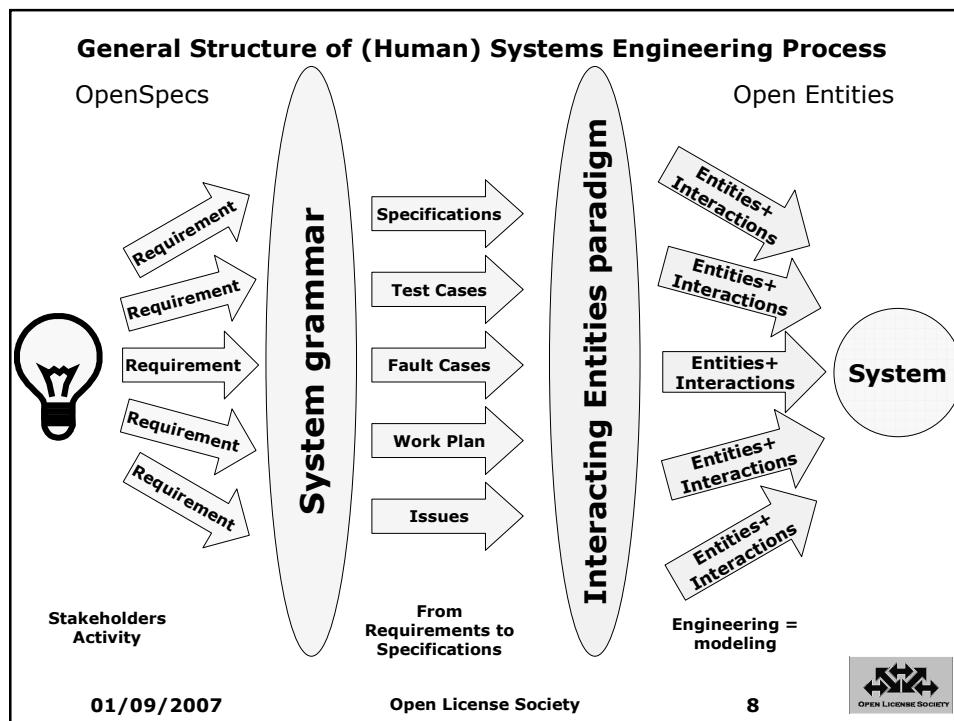
Part 2

How do we get
from ideas to products ?

01/09/2007

Open License Society

7



Fase 1: Requirements & Specifications capturing

- From natural language Requirements to Specifications in formalised language
 - => including Test Cases, Failure Cases
- System = All aspects, incl. workflow
- Minimum system = (3 Entities)+Interactions:
 - System under development (the black box)
 - Environment (not controlled)
 - Operator (interfering or reacting)
- Tools to verify consistency and completeness

01/09/2007

Open License Society

9



Requirements: natural language is fuzzy !

It is foreseen that a alarm signal be raised once pressure levels exceed about the acceptable safety level

- foreseen: is this a mandatory requirement ?
- alarm signal: what? light, sound- ... ?
- raised: how? duration, intensity, ...
- pressure levels: which ones ? how measured ?
- exceed: by how much ?
- about: accuracy ?
- acceptable safety level: units ? values ?
- when to switch off the alarm ?
- ...
 - => poor engineer who has to implement it
 - => poor customer who might not get what he wanted

01/09/2007

Open License Society

10



Specifications: should be unambiguous

1. An auditive signal of 90 dB (+/- 0.1 dB) will sound for 60 seconds when the main boiler pressure gauge subsystem detects an increasing pressure of more than 100 Bar for more than 3 consecutive seconds.
2. The alarm signal will be terminated when the pressure drops again below 90 Bar.
3. Pressure measurements will be updated every 100 milliseconds.
4. The pressure gauge subsystem must have an expected lifetime of 10 years.

Remark: how does one test it?

01/09/2007

Open License Society

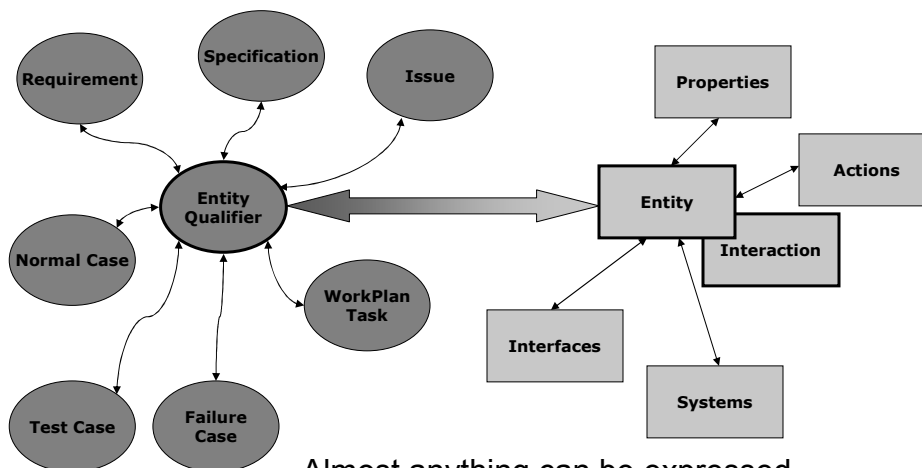
11



Cognitive and ontological levels

Requirements and Specifications capturing ("abstract")

Architectural and functional modeling ("concrete")



Almost anything can be expressed as a set of Interacting Entities

01/09/2007

Open License Society

12



The need for a SE metamodeling-language

- **Why?**
 - Gap between “natural language” and implementation
 - Natural language is not precise enough
- **Solution: “formalised language”**
 - Sentences following “systems grammar and semantics”
 - Independently of modeling domains: high level
 - Stick to essential concepts first, instances can be adapted to specific domain
 - Benefits: unique description
- **Challenges:**
 - Covers many “views” of systems engineering process
 - Can it be used as a “programming language” ?
 - Systems compiler(s) ?
 - Can it be descriptive and executable?

01/09/2007

Open License Society

13



Fase 2: from specifications to modeling

- Specifications are fulfilled by concrete architectural elements (N – N relationship)
- But before we implement we should model
 - This is what engineering is really about
- **Simulation modeling:**
 - What-if-analysis: are we building the right system ?
- **Architectural modeling:**
 - Decompose and define interface
- **Formal modeling:**
 - Verification: are we building the components right ?
- **Strong interaction between these models**

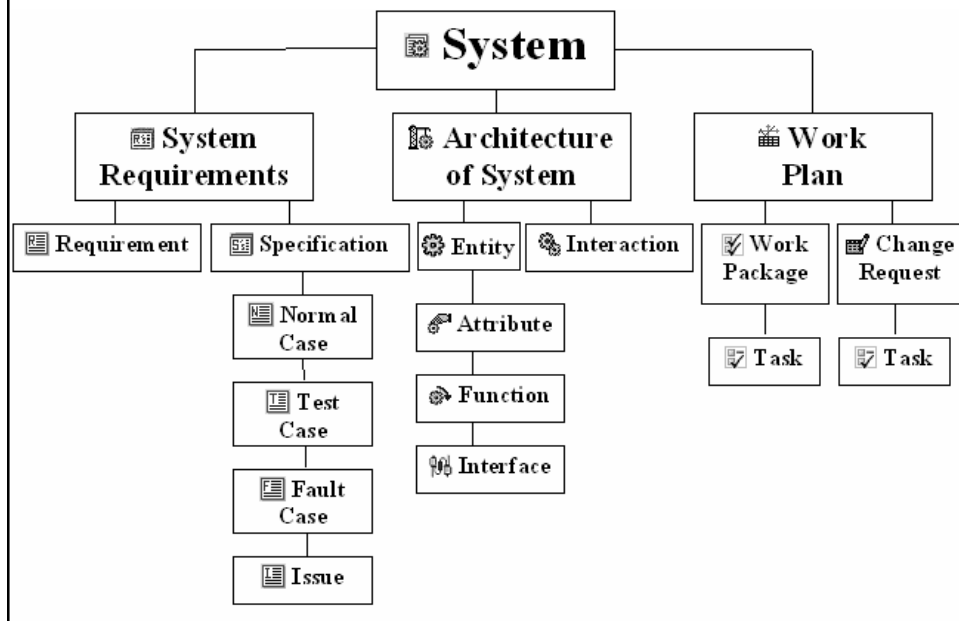
01/09/2007

Open License Society

14



Example Systems grammar in OpenSpecs



Example systems grammar

OpenComRTOS

```

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
  SystemTasks (2+N*2) 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.
  
```

Fase 3: From Modeling to Code

- Each model can be seen as a specific execution engine (virtual machine)
 - Writing at low level is error-prone
 - Each execution engine should be independent
- Solution is to insert metamodeling-layer
 - Using a HL metamodeling-language
 - Metamodeling language exploits meta-model
 - Formalised description expressed in formalised concepts, systems grammar and syntax, sentences
 - Generic but covering all views
- 'Systems compiler':
 - Preserve the properties from R&S till implementation
- Validation:
 - Did we build the right **system** and did we do it right ?

01/09/2007

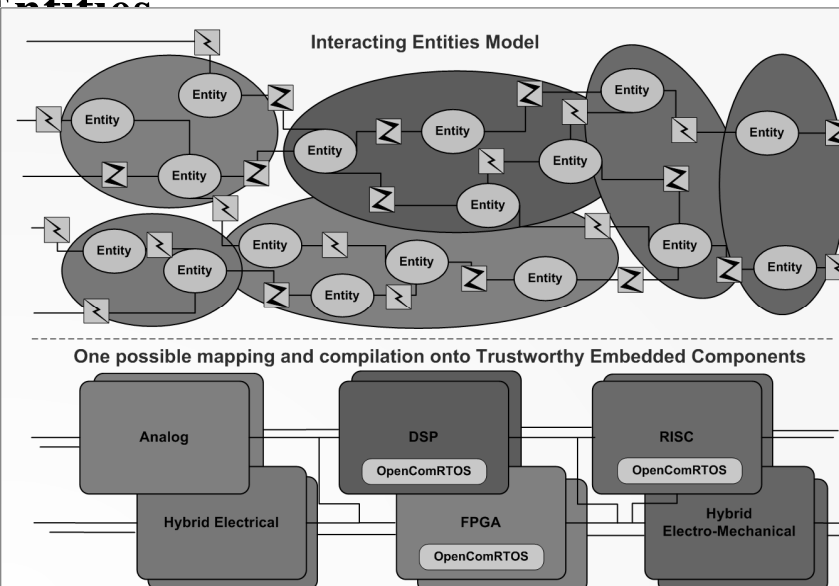
Open License Society

17



Unifying paradigm: Interacting

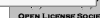
Entities



01/09/2007

Open License Society

18



Part 3

Interaction Entities at the runtime level:

Example of applying the formalised approach to the development of OpenComRTOS

01/09/2007

Open License Society

19



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

01/09/2007

Open License Society

20



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
 - Distributed real-time
 - Safe, secure
 - Small code size, low latency (=high performance)

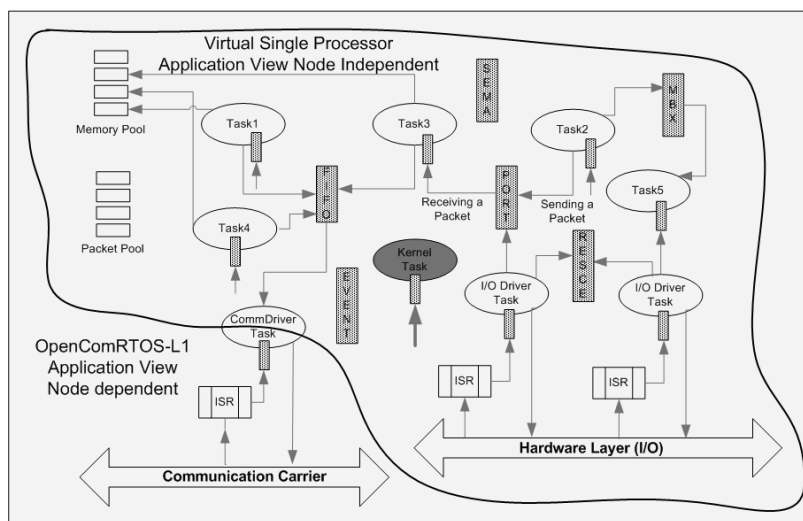
01/09/2007

Open License Society

21



L1 application view



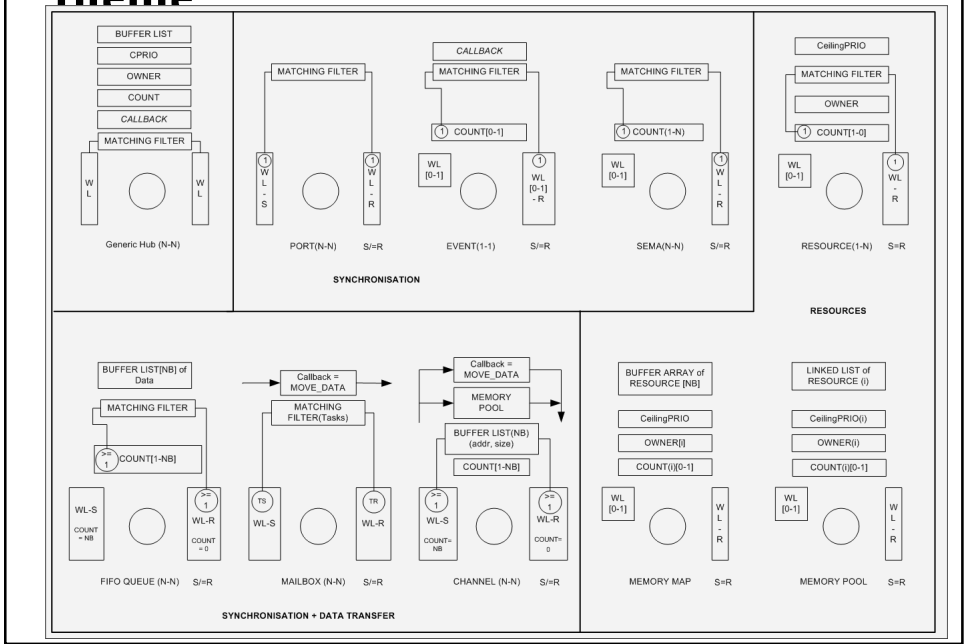
01/09/2007

Open License Society

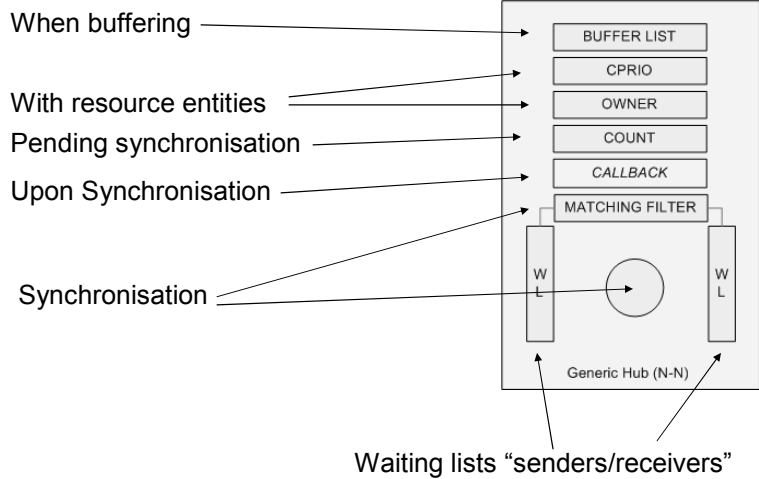
22



All RTOS Entities: variation on a theme



Generic hub entity



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
 - One of first RTOS developed using formal modeling
- 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
 - Scheduling algorithm could be improved to reduce 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

01/09/2007

Open License Society

26



Part 4

Formal TLA models of OpenComRTOS entities

01/09/2007

Open License Society

27



OpenComRTOS Metamodel

- Based on Interacting Entities Paradigm
- Application can be constructed from trustworthy 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

01/09/2007

Open License Society

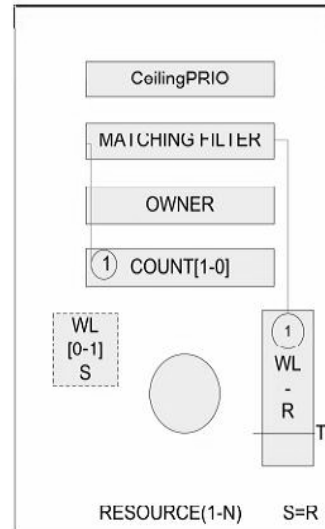
28



TLA modelling results

We modeled entities
of OpenComRTOS:

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



01/09/2007

Open License Society

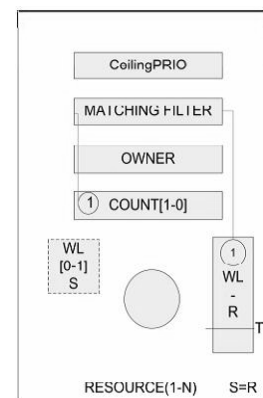
29



Example: Resource

Verified Properties:

- No Task waiting for an Resource can be ready
- Any Task that is ready cannot be in a waiting condition
- When the resource is free, then no Task will be waiting for it

$$\begin{aligned}
 &\wedge \forall p \in ResourceId : \\
 &\forall i \in 1 .. Len(ResourceWL[p]) : \\
 &ResourceWL[p][i] \notin ReadyList \\
 &\wedge \forall t \in TaskId : \\
 &(t \in ReadyList) \Rightarrow (\forall i \in ResourceId : \\
 &\forall j \in 1 .. Len(ResourceWL[i]) : \\
 &PreallocatedPacket[ResourceWL[i][j]].RequestingTaskID \neq t) \vee \\
 &(\forall i \in 1 .. Len(KernelPortWL) : \\
 &PreallocatedPacket[KernelPortWL[i]].RequestingTaskID \neq t) \\
 &\wedge \forall p \in ResourceId : \\
 &\wedge isResourceAvailable(p) \Rightarrow List_isEmpty(ResourceWL[p])
 \end{aligned}$$


01/09/2007

Open License Society

30



OpenComRTOS Entities

Metamodel

= RTOS hub instances

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Metamodel>
  <Entity Name="Task" SvgPath="Task.svg">
    <Attribute Name="Name" Type="String"/>
    <Attribute Name="Priority" Type="Integer" MinValue="1" MaxValue="255"/>
    <Attribute Name="Arguments" Type="String" DefaultValue="NULL"/>
    <Attribute Name="Status" Type="Enum" Values="LO_INACTIVE:LO_STARTED" DefaultValue="LO_STARTED"/>
    <Attribute Name="Node" Type="Node"/>
    <Attribute Name="StackSize" Type="Integer" DefaultValue="170"/>
    <Function Name="EntryPoint"/>
  </Entity>
  <Entity Name="Port" SvgPath="Port.svg">
    <Attribute Name="Name" Type="String"/>
    <Attribute Name="Node" Type="Node"/>
  </Entity>
  <Entity Name="Event" SvgPath="Event.svg">
    <Attribute Name="Name" Type="String"/>
    <Attribute Name="Node" Type="Node"/>
  </Entity>
  <Entity Name="Semaphore" SvgPath="Semaphore.svg">
    <Attribute Name="Name" Type="String"/>
    <Attribute Name="Node" Type="Node"/>
  </Entity>
  <Entity Name="Hub" SvgPath="Hub.svg">
    <Attribute Name="Name" Type="String"/>
    <Attribute Name="Node" Type="Node"/>
    <Attribute Name="Type" Type="Enum" Values="L1_PORT:L1_EVENT:L1_SEMAPHORE:L1_RESOURCE:L1_FIFO:L1_P
    <Function Name="UpdateFunction"/>
    <Function Name="SyncFunction"/>
  </Entity>

```

01/09/2007

Open License Society

31



OpenComRTOS Interactions'

Metamodel = kernel services

```
<Interaction Name="L1_StartTask_w" Subject="Task" Object="Task" />
<Interaction Name="L1_StopTask_w" Subject="Task" Object="Task" />
<Interaction Name="L1_SuspendTask_w" Subject="Task" Object="Task" />
<Interaction Name="L1_ResumeTask_w" Subject="Task" Object="Task" />
<Interaction Name="L1_SendPacket_w" Subject="Task" Object="Port" />
<Interaction Name="L1_ReceivePacket_w" Subject="Task" Object="Port" />
<Interaction Name="L1_SendPacket_NW" Subject="Task" Object="Port" />
<Interaction Name="L1_ReceivePacket_NW" Subject="Task" Object="Port" />
<Interaction Name="L1_SendPacket_WT" Subject="Task" Object="Port" />
<Interaction Name="L1_ReceivePacket_WT" Subject="Task" Object="Port" />
<Interaction Name="L1_SendPacket_A" Subject="Task" Object="Port" />
<Interaction Name="L1_ReceivePacket_A" Subject="Task" Object="Port" />
<Interaction Name="L1_AllocatePacket" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_DeallocatePacket_w" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_AllocatePacket_w" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_WaitForPacket" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_WaitForPacket_w" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_AllocatePacket_NW" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_WaitForPacket_NW" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_AllocatePacket_WT" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_WaitForPacket_WT" Subject="Task" Object="PacketPool" />
<Interaction Name="L1_SendPacket_w" Subject="Task" Object="Port" />
<Interaction Name="L1_ReceivePacket_w" Subject="Task" Object="Port" />
<Interaction Name="L1_RaiseEvent_w" Subject="Task" Object="Event" />
<Interaction Name="L1_TestEvent_w" Subject="Task" Object="Event" />
<Interaction Name="L1_SignalSemaphore_w" Subject="Task" Object="Semaphore" />
<Interaction Name="L1_TestSemaphore_w" Subject="Task" Object="Semaphore" />
<Interaction Name="L1_LockResource_w" Subject="Task" Object="Resource" />
<Interaction Name="L1_UnlockResource_w" Subject="Task" Object="Resource" />
<Interaction Name="L1_EnqueueFifo_w" Subject="Task" Object="Fifo" />
<Interaction Name="L1_DequeueFifo_w" Subject="Task" Object="Fifo" />
<Interaction Name="L1_SendPacket_NW" Subject="Task" Object="Port" />

```

01/09/2007

Open License Society

32



OpenComRTOS Visual Environment (beta)

The screenshot displays the OpenComRTOS Visual Environment interface. It includes a task graph, a configuration table, a code editor, and a process tracer.

Timestamp	ID
1 2633724335	2
2 2633725805	0
3 2633725941	0
4 2633726109	0
5 2633726320	0
6 2633726328	3
7 2633726629	0
8 2633726719	0

Key observations

Interacting Entities promotes Evolutionary Approach

- Successive iterations: evolutionary
 - > 28 successive models from 2 pages to 25 pages
 - Initially very abstract, neglecting details
 - All successive models were correct, why ?
 - Iterative, incremental process! 15 minutes from one model to the next
- Interaction and abstraction
 - Interplay between SW architects and formal modeling engineer
 - Architectural model polluted by programming concepts
 - Abstraction from TLA helped to find these issues
 - Formalised thinking
- Much cleaner, safer and performant architecture
- Caveat: FM do not prove software is correct (! ?)
 - Proves that Formal **Models** are correct

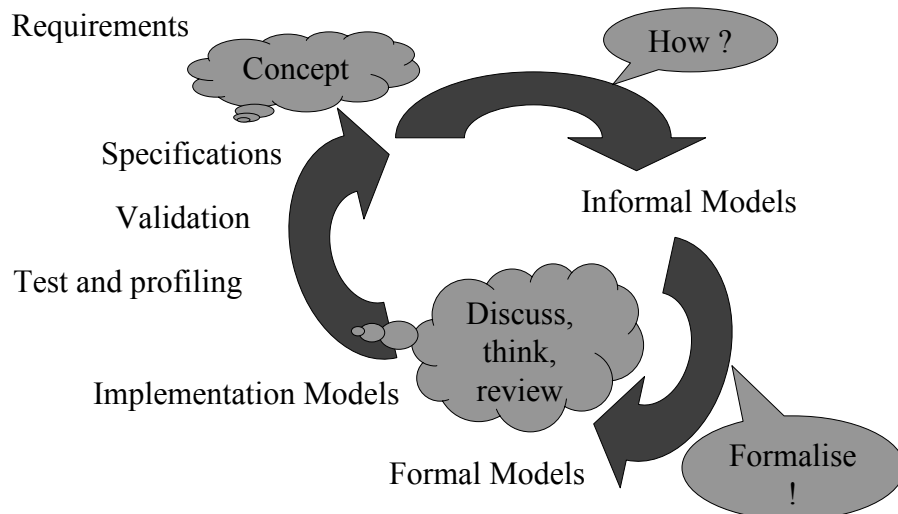
01/09/2007

Open License Society

35



How it really works: teamwork



01/09/2007

Open License Society

36



Summary

- Open License Society's approach is about 'formalised thinking'
- The essence is the SE process
 - not the tools, but they help a lot
 - Applying occam's rule: find the minimal solution
 - Teamwork and continuous feedback loops through review
- The benefits are "things being done better"
 - More clean, safe and performant architectures
- OpenComRTOS project has proven the correctness of the approach with break-through results

- Contact:
eric.verhulst@OpenLicenseSociety.org

01/09/2007

Open License Society

37

