



Open License Society

www.OpenLicenseSociety.org

Unifying and systematic system development methodologies
with trustworthy embedded components

Eric.Verhulst@OpenLicenseSociety.org

From very small to very large

- **Targets:**

- Deeply embedded distributed systems
- But linked with rest of the world
- Heavy constraints: real-time, memory, ...

- **Examples**

- **Automotive:**
 - 50-100 nodes
 - E.g. Tire pressure sensor: A/D, uC, 2KB, wireless interface
 - Connected wirelessly with manufacturer
- **EU security GRID**
 - Millions of sensors, real-time analysis, storage, ...
 - Widely distributed
 - Fully linked, incl. with back-office on-line databases,...

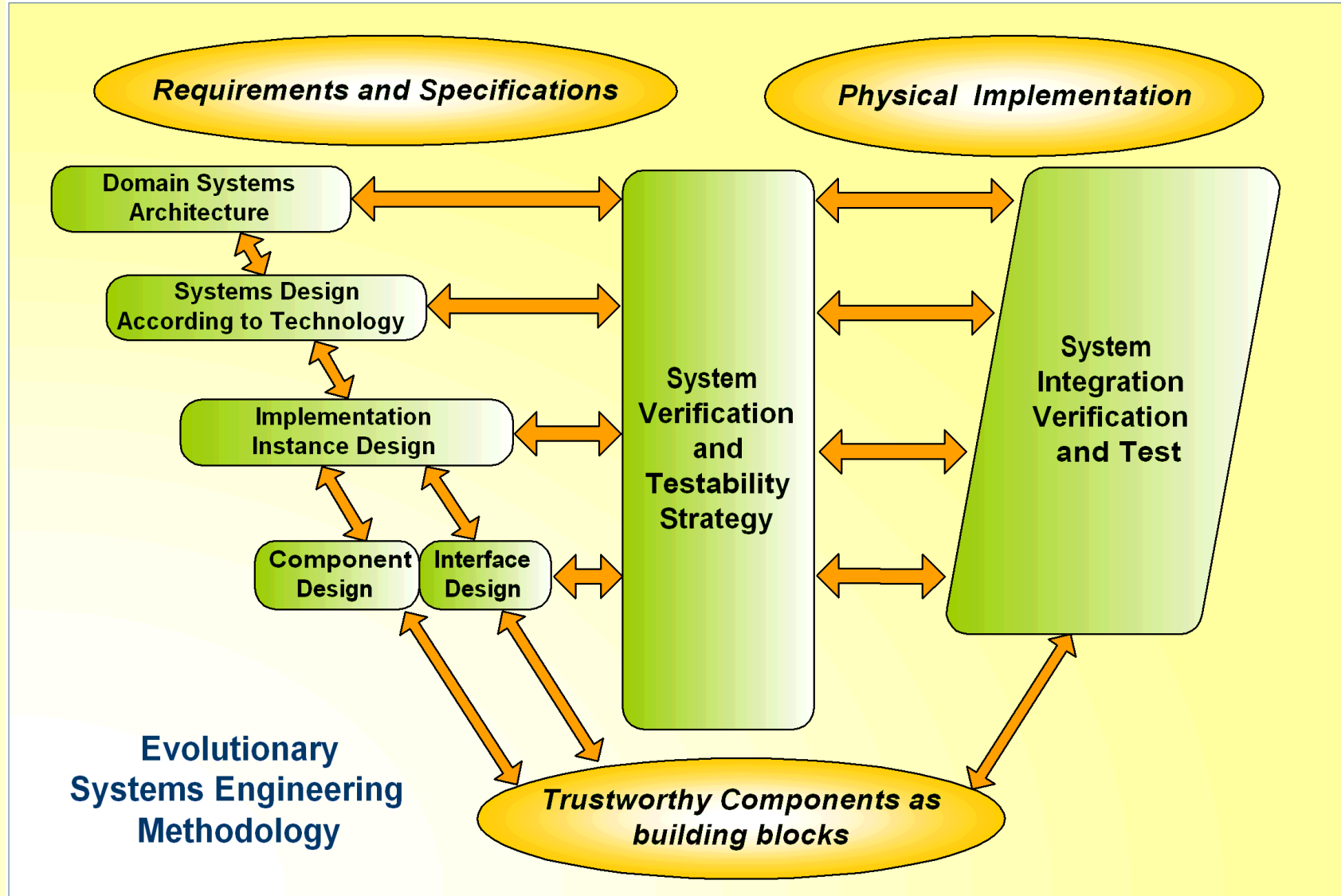
The Wall of Complexity challenge(1)

- Murphy's law: things going wrong is a matter of probability, and the odds are getting worse
- Focus domain: embedded systems
- Products become systems:
 - ,Smart` by using 10`s of processors
 - Distributed operation
 - Connected to other systems (wireless)
 - Human in the loop
 - Requirements:
 - High quality, high reliability
 - High level of safety, fault-tolerance
 - Secure operation
 - And as well:
 - Cost-efficient (life-cycle cost)
 - Competitive
 - Upgradeable

The Wall of Complexity challenge(2)

- The solution is NOT to link the myriad of existing processors, software tools, etc.
- Because they are semantically too different
- Because we have too many of them
- But none supports scalability requirement
- Time to apply occam`s razor:
 - Back to basics
 - Get rid of unnecessary complexity and historical ballast
 - More engineering, less crafting, more scalability and real re-use
- => ,Trustworthy Embedded Components`
 - Reliability, correctness
 - Safety, fault-tolerance
 - Security
 - Formally developed and validated software & IP
 - Open License: source code + validation & design data

Evolutionary Systems Engineering



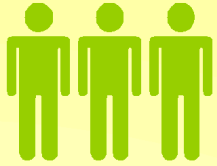
Evolutionary: what does it mean?

- Waterfall and V-method are less applicable to dynamic markets because the feedback loops are now much shorter
- Big bang approach doesn't work
- Evo = pragmatic blend between ,extreme programming` and static project planning
 - Longer term planning as a backbone
 - Weekly milestones, weekly deadlines
 - Limits scope and increases control
 - More in line with human capacity
- Most tools on market still assume top-down approach only

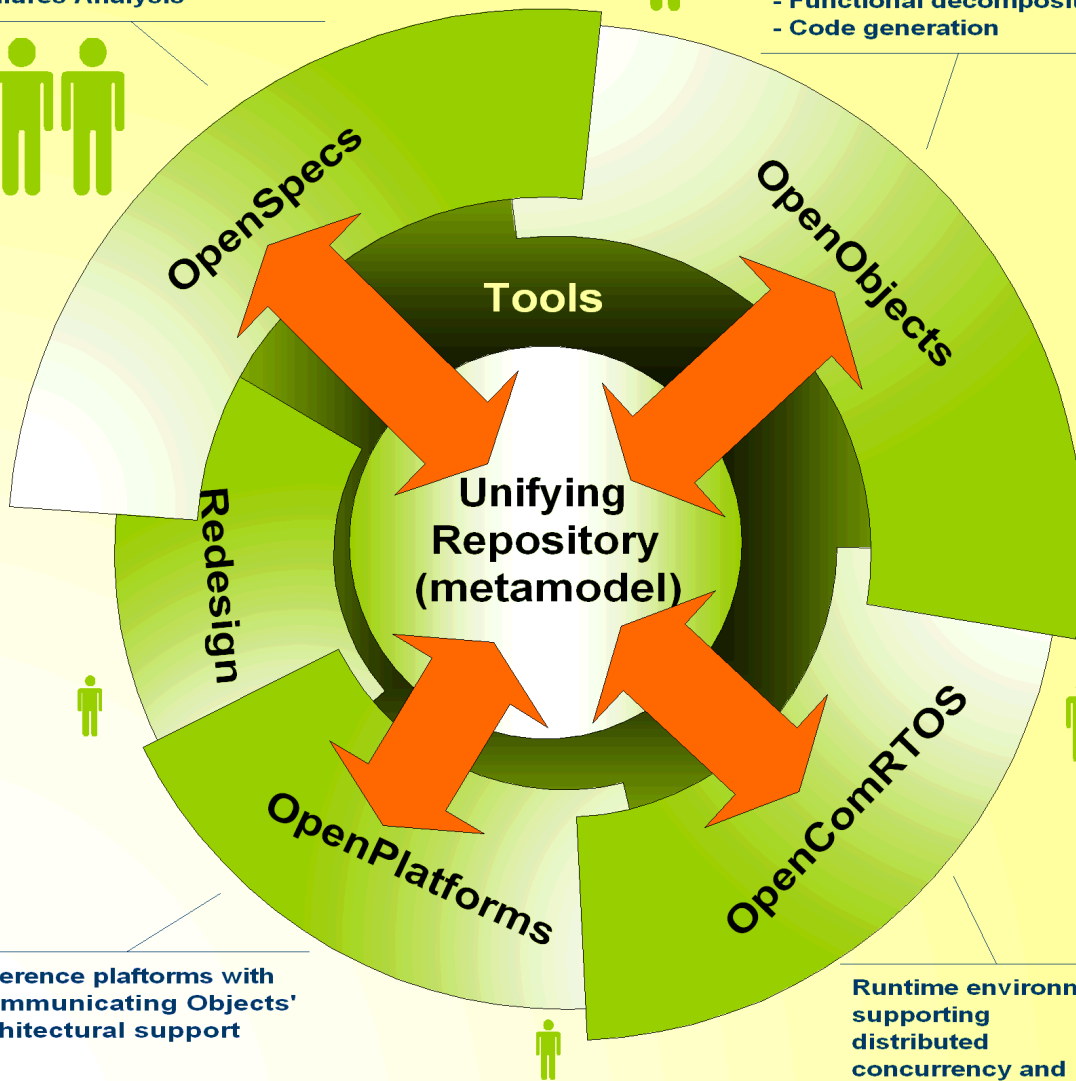
Requirements for Evo (ideally)

- All supporting tools must be integrated
 - Requires common ,semantics`
 - All activities from design to software to hardware must be compatible
 - Instant notification of any changes to all activities
- Scale of development must be consistent:
 - Modular architecture
 - Information hiding and isolation between modules
 - Formal basis
- Selected paradigm:
 - ,Communicating Objects`: universal
 - CSP as formal foundation
 - Concurrent programming model

- Requirements capturing
- Specification capturing
- Test cases
- Failures Analysis



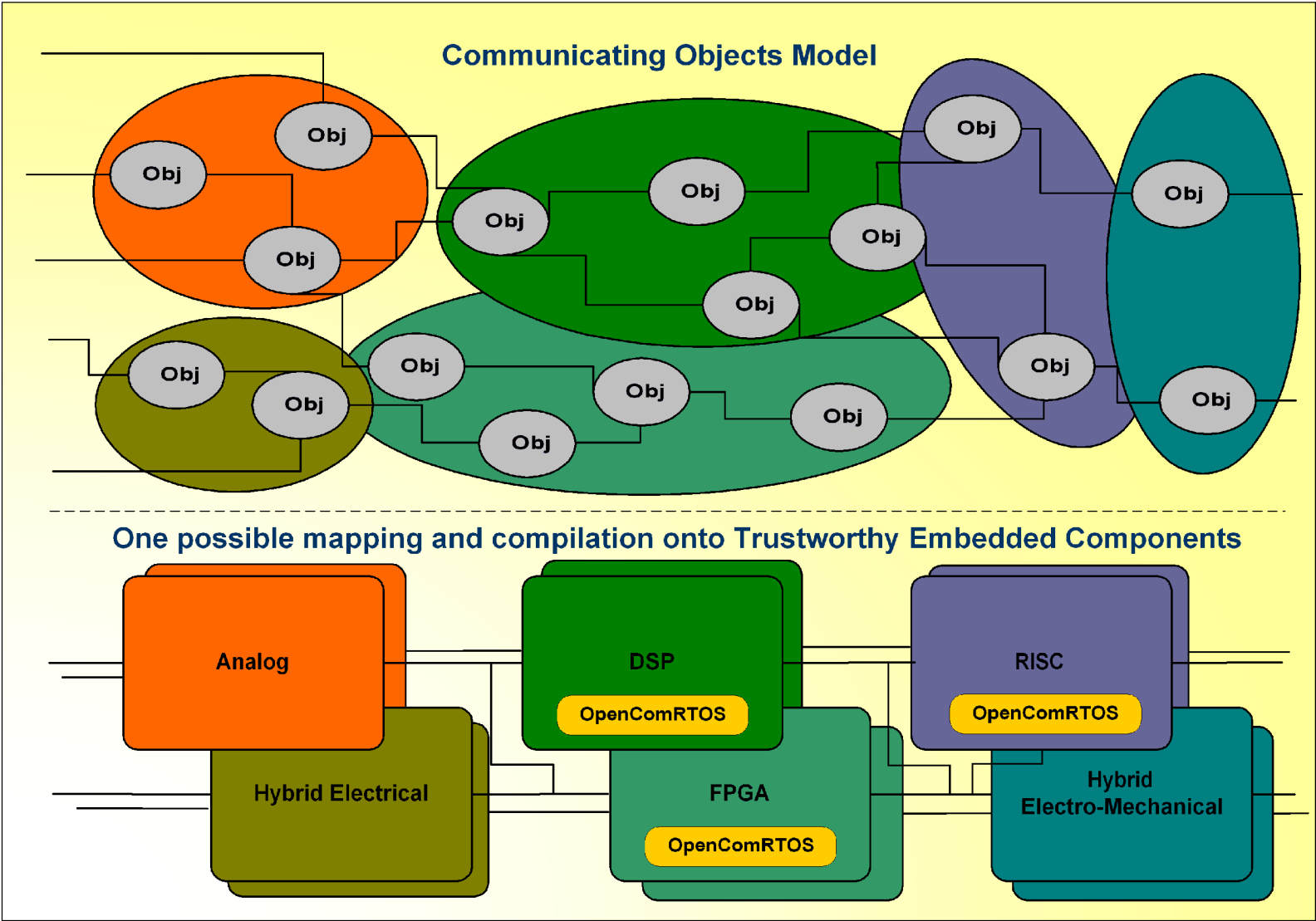
- Formal model building
- Logic simulation model
- Functional composition
- Functional decomposition
- Code generation



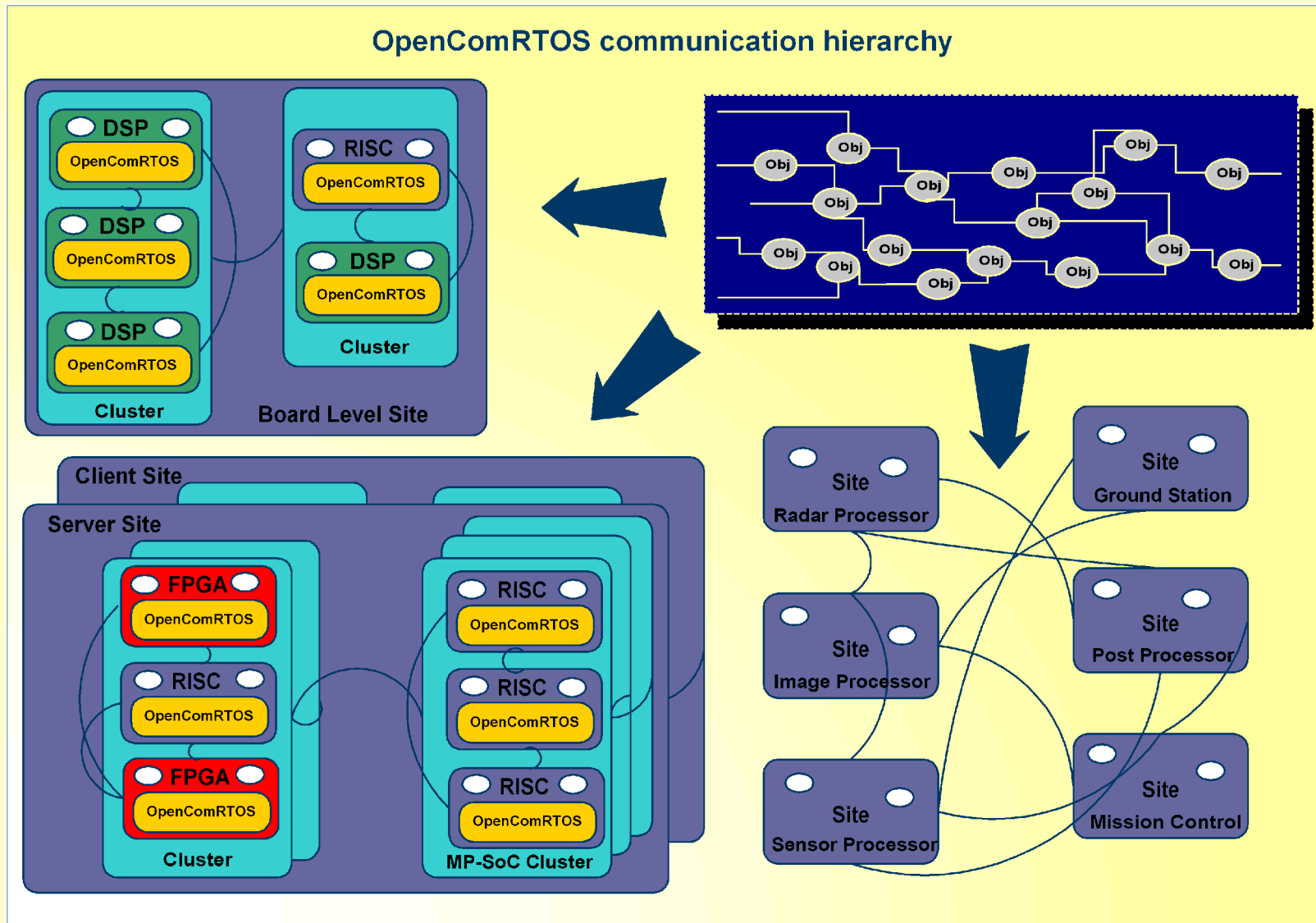
Open License Project: the unified view



Unifying paradigm (1): Communicating Objects



Unifying paradigm (2): Scalable Communication



OpenSpecs

- Requirements and Specification building
 - Prototype derived from TOFS (Romet AB, Sweden)
 - Being redeveloped as web-based tool (team-aware)
 - Improved approach:
 - Systems engineering project with different ,views`
 - Attributes, dependencies, ...
 - Project schedule (work packages)
 - Functional (de)composition
 - Test cases, Fault analysis
 - Automatic document generation
 - Less a product, more of an enabler

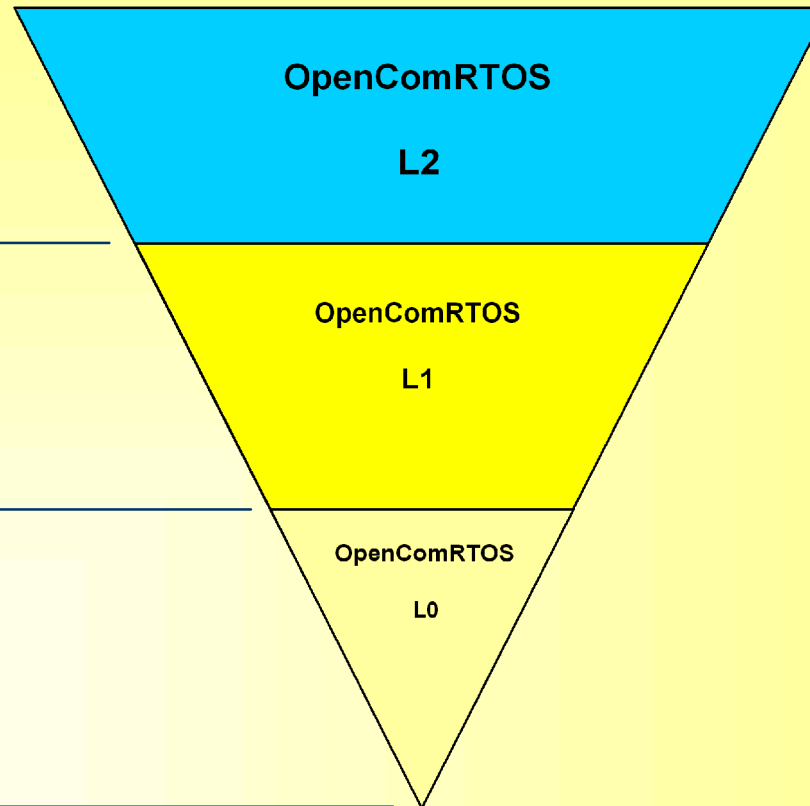


OpenComRTOS

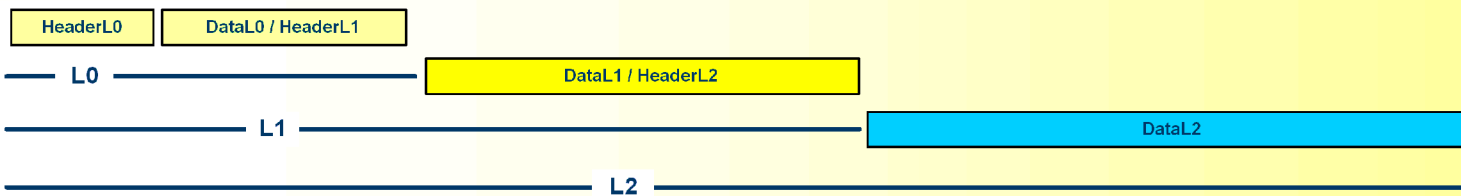
- variable size packets
- widely distributed addressing
- dynamic protocol packets
- extensible API

- fixed size packets
- cluster addressing
- dynamic protocol packets
- API emulation

- fixed size packets
- tightly clustered addressing
- static protocol packets
- system packets
- scheduler
- routing and buffering
- runtime monitor



Packet structure



Applicable now?

- ,Looking ahead` unifying solution
- Software not to be seen as ,add-on` but is growing component of any product
- Hardware-software co-design is more than using SystemC, it`s a process paradigm
- Essence is better ,formalised thinking`
- Benefits is higher quality, less life-cycle costs, higher added-value and margins
- Hence, can be applied incrementally
- Better specifications: easier to develop in a decentralised way
- A goal to be aimed for incrementally

Starting projects for methodology

- Specific projects:
 - A new product development
 - How can formal methods be applied?
 - Alternative paths to bottom-up approach
 - ...
- Roadmaps:
 - Next generation microcontrollers + software toolchain + reference platforms
 - Wireless distributed sensor networks
 - Scalability and fault-tolerance ,designed-in`
 - Application level products
 - ...

Summary

- Open License Society's approach is about ,formalised thinking`
- The essence is the process, not the tools
- The benefits are ,things being done better`

- Project approach is best start:
 - Workshops
 - New product development
 - Roadmap

- Contact:
info.request@OpenLicenseSociety.org