The cost of communication protocols and coordination languages in embedded systems

Kees Goossens
Om Prakash Gangwal

background

- Philips
  - consumer electronics, components, medical systems, …
  - consumer/embedded systems
  - hardware, software
  - cost, cost, cost

- communication protocols & component models are important
  - complex real-time systems with huge computational loads
  - exponential software content of TVs
  - product families (Koala)
  - exponential number of blocks on ICs
  - software shipped with components (ICs)

background

- existence of several communication protocols
  - is this desirable or required?
- it can be motivated by cost
- we give a qualitative analysis
  - awareness of cost factors
  - what to optimise for

overview

1. things we make
2. a cost break down

3. protocol services & implementations
4. their relation

5. three communication protocols

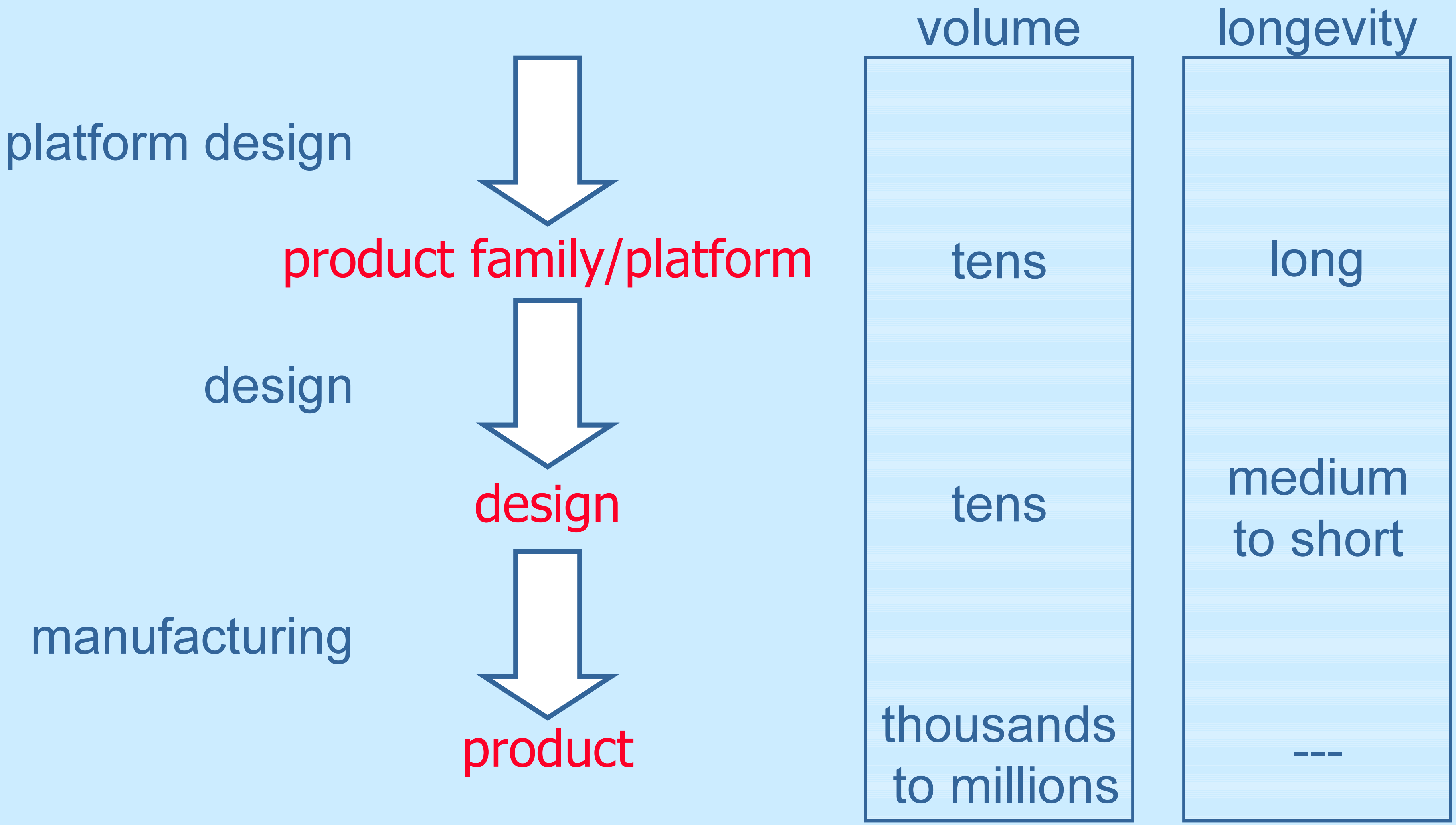
6. cost of services & implementations

7. final remarks

things we make

- product family
  - TV models with shared characteristics (**application domain**)
  - **platform** (template, design method) designed once
  - **re-used** over multiple designs
- design (blue print, implementation)
  - a TV model
  - instance of a product family
  - **designed once**, instance of a platform
- product
  - a single TV
  - instance of design
  - **manufactured** many times

things we make



platform design

product family/platform

design

design

manufacturing

product

	volume	longevity
product family/platform	tens	long
design	tens	medium to short
product	thousands to millions	---

different types of costs

1. platform design
   - not further addressed
2. cost of designing
   - for every design
3. cost of manufacturing
   - for every product
   - "the cost of the design"

different types of costs

- cost of designing (for every design)
  - non-recurring engineering costs (NRE)
    - time & man power to design, integrate, test/verify/simulate
  - is amortised over number of products sold
  - products have short life times
  - lead products
    - have higher profit margins
    - sell more products

⇒ reducing time to market is essential

different types of costs

- cost of manufacturing
  - for every product
  - bill of material (BOM)
    - software
      - code size, type of code
    - hardware
      - chip area, power dissipation, EMI, pin count
  - cost of testing
    - testing chip in factory
      - redundancy, fuses
    - yield

⇒ any (small) gain is multiplied many times

aside:
hardware is cheapest

significant part of cost

interaction language services

- an interaction language offers its services via an API

- service classification
  1. coordination or configuration
     - reconfiguration, discovery, leases
  2. communication or steady-state
     - data transfer, synchronisation
     - memory management, task scheduling
  3. inspection
     - configuration: task activity, deadlock
     - communication medium: polling

sliding scale:
- scheduling
- mem management

interaction language implementations

- classification of services
  - local versus global
  - static versus dynamic

local	& static
global	& dynamic

- classification of implementations
  - centralised versus distributed
  - hardware versus software
  - emulated versus native

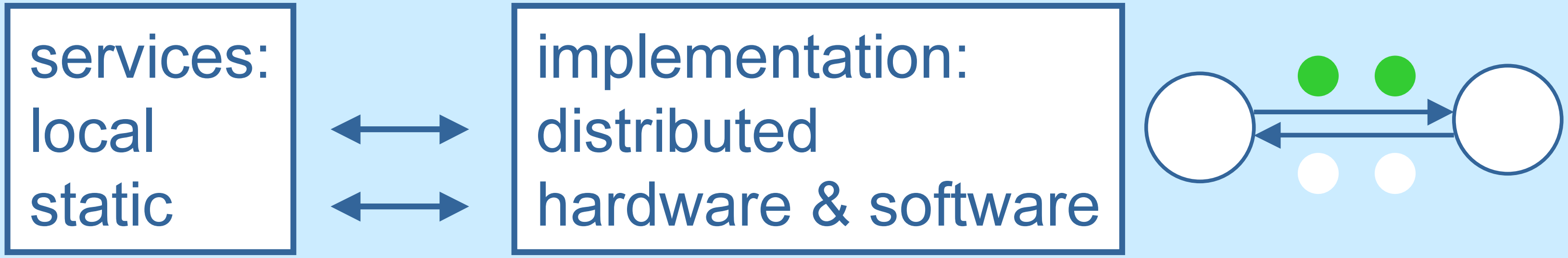
emulated	& software
native	& hardware

- services and implementation must be balanced

global	& centralised
local	& distributed
dynamic	& software
static	& hardware

three communication protocols

- c-heap
  - "CPU-controlled heterogeneous architectures for signal processing"
  - point-to-point channels
  - communication based on fixed-size tokens
  - local, static buffer allocation to channels
  - task & channel reconfiguration
- optimise: memory in static system

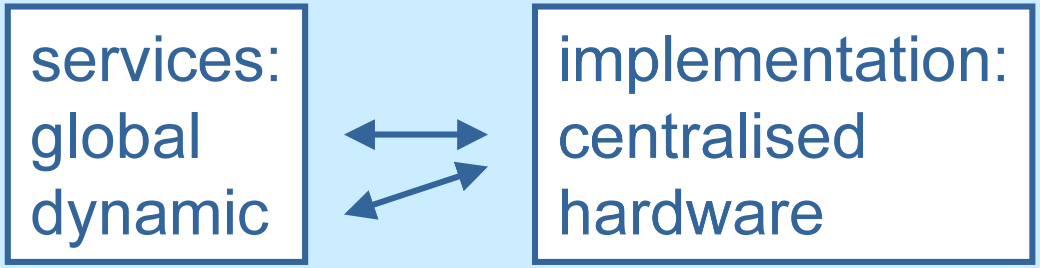


services:
local
static

implementation:
distributed
hardware & software

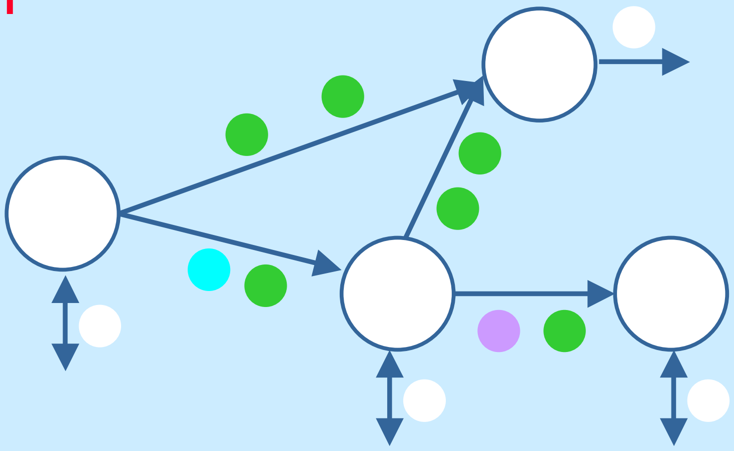
three communication protocols

- Arachne
  - multi-cast, narrowcast, and merge channels
  - token-based communication
  - global, dynamic buffer allocation to channels
  - channel reconfiguration only
- optimise: memory in dynamic system



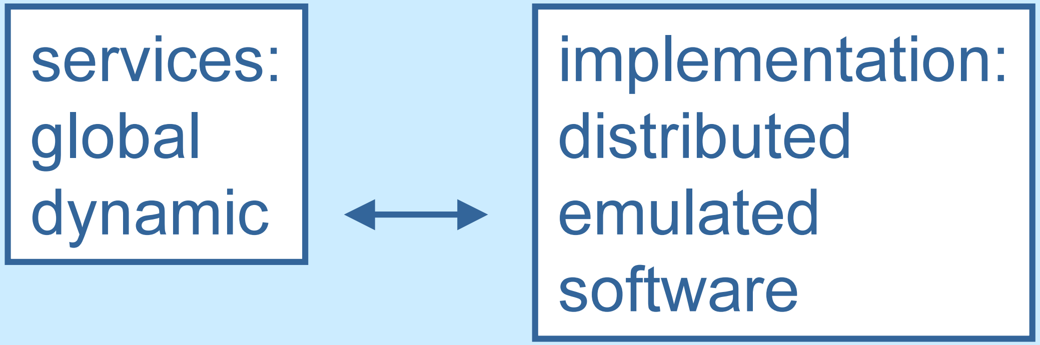
services:
global
dynamic

implementation:
centralised
hardware



three communication protocols

- space-time memory (STM)
  - out-of-order, multiple reader/writer channel
  - (over)sampling
  - garbage collection
  - task & channel reconfiguration
- optimise: memory in dynamic system



services:
global
dynamic

implementation:
distributed
emulated
software

interaction languages and costs

- cost of designing
  - affected by the services of interaction language
- cost of manufacturing
  - affected by the implementation of interaction language

interaction languages and costs

	amortised over
• cost of designing – abstraction, structuring, decomposition – application domain tailoring – re-use	design method application domain product families & design method
• cost of manufacturing	

interaction languages and costs

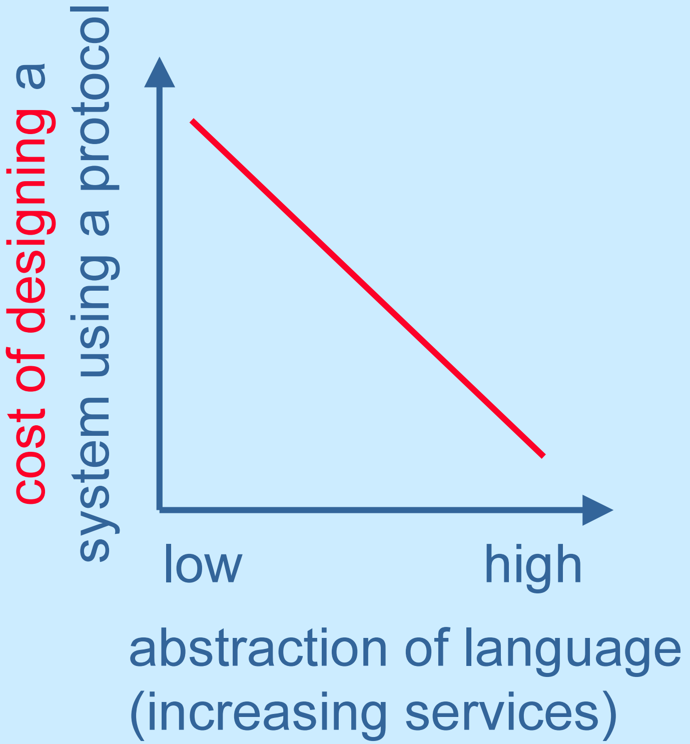
	amortised over
• cost of designing	
– abstraction, structuring, decomposition	design method
– application domain tailoring	application domain
– re-use	product families & design method
• cost of manufacturing	
– cost of use of interaction language	product
– interaction language implementation cost	product
– running cost	product

services and the cost of designing

- more services make it easier to design
  - abstraction
  - application domain
  - re-use

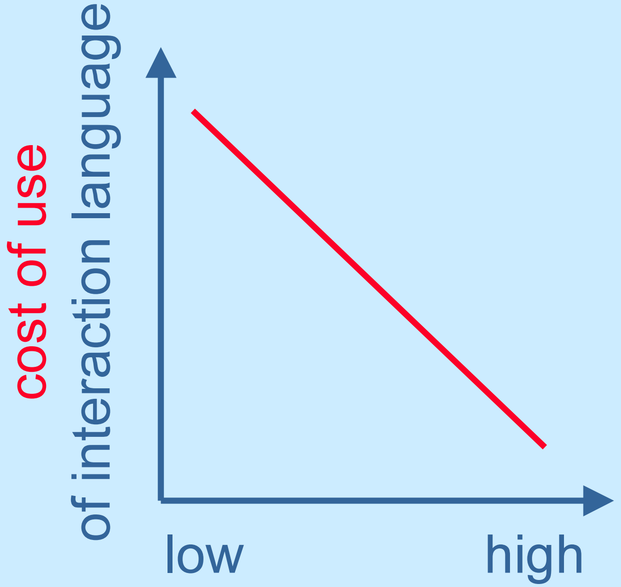
- data transfer
- synchronisation
- memory management
- task scheduling
- event notification

- load/store v. message passing



implementations and the manufacturing cost

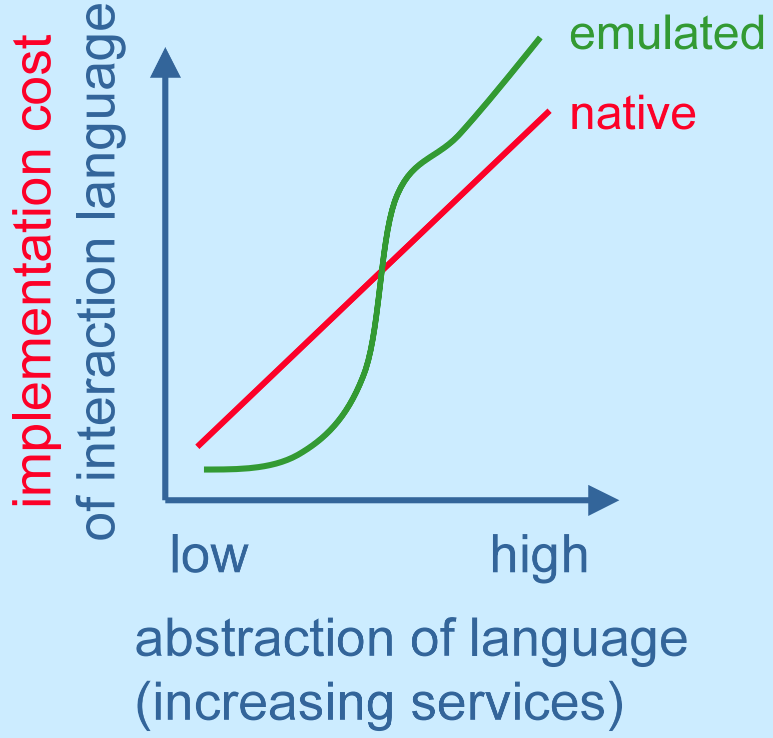
- cost of use
- more services make it cheaper
  - doit() versus load/store



abstraction of language
(increasing services)

implementations and the manufacturing cost

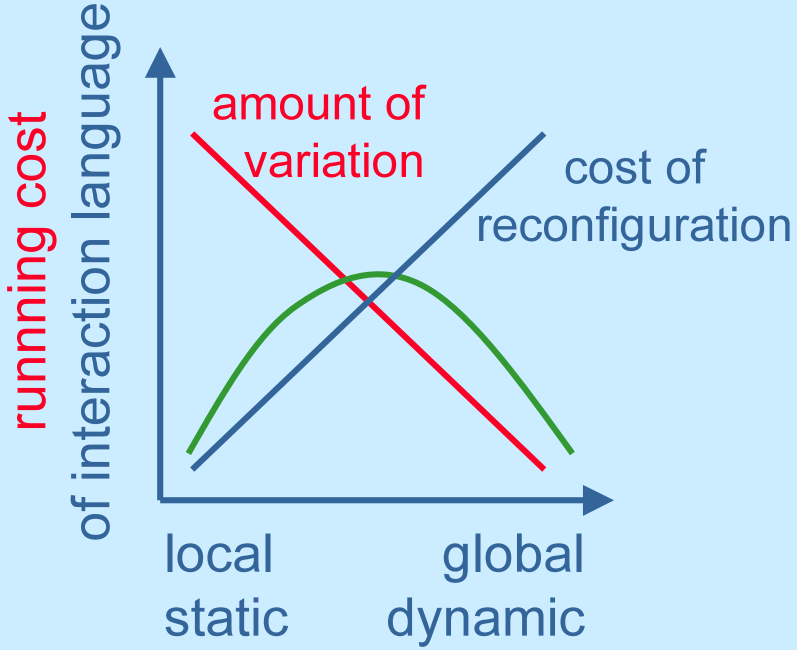
- interaction language implementation cost
- more services make it more expensive
  - doit() versus load/store
  - local versus global
  - static versus dynamic
  - emulated versus native



implementations and the manufacturing cost

- running cost
- more services make it cheaper
  1. global versus local
  2. dynamic versus static
- but consider
  1. variation in space or time
  2. cost of reconfiguration

- memory management
- local static scheduling
- multiprocessor pre-emptive scheduling



closing remarks

- for embedded systems in consumer market cost is key
- services and implementations are related
- more services
  - + reduce cost of designing
  - + reduce cost of use
  - – raise implementation cost
    - protocol stripping may help
  - – may raise or reduce running cost
    - variation/overhead v. dynamism/locality trade off
- potentially a bright future for interaction languages