Application-driven Embedded System Design

Prof. Antônio Augusto Fröhlich
UFSC/LISHA
guto@lisha.ufsc.br
http://epos.lisha.ufsc.br/

Jul 2, 2010

Overview

- Embedded System
- Embedded System-on-a-Chip
- Established Embedded System Design Methods
- Application-driven Embedded System Design
- EPOS
- Final remarks
- Case studies and tales

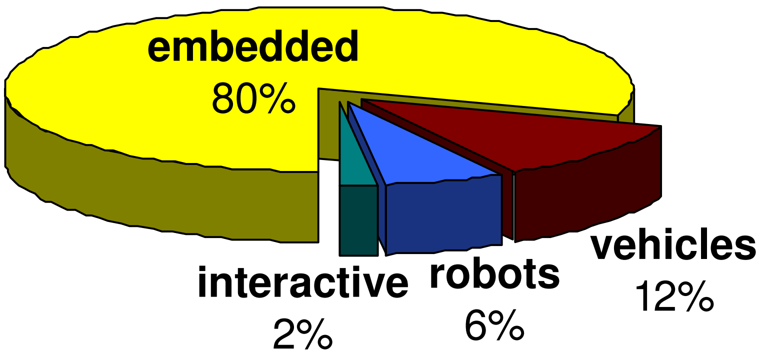
Embedded Systems: embedded!

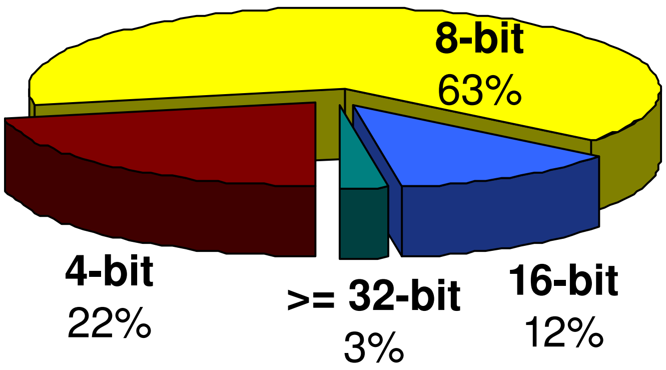
“Hardware and software which forms a component of some larger system and which is expected to function without human intervention.”

[Foldoc]

Really Embedded!

Where are the processors?
(Tennenhouse, CACM 43(5):44)



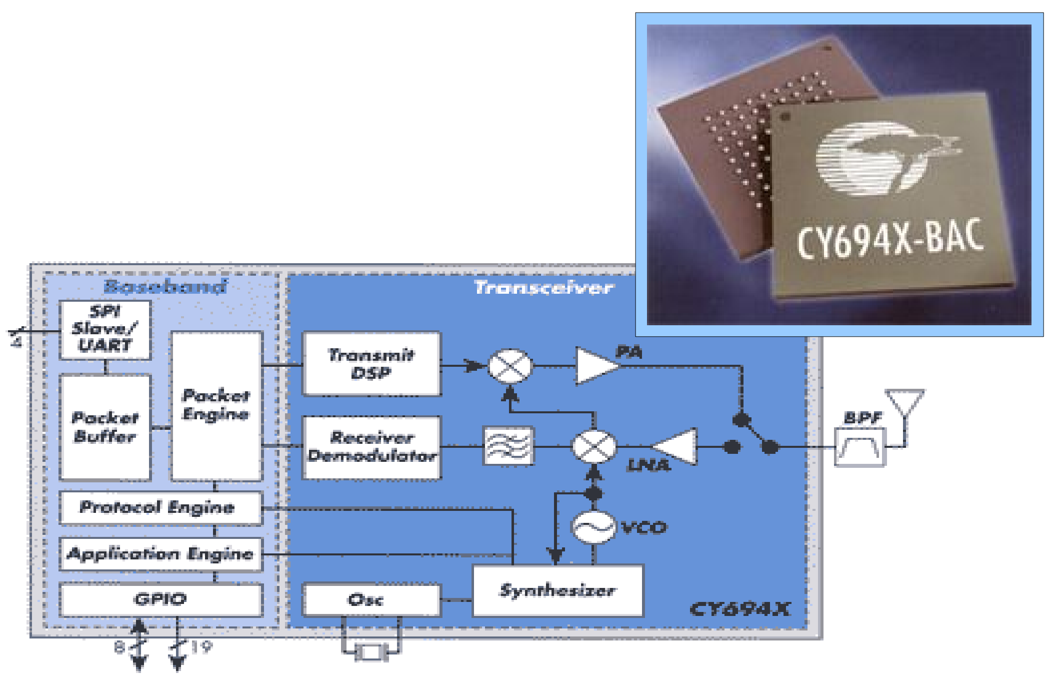


Embedded X All-purpose

- Embedded
  - Dedicated
  - Single, previously known application
  - Small set of application-specific services and features
  - Integrated hardware and software design
  - Example
    - EPOS SoC
- All-purpose
  - Generic
  - Any, many applications
  - Comprehensive set of services and features
  - Independently designed computer, operating system, and middleware
  - Examples
    - PC + LINUX + JRE
    - IPhone + MacOS + ???

Extreme Integration

- Advances in microelectronics are enabling developers to integrate complex hardware designs in a single silicon pastille
- Embedded System-On-a-Chip



Embedded System Design

R E U S E ! ! !

Component Evolution

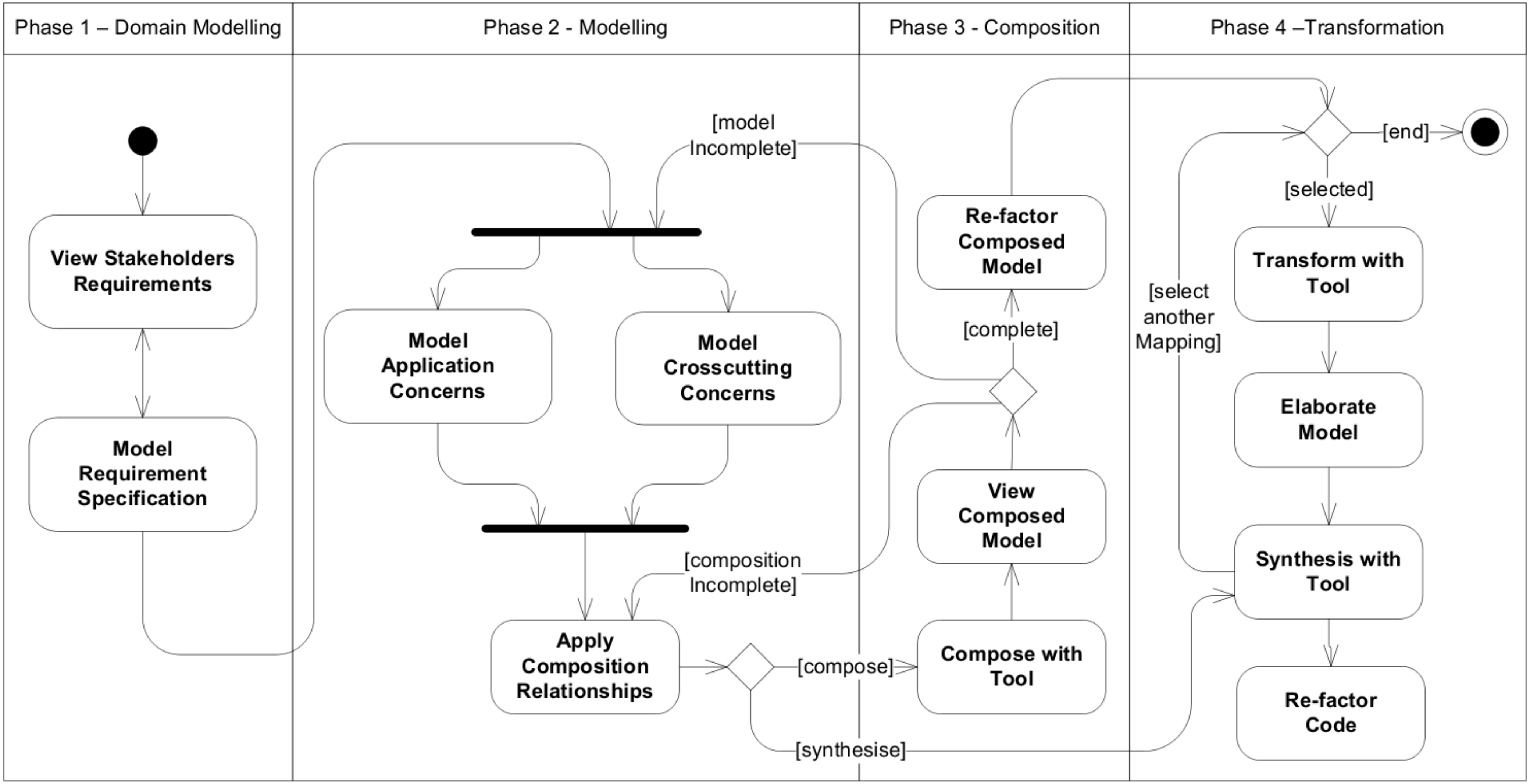
				OO++
procedures	**modules**	**classes**	**VB** **Java**	**models**
1960	1970	1980	1990	2000
circuits	**ICs**	**ISA**	**HDL IPs**	**platform**

Contemporary Design Approaches

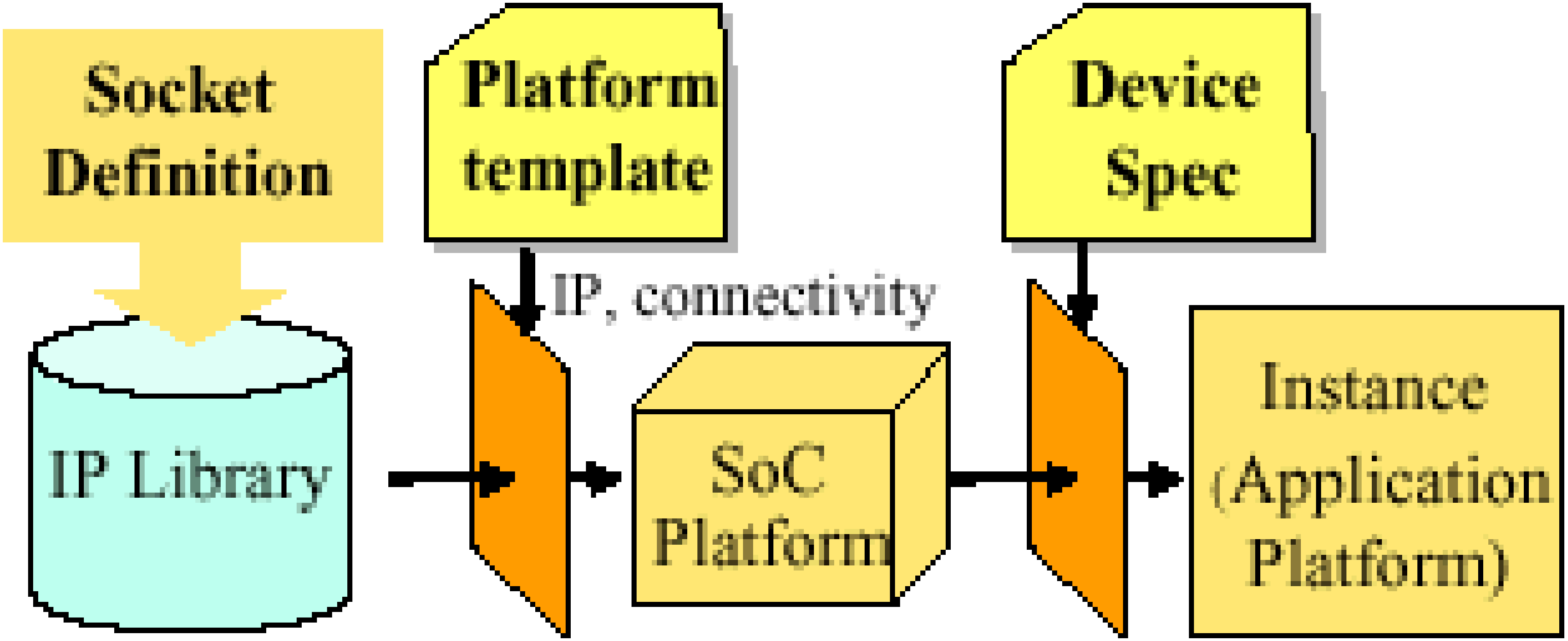
- Model-driven Engineering
  "A promising approach to address the inability of third-generation languages to alleviate the complexity of platforms and express domain concepts effectively."
  [Scmidt 2006]

- Platform-based design
  “In essence, a platform is a frozen architecture. Once the architecture is frozen, you may standardize the interfaces and give the engineers some choice of building blocks.”
  [Smith 2004]

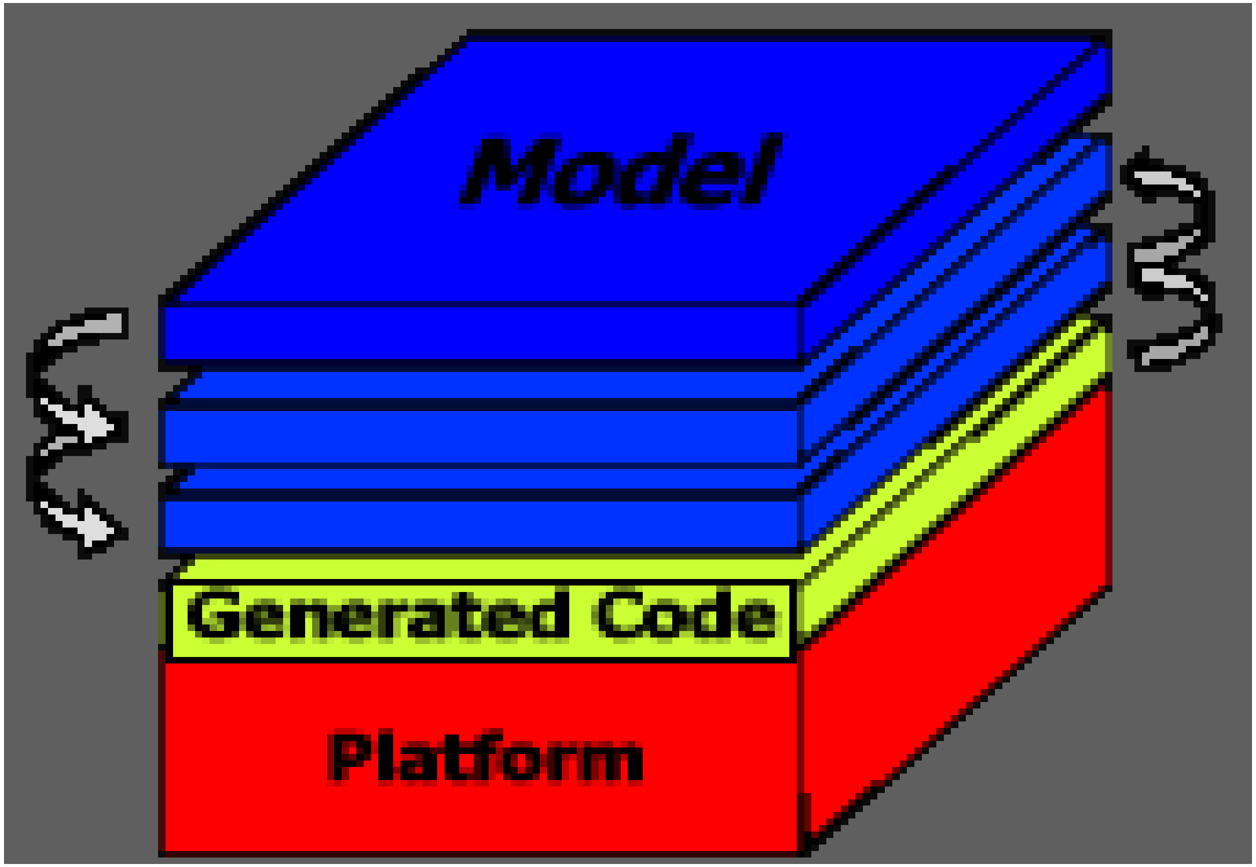
From PIM to PSM



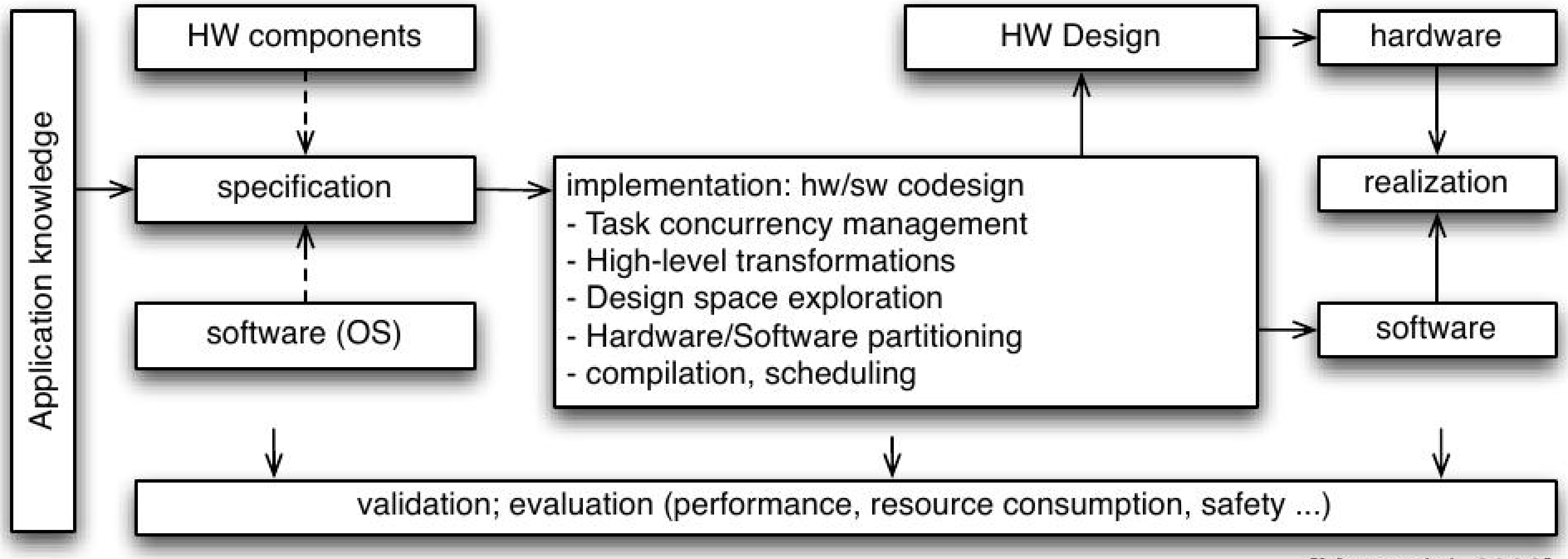
From PSM to SoC



The Magic Behind MDE + PBD



Contemporary Design Flow



[Marwedel, 2003]

Contemporary Development Tools

- Hardware
  - Focus on IP reuse and glue-logic generation
  - Run-time support is mostly considered part of application's duties
  - Examples
    - SOPC Builder from ALTERA
    - CORAL from IBM
    - EDK from XILINX
- Software
  - Focus on models, refactoring, and transformations based on middleware
  - Hardware and OS have existed since the creation of the world
  - Examples
    - UML and MDE tools
    - JAVA and PHP RTSS
    - Builders

A few words abut the OS...

- The more complex the application is, the greater is the probability it will need some sort of run-time support system
  - Core OS services (scheduling, memory management, communication, etc)
  - Peripherals abstraction (sensors, actuators, etc)
  - Power management
  - Dynamically reconfiguration
  - ...
- Ordinary operating systems cannot go with the dynamism of SoCs

and about Hardware Soft IPs

- HDLs such as VHDL, Verilog, and System C are closer to software programming language than to older hardware development strategies
- There might soon be no reason to treat them differently from software components
- Both domains can to learn from each other
  - Software can improve on handling parallelism, coordination, and timing
  - Hardware can improve on factorization, composition, and separation of concerns
- Embedded system developers could thus concentrate on what really matters: applications

The Embedded System Challenge

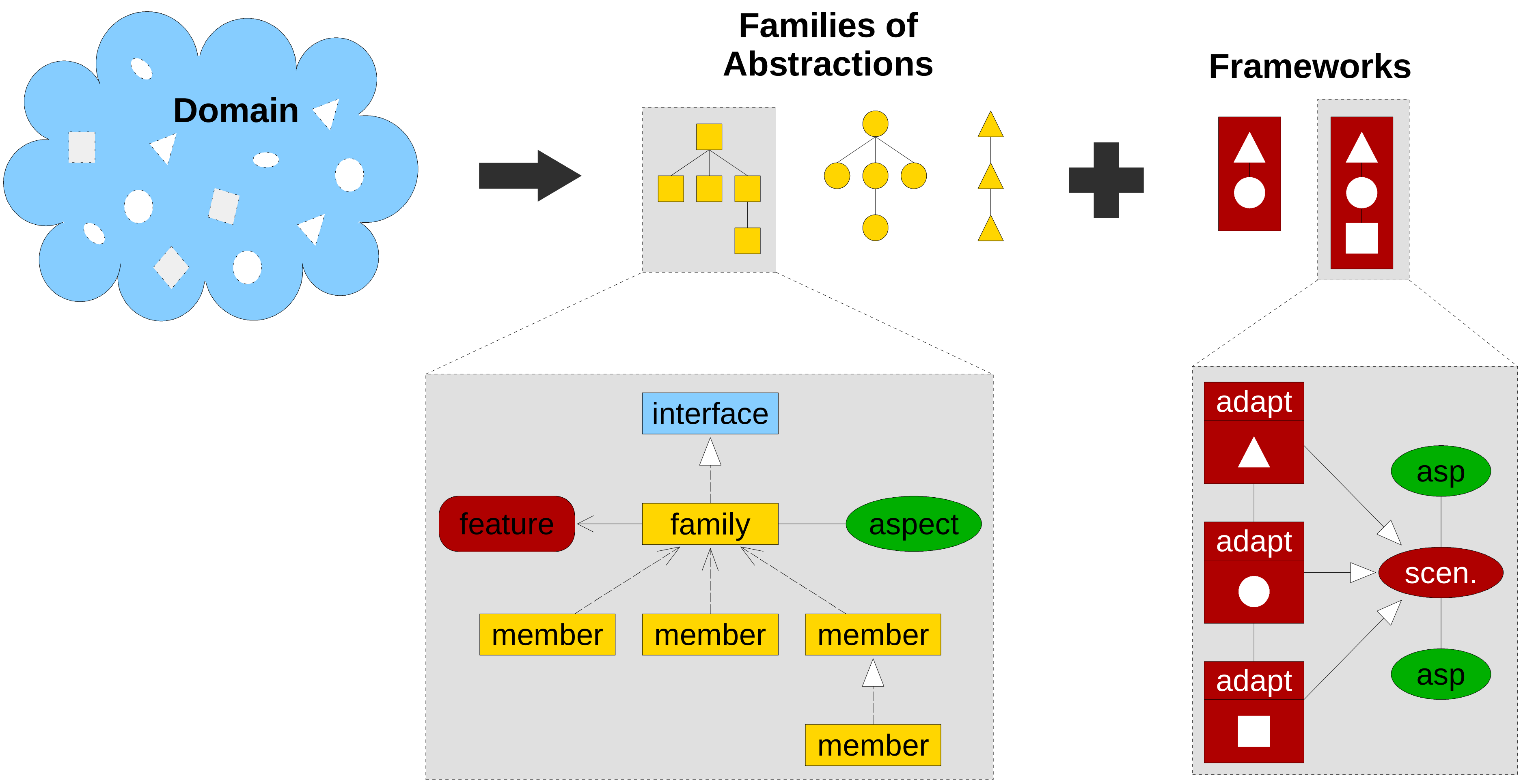
- We must give each embedded application an adequate execution platform ...
  - that properly fulfills its requirements (no workarounds, no middleware, etc)
  - that is delivered as required (application-specific API)
  - it doesn't matter what is HW and what is SW
- without having to design a new system for each application ...
- and without requiring application developers to undergo complicated configuration procedures

A Plausible Solution

- Apply domain engineering techniques
  - Family-based design
  - Object-oriented design
  - Feature-based modeling
  - Application-oriented design
  - Aspect-oriented programming
  - Generic programming
  - Static metaprogramming
  - Generative programming

  to produce embedded system components (SW / HW / hybrid) that can be (automatically) tailored according with the needs of specific applications
- A new methodology emerged
  - Application-driven Embedded System Design

Application-oriented Domain Decomposition

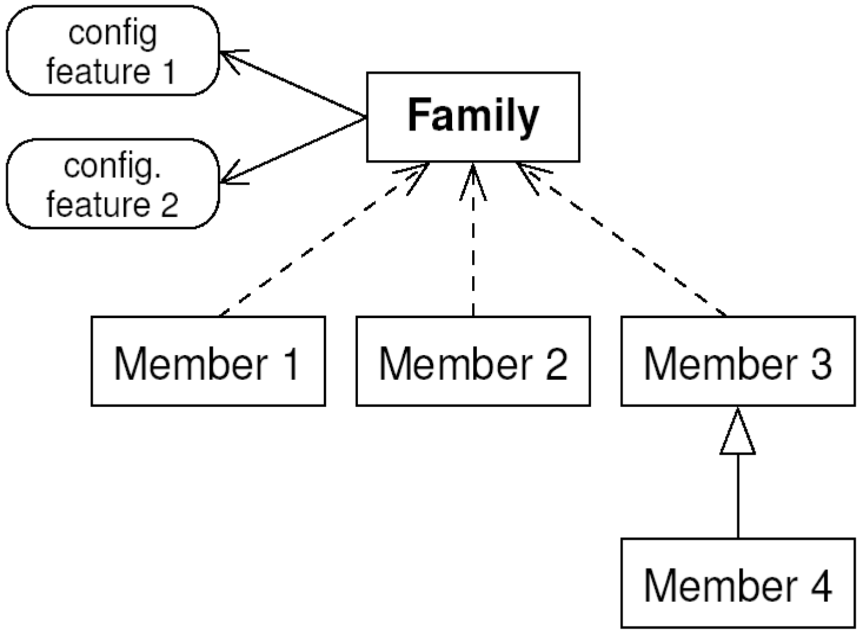


Application-oriented Domain Decomposition

- Abstractions model domain entities
- Commonality analysis
  - Build families of abstractions
- Variability analysis
  - Shape family members (subclassing or not)
  - Isolate scenario aspects
- Factorization
  - Configurable features
- Inter-family relationships
  - System-wide properties
  - Reusable architectures

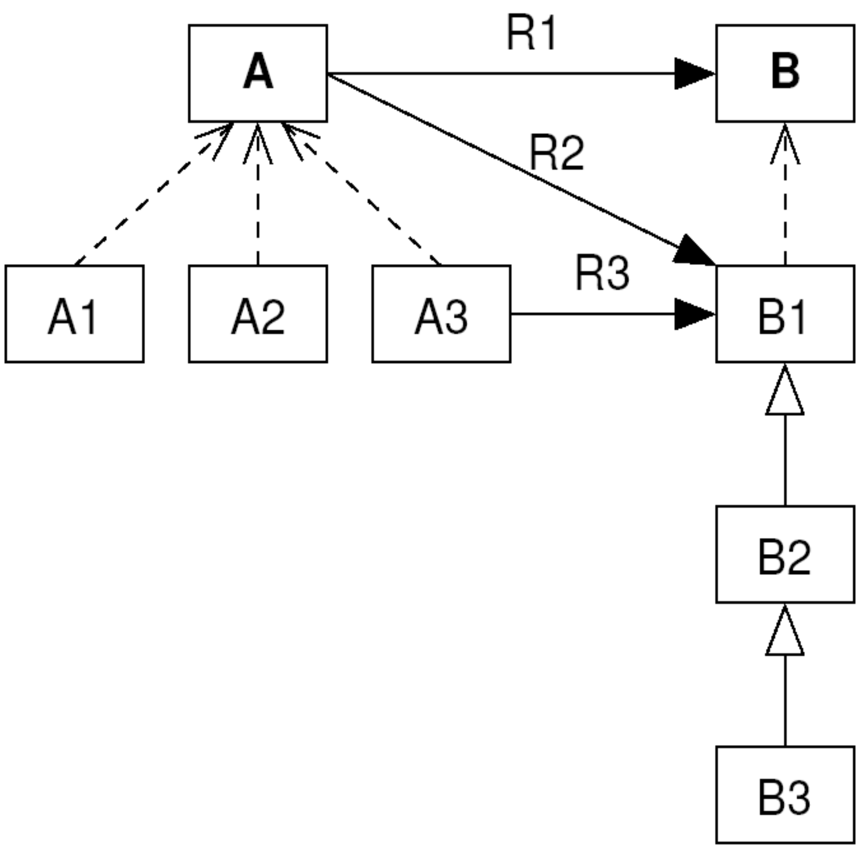
Scenario-Independent Abstractions

- Can be reused in a variety of scenarios
- Yield components
  - Application-ready ADTs
  - Correspondence with domain entities
- Families
  - Class hierarchy
  - Cooperating classes
  - Common packages
    - Base class or utility classes
    - Configurable features



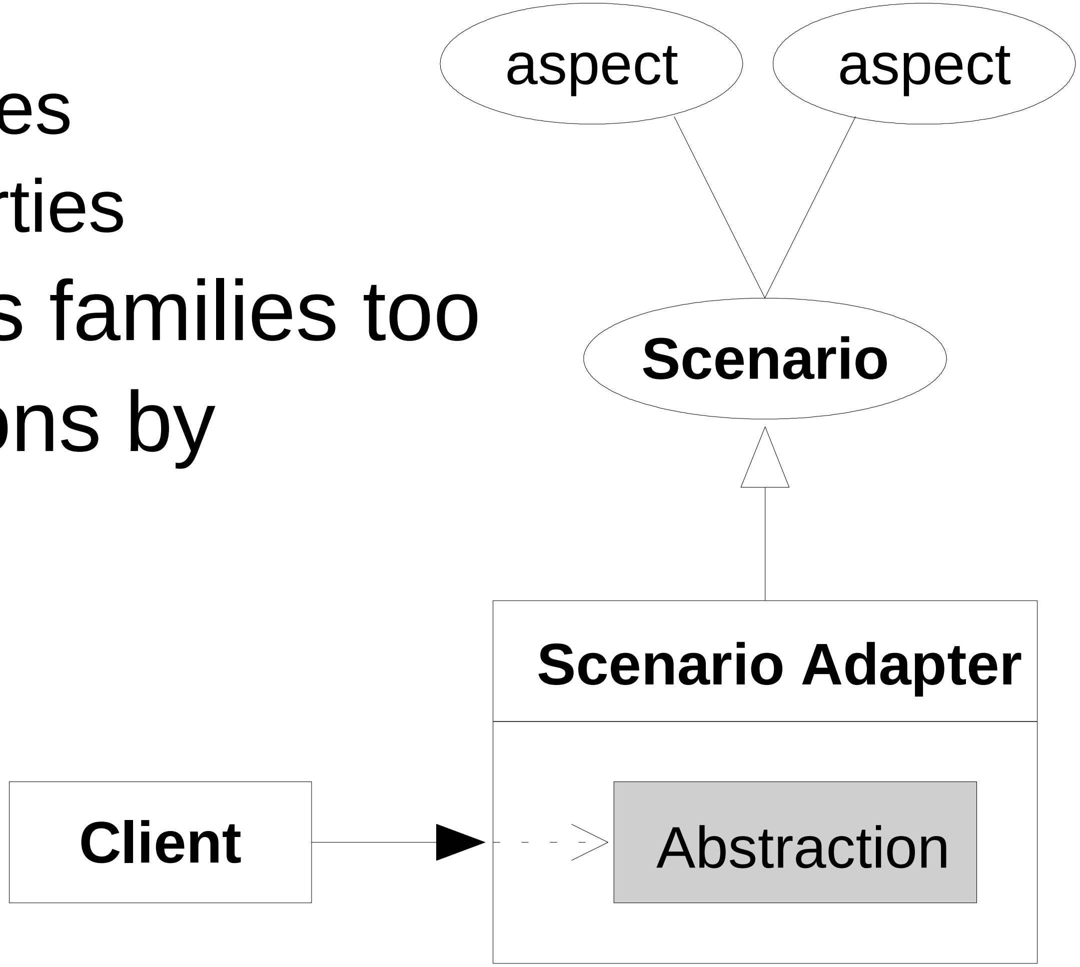
Inter-Family Relationships

- Shape framework composition rules
- Well captured with feature-based models
- Avoid
  - Restrictive rules
  - Loose rules
  - Relations for the sake of reuse
    - Factorization



Scenario Aspects

- Properties that transcend the scope of single abstractions
  - Scenario dependencies
  - Non-functional properties
- Can be organized as families too
- Applied to abstractions by
  - Weavers
  - Scenario adapters



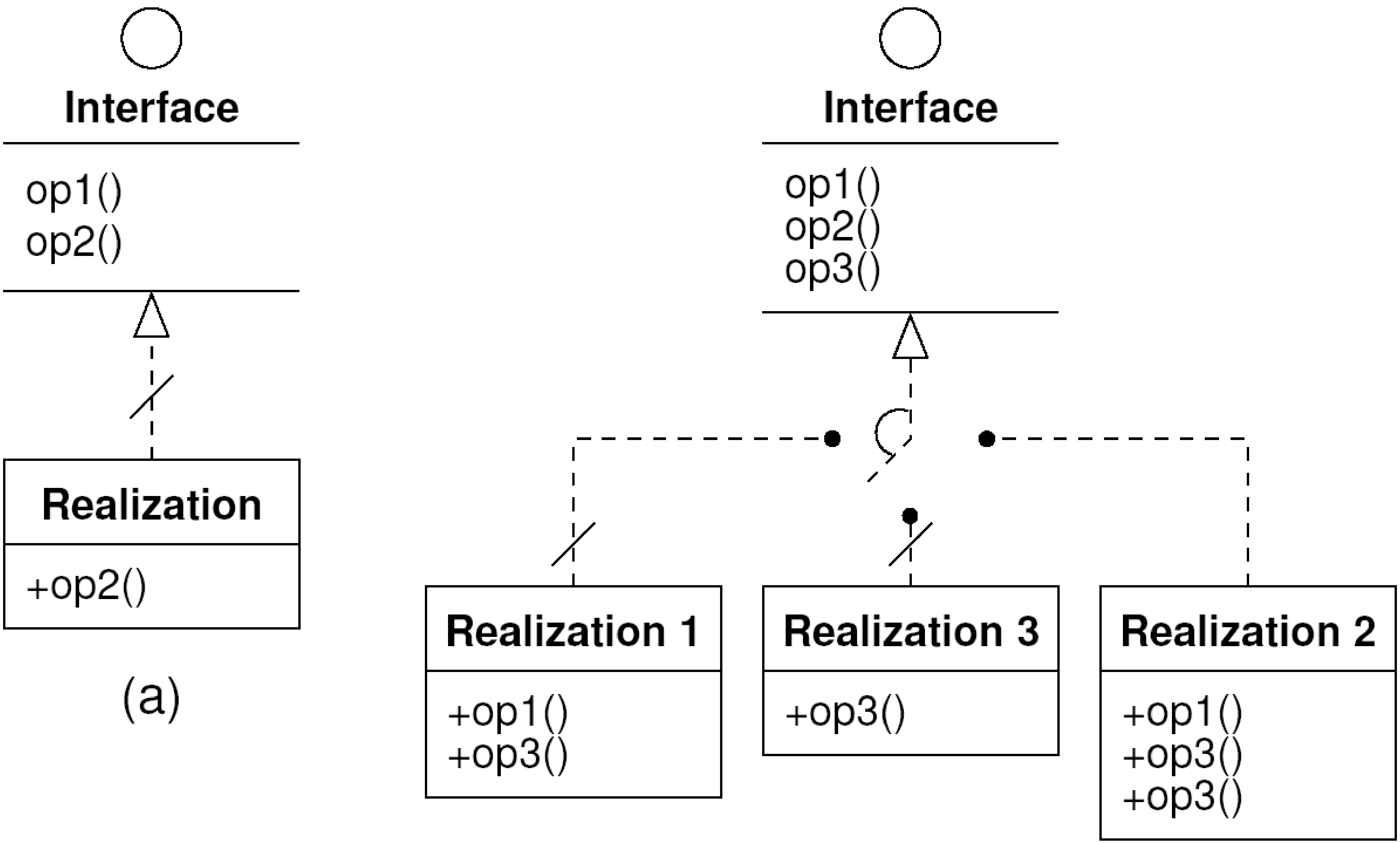
Configurable Features

- Configurable features differ from aspects in that
  - They are specific to a single family of abstractions (do not crosscut families)
  - They are not transparent to abstractions
    - but encapsulate generic programming implementations of algorithms and data structures associated to the feature that can be reused by abstractions when the feature is turned on

Inflated Interfaces

- Export families of abstractions to applications as if they were a single abstraction
  - Well-known to application programmers
  - Comprehensive
  - Promote requirement analysis
- Support automatic generation
  - Interface references can be extracted from specifications and trigger the search for adequate components
- Rely on feature models

Partial and Selective Realization



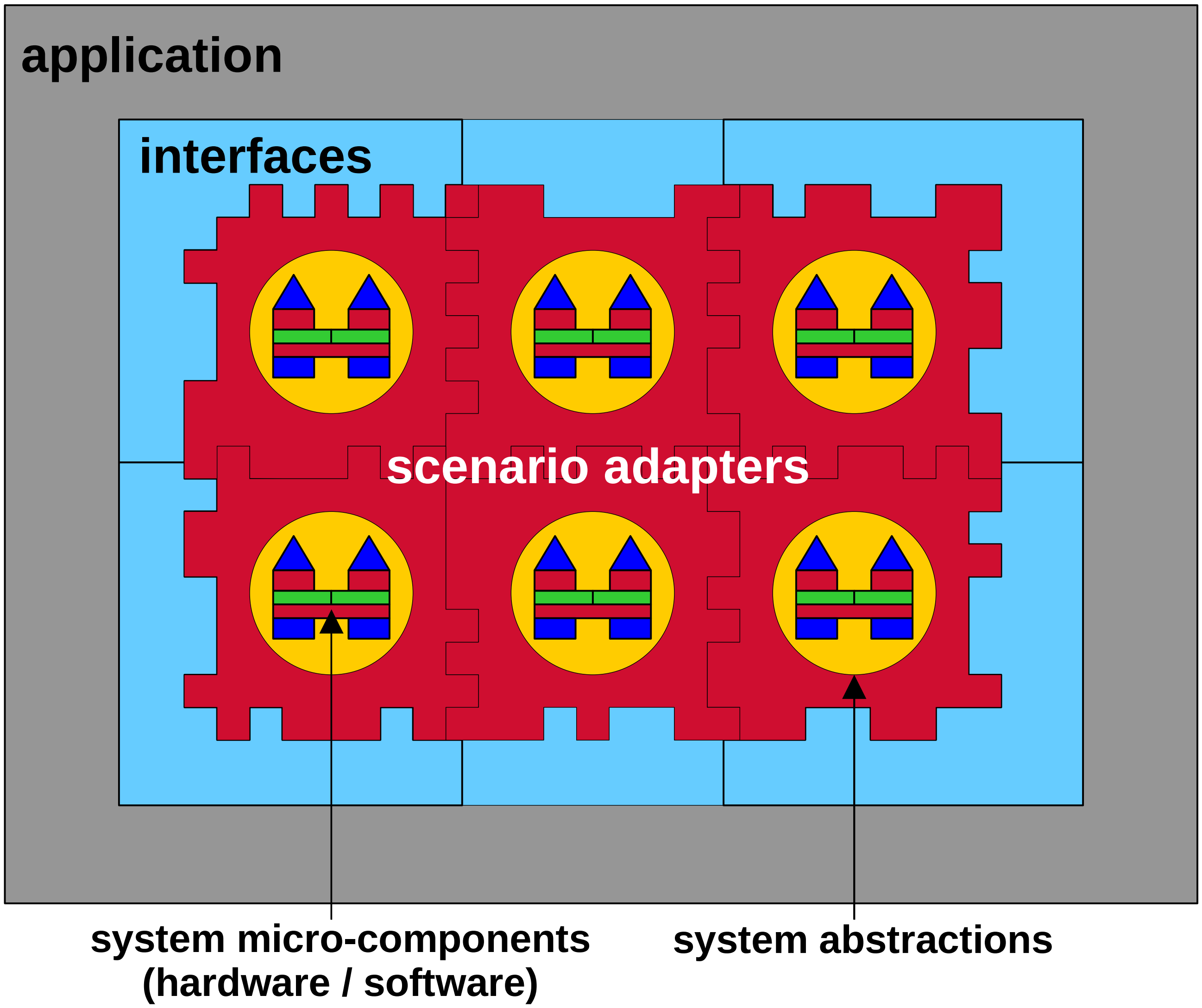
(a)

(b)

Component Frameworks

- Also known as “black-box frameworks”
  - Based on the idea of components and defined interfaces (in opposition to inheritance and overriding used in white-box frameworks)
  - The reuse of a component does not imply on reusing the whole framework along with it
- Defined as compositions of scenario adapters (placeholders for components) and a configuration knowledge base that specifies components' requirements and dependencies

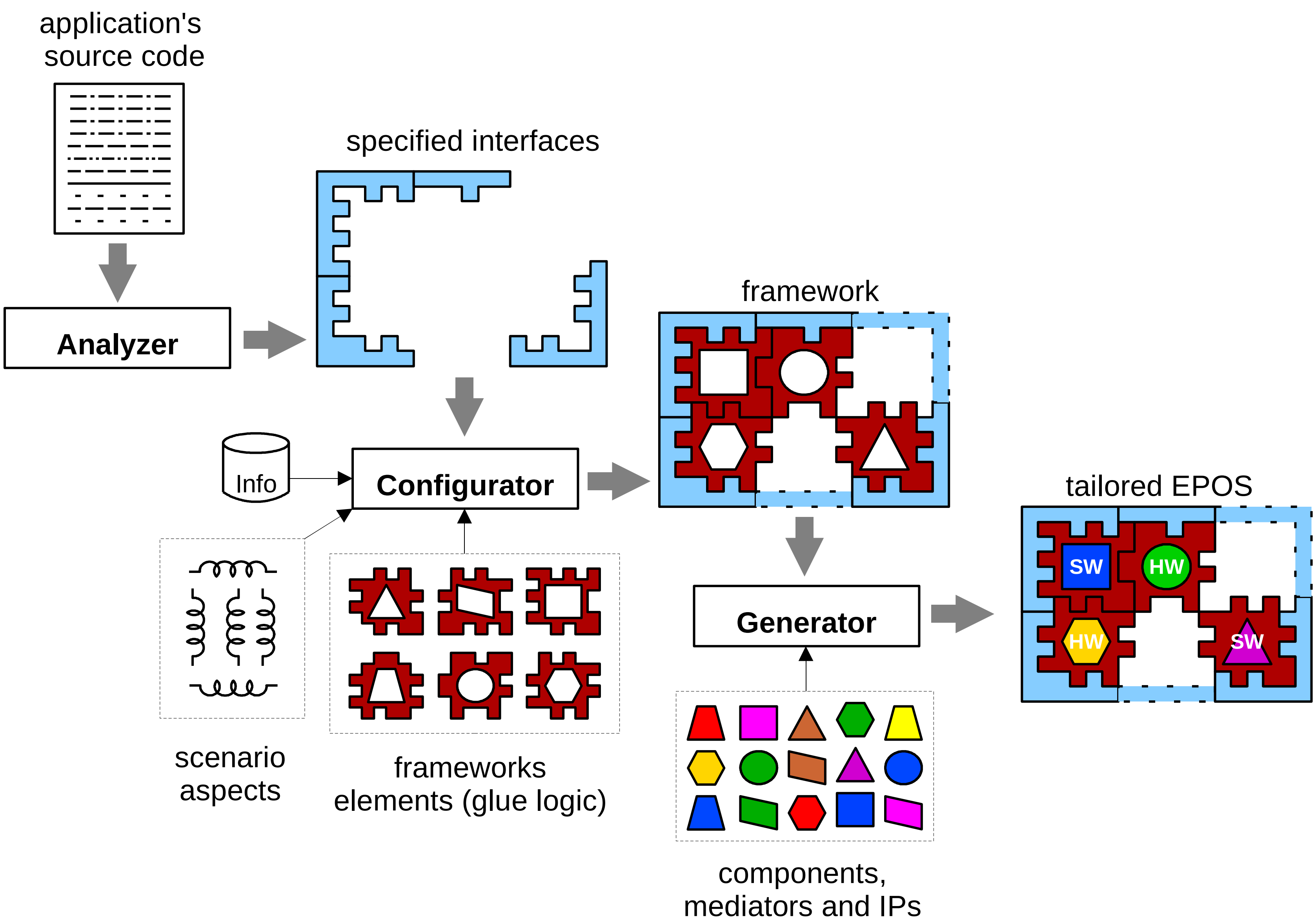
Application-oriented Embedded System



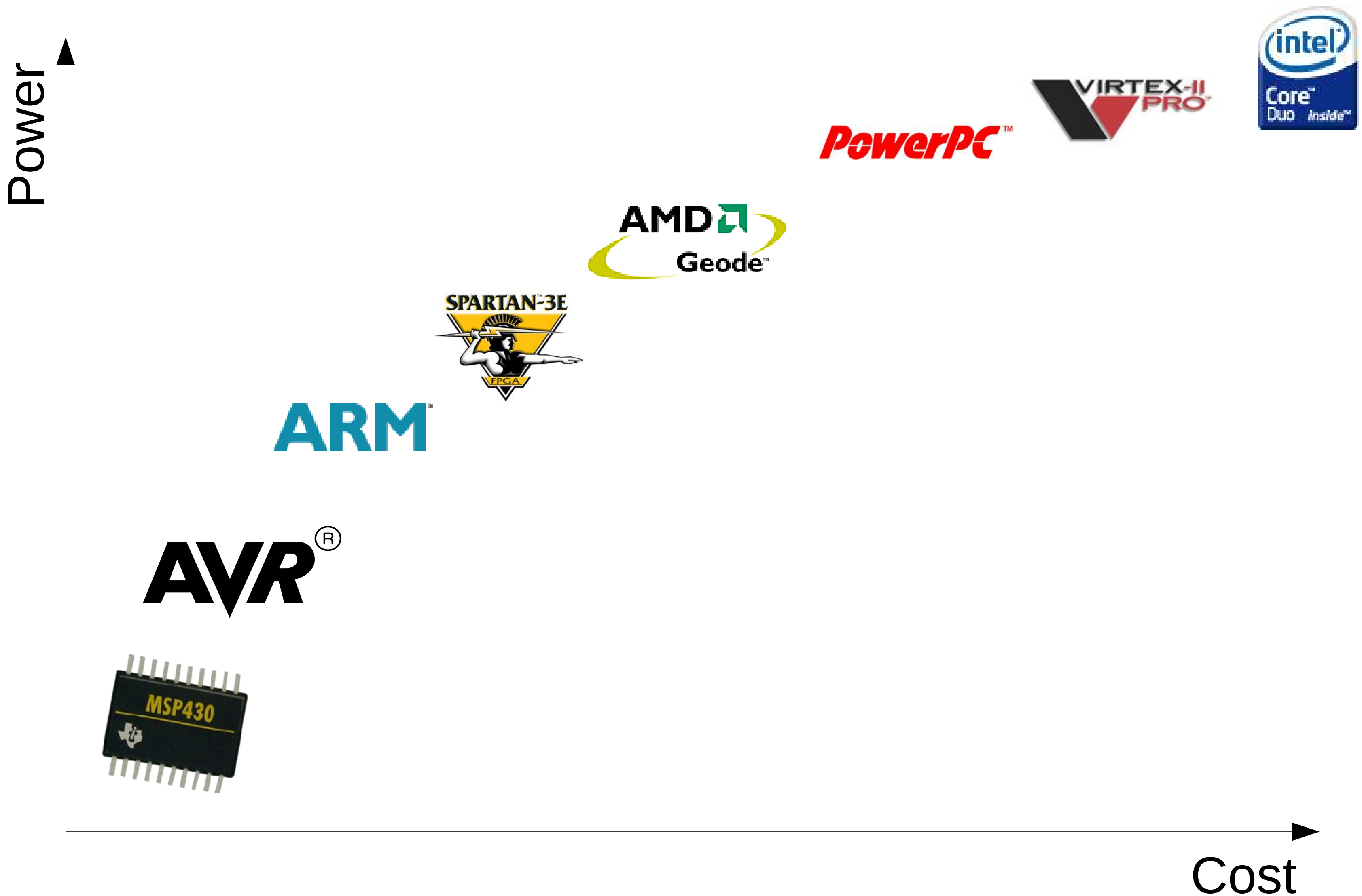
The EPOS System

- Embedded Parallel Operating System
  - A collection of SW/HW components
  - A meta-programmed framework
  - A set of tools to assist the selection, configuration and adaptation of components
- 10 years old
- 50 man/year work (10 % committed)
- Mostly academic
  - CS courses on OS and Embedded Systems
- But also industrial
  - Telecom
  - Multimedia

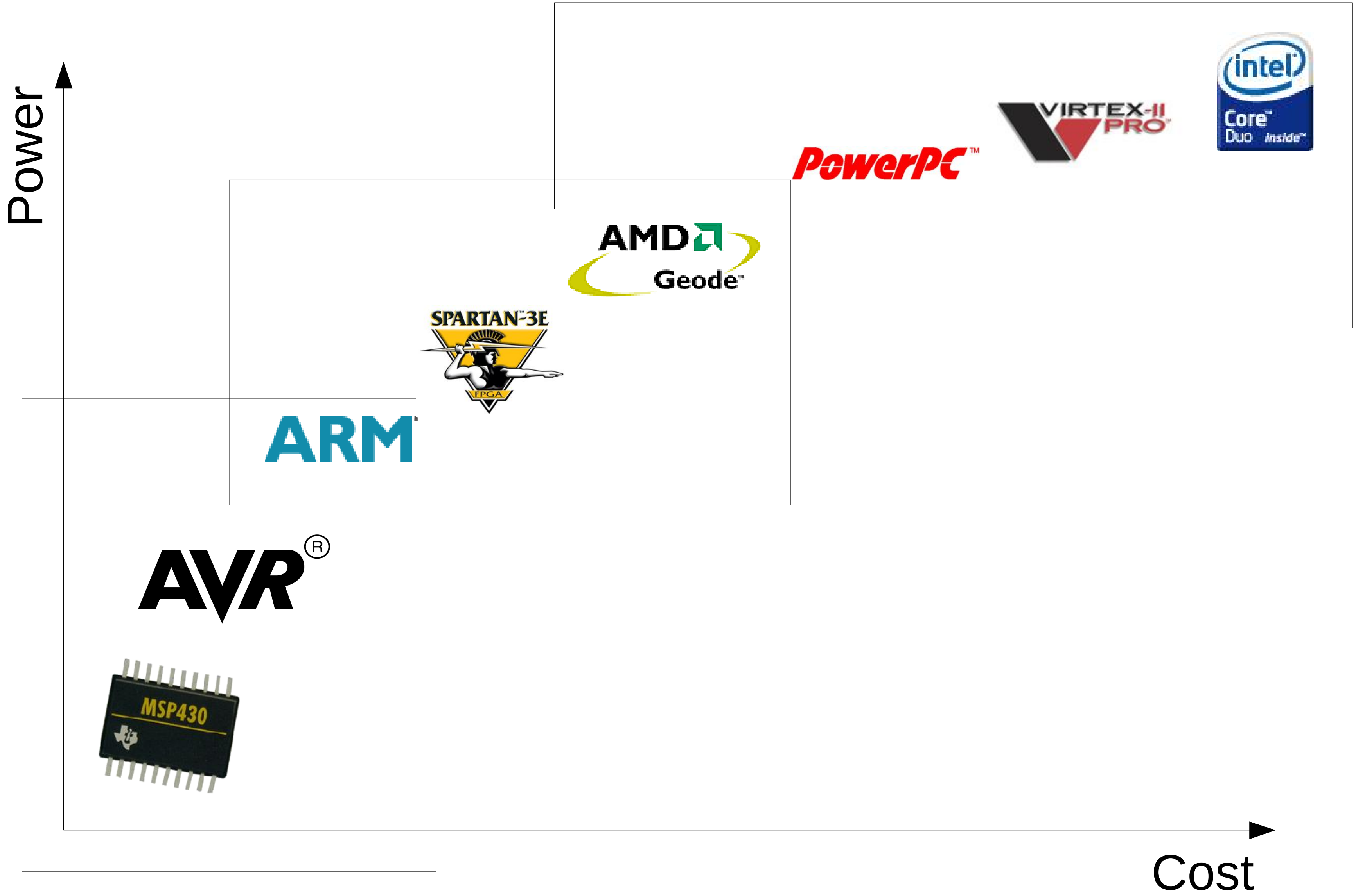
EPOS Tool Chain



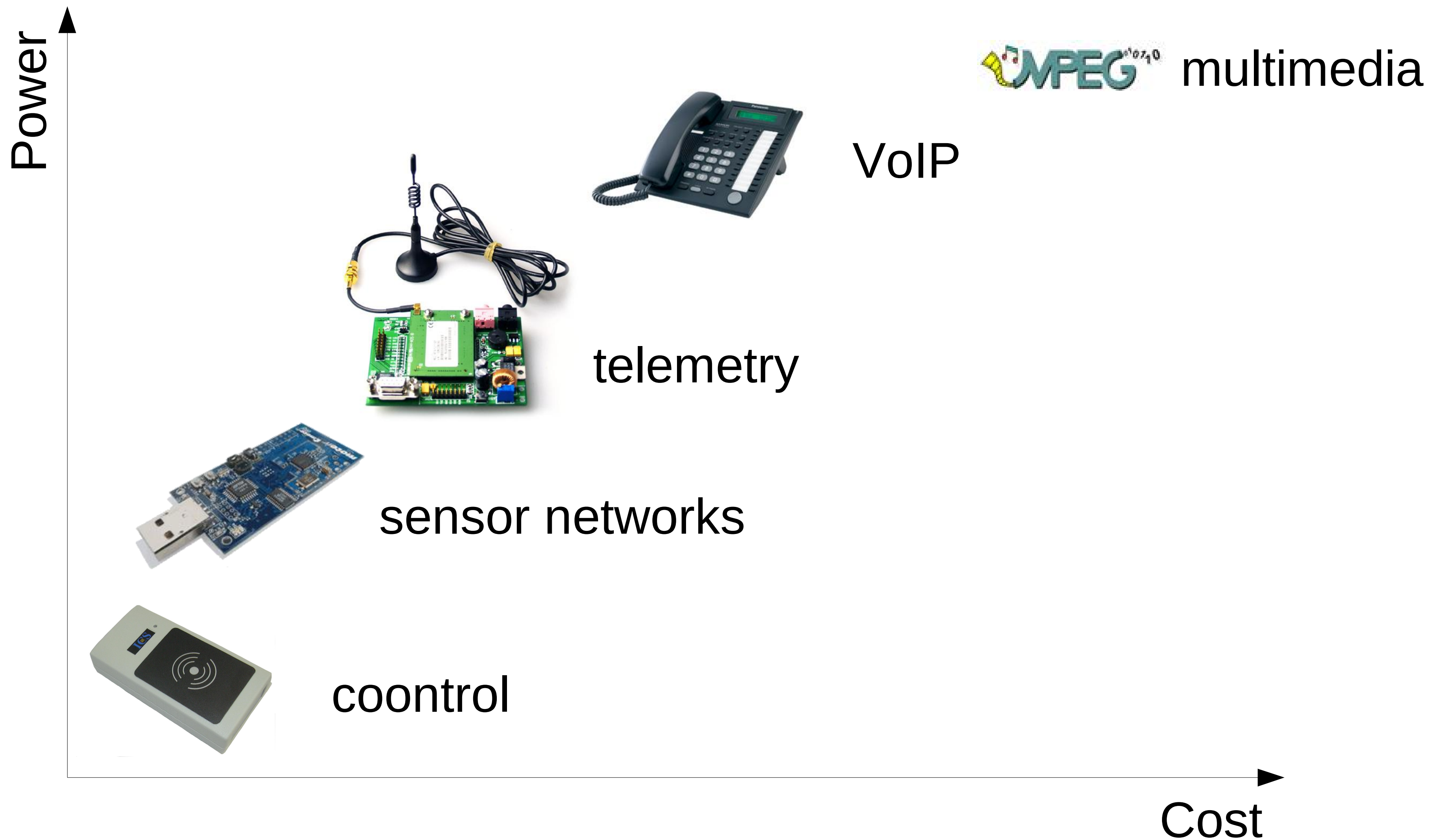
EPOS Scales ...


Power
MSP430
AVR
ARM
SPARTAN-3E
FPGA
AMD
Geode
PowerPC
VIRTEX-II PRO
intel
Core Duo inside
Cost

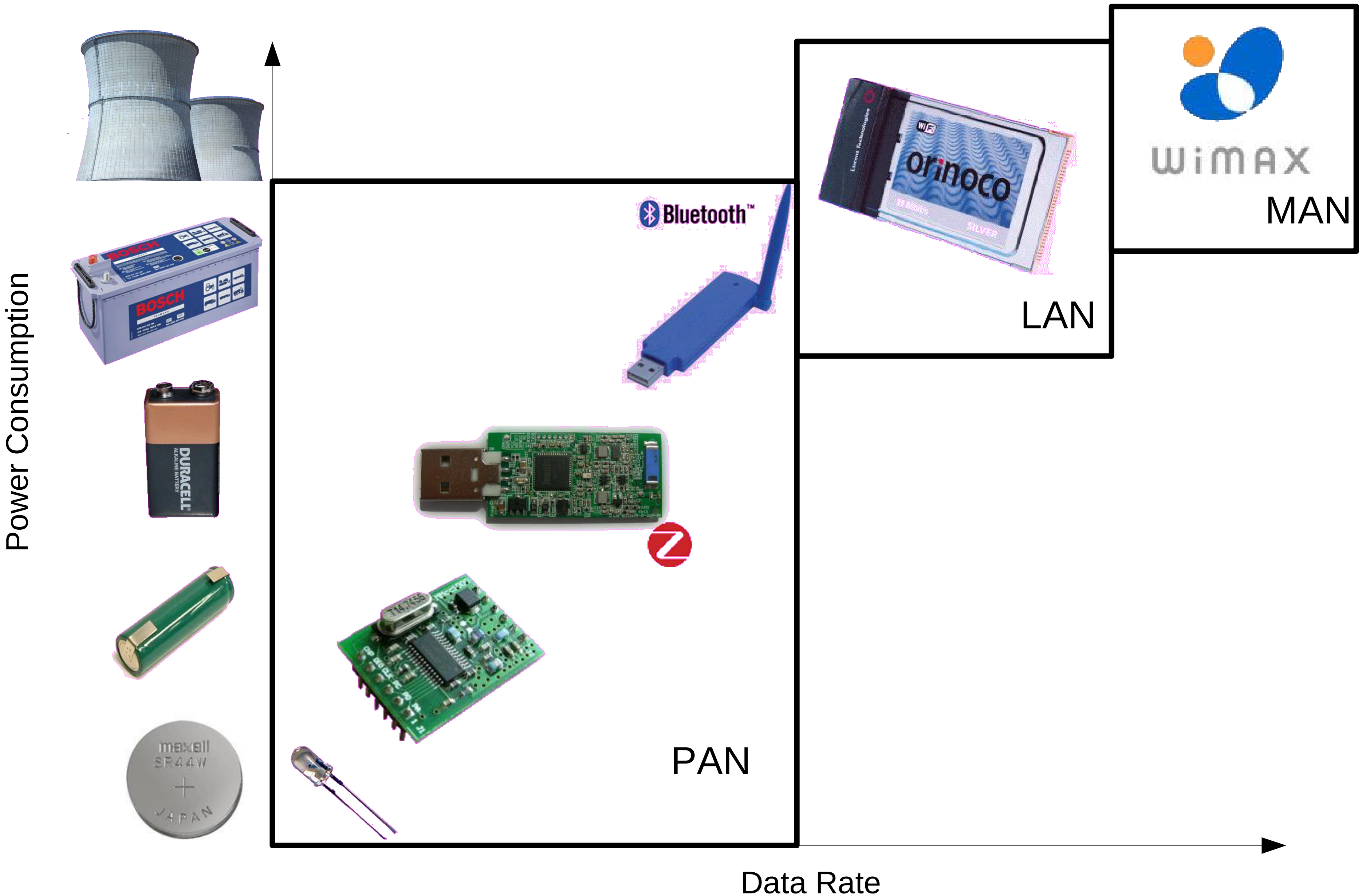
... sustaining Real Design Space Exploration ...



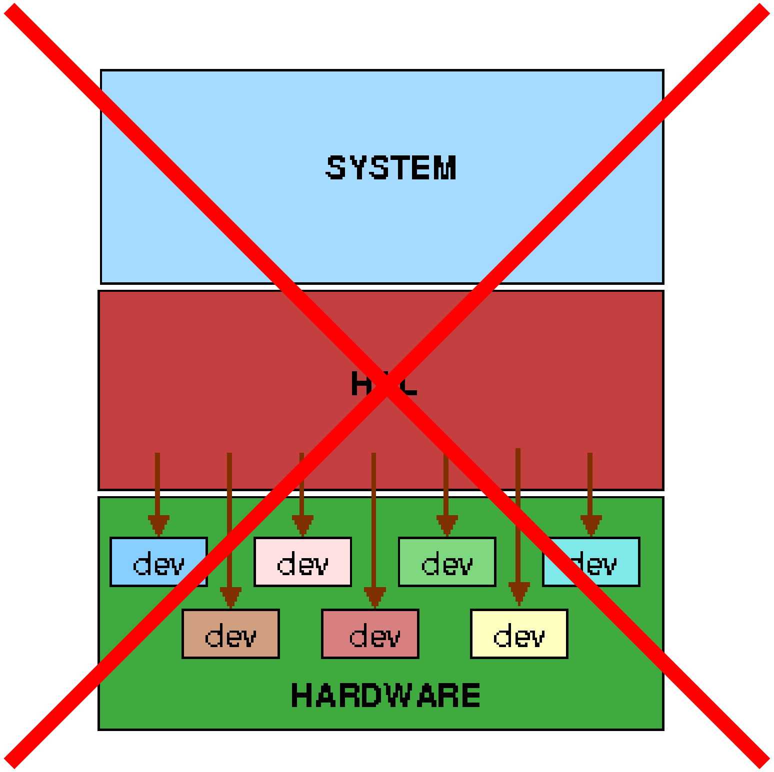
... for the sake of Applications ...



... with power efficiency

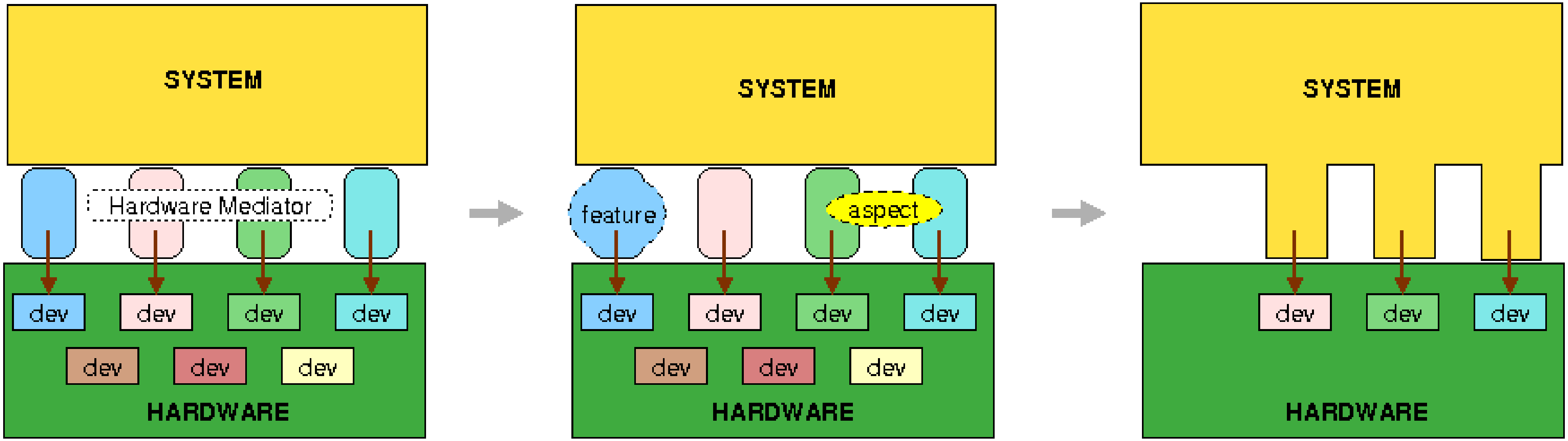


Hardware Abstraction Layers



Hardware Mediators

- Sustain an interface contract between system abstractions and the machine
- Mostly metaprogrammed



EPOS Sample Instance

- Single task
- Multiple threading
- Cooperative scheduling (co-routines)
- Dynamic memory allocation

Arch.	**.text**(bytes)	**.data**(bytes)	**.bss**(bytes)	**total**(bytes)
IA- 32	926	4	64	994
H8	644	2	22	668
PPC32	1,692	4	56	1,752

EPOS X eCos: footprint

- eCos - Embedded Cygnus Operating System
  - Customizable run-time support system by RedHat
  - Manual configuration
  - HAL-based

- Evaluated instance of eCos
  - Same configuration as EPOS

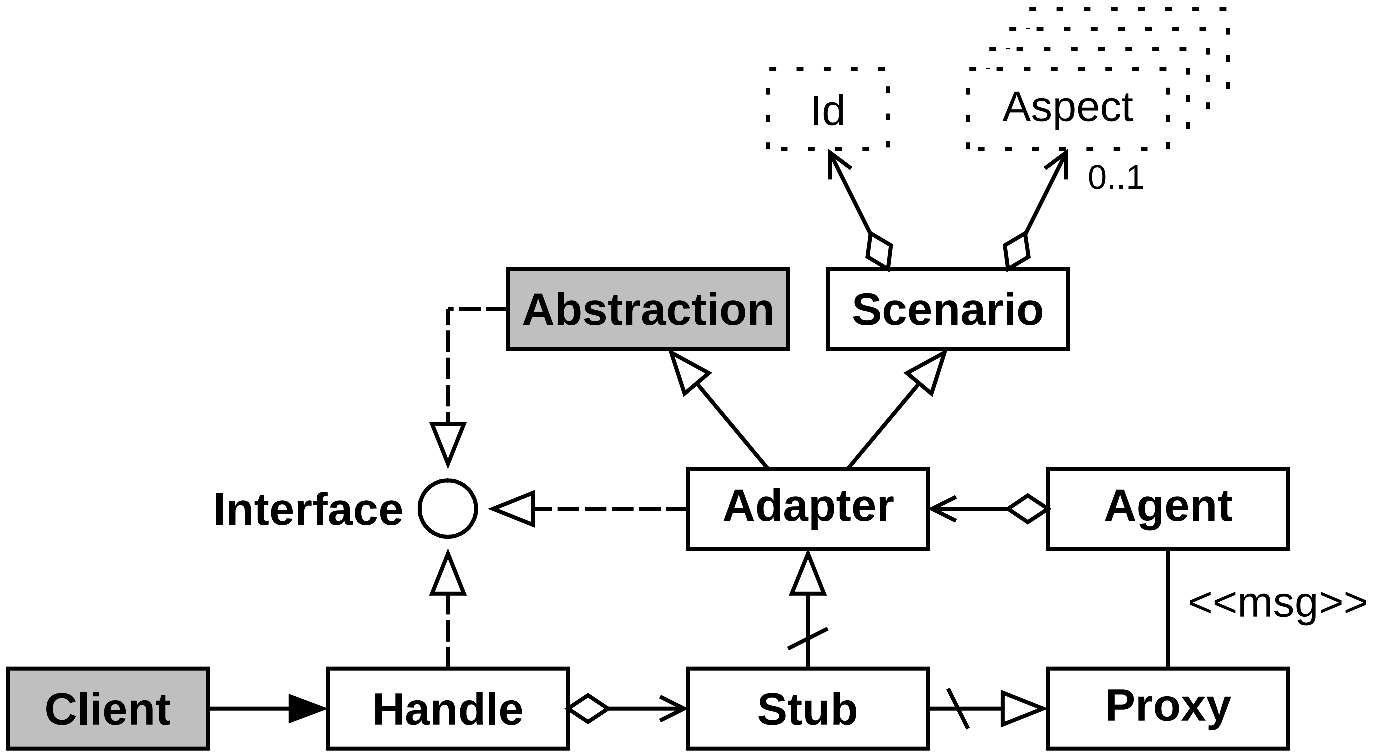
System	Portability Strategy	Size (bytes)
EPOS	Hardware Mediators	994
eCos	HAL	35,85

EPOS X eCos: performance

- IA-32-based platform
  - Time taken for a consecutive number of context-switching operations and memory allocations

System	Benchmark	Time taken (µ s)
EPOS	context-switching	1,502
eCos	context-switching	2,915
EPOS	memory allocation	762
eCos	memory allocation	3,180

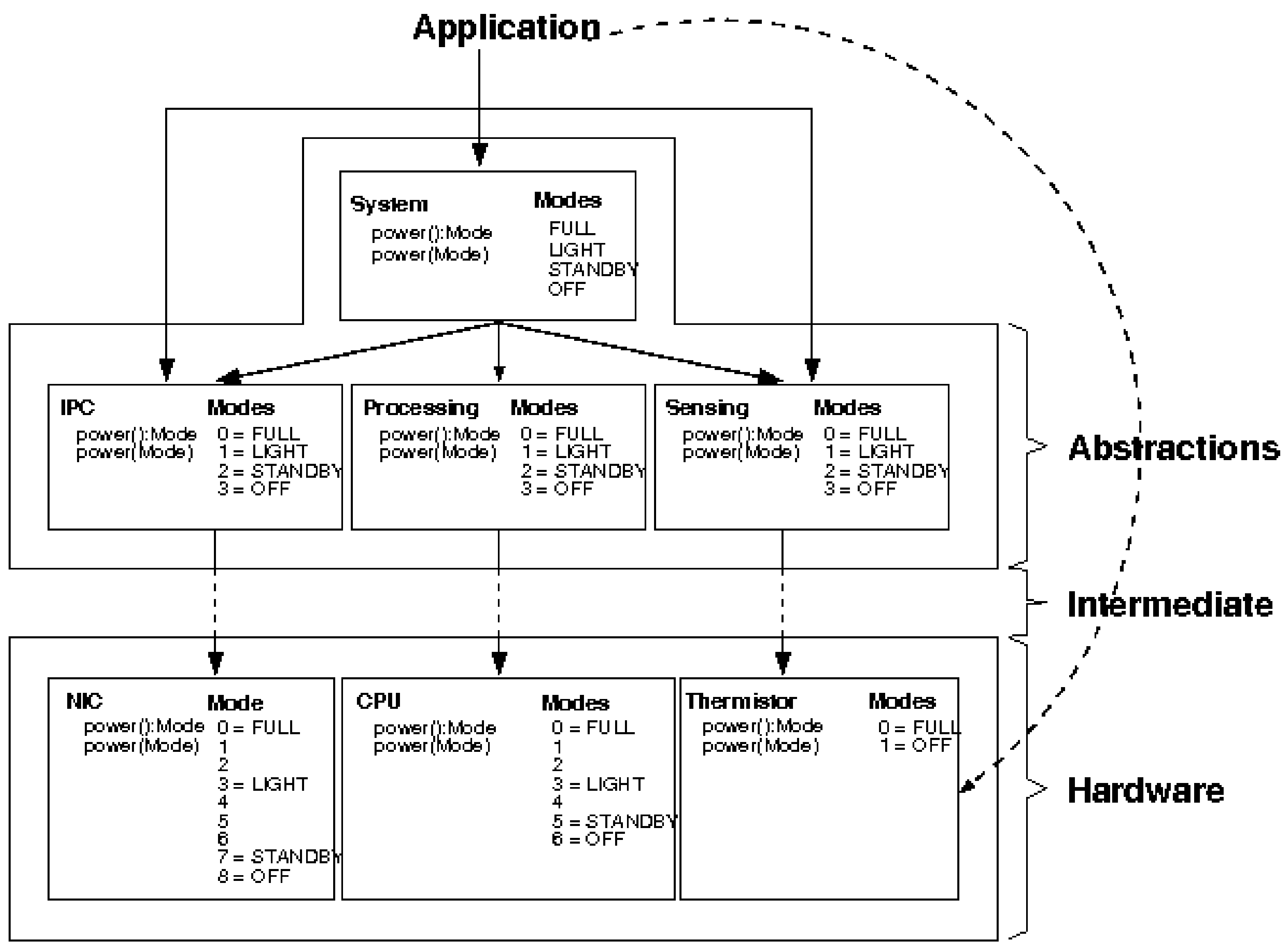
EPOS Framework Metaprogram



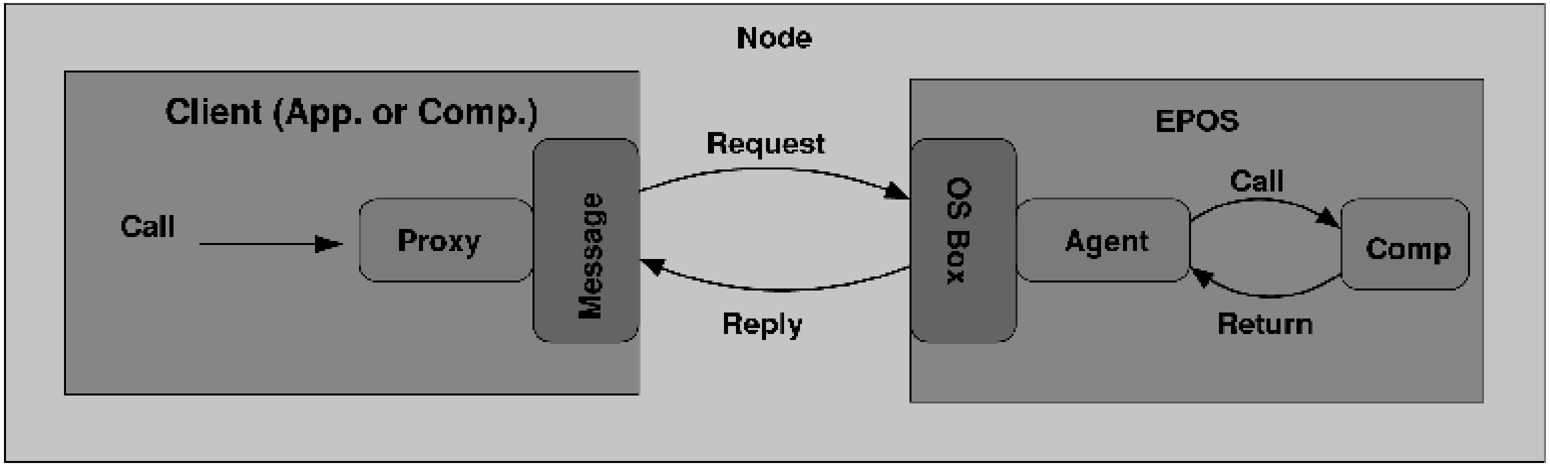
Power Management in EPOS

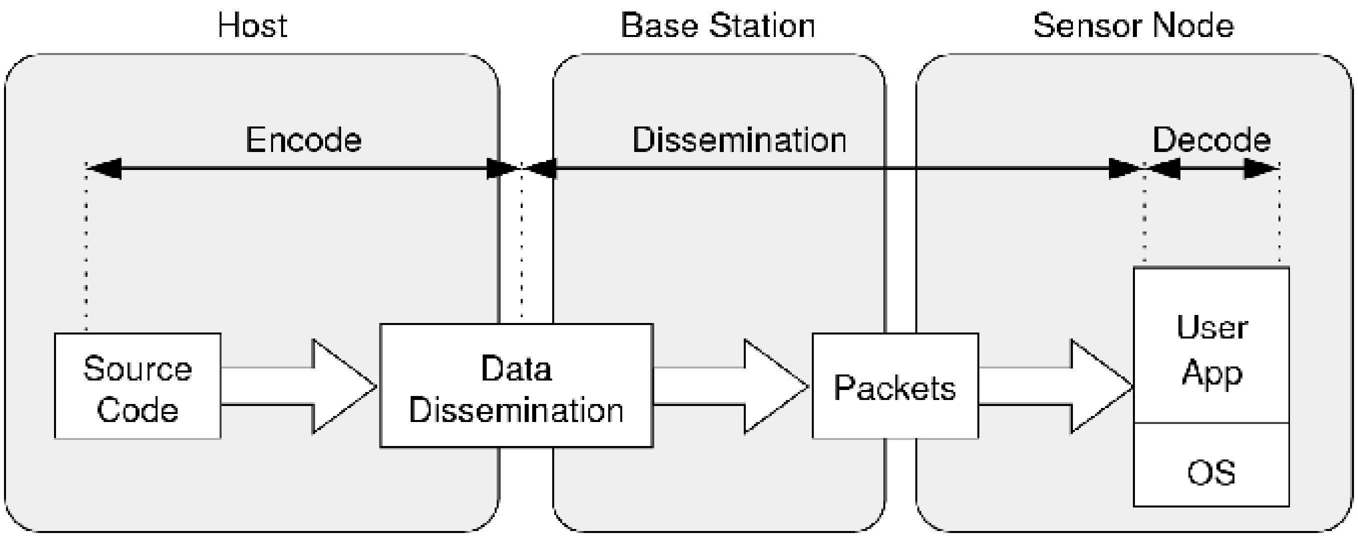
- Application-driven
- Hierarchical
  - At high-level abstractions, propagated to mediators
  - Formalized with Petry Nets
- Semantic modes
  - OFF
  - SUSPEND (hibernation and reconfiguration)
  - STAND BY (short-time resume)
  - LIGHT (fully functional, low power)
  - FULL (performance)
- Autonomous manager integrated within the real-time scheduler

PM Event Propagation



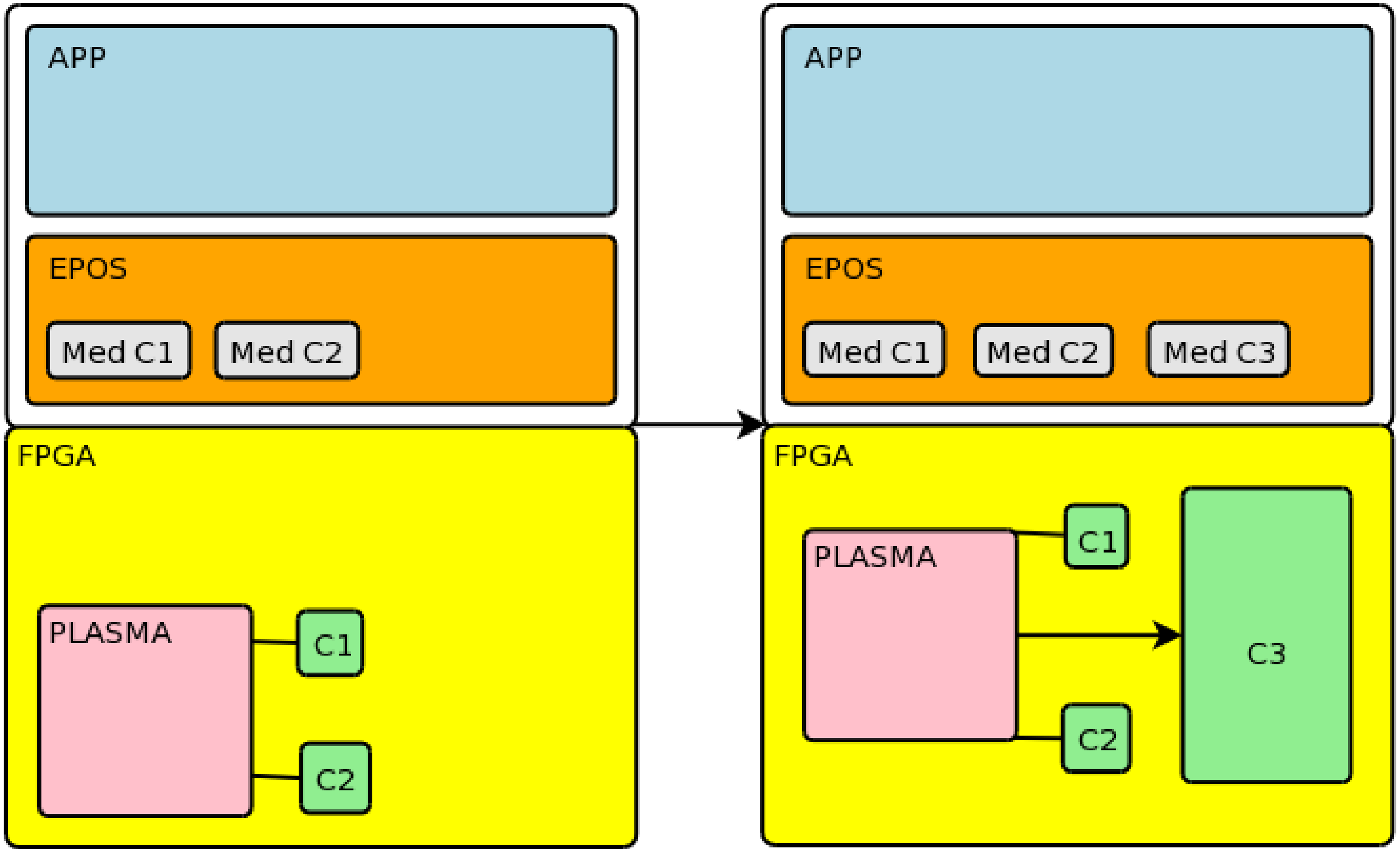
Software Update in EPOS





Dynamic Reconfiguration in EPOS

- PM + SW Update + mediators



Final Remarks

- ADESD
  - Complements traditional ES methodologies with a domain engineering strategy
  - Extends the notion of platform to multiple architectures (hardware mediators)
- EPOS SoCs
  - Automatically generated by tools according with application requirements
    - Properly designed IPs
    - Hardware mediators for the target machine
  - Limited by current HDL (aspects, metaprograms, etc)