UC Leuven Limburg

MOVING MINDS

Internet of Things? Let's get real!

Paul Valckenaers

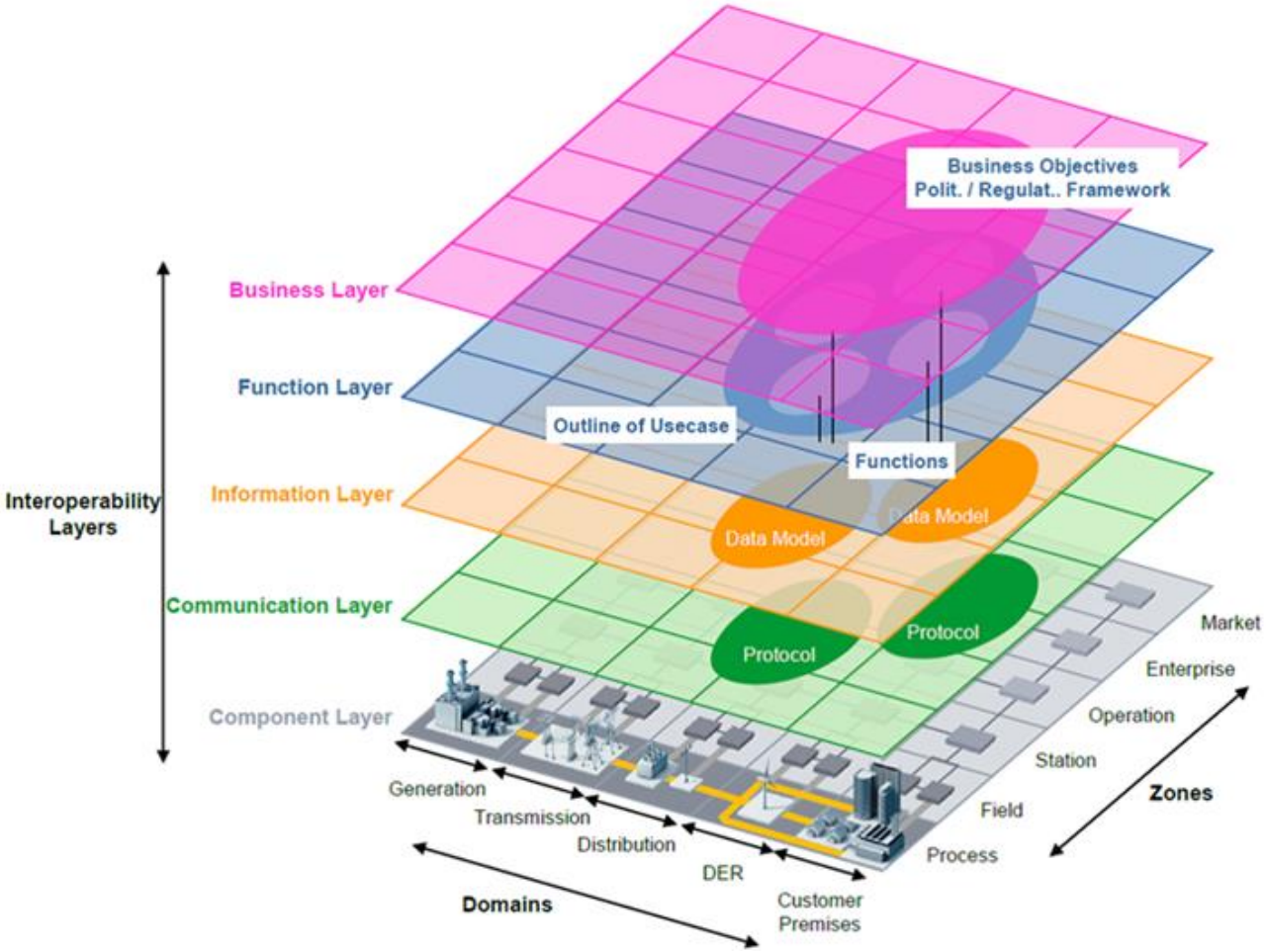
paul.Valckenaers@ucll.be

Software boldly goes where no software has gone before

Software <-> Reality – the gap is closing

- Software becomes directly involved with and connected to reality (no human intermediary)
  - Sensors –> observe and measure
  - Actuators –> affect and influence
- Software is becoming 'nearly-mission-critical' to everyone, everywhere, all the time, …

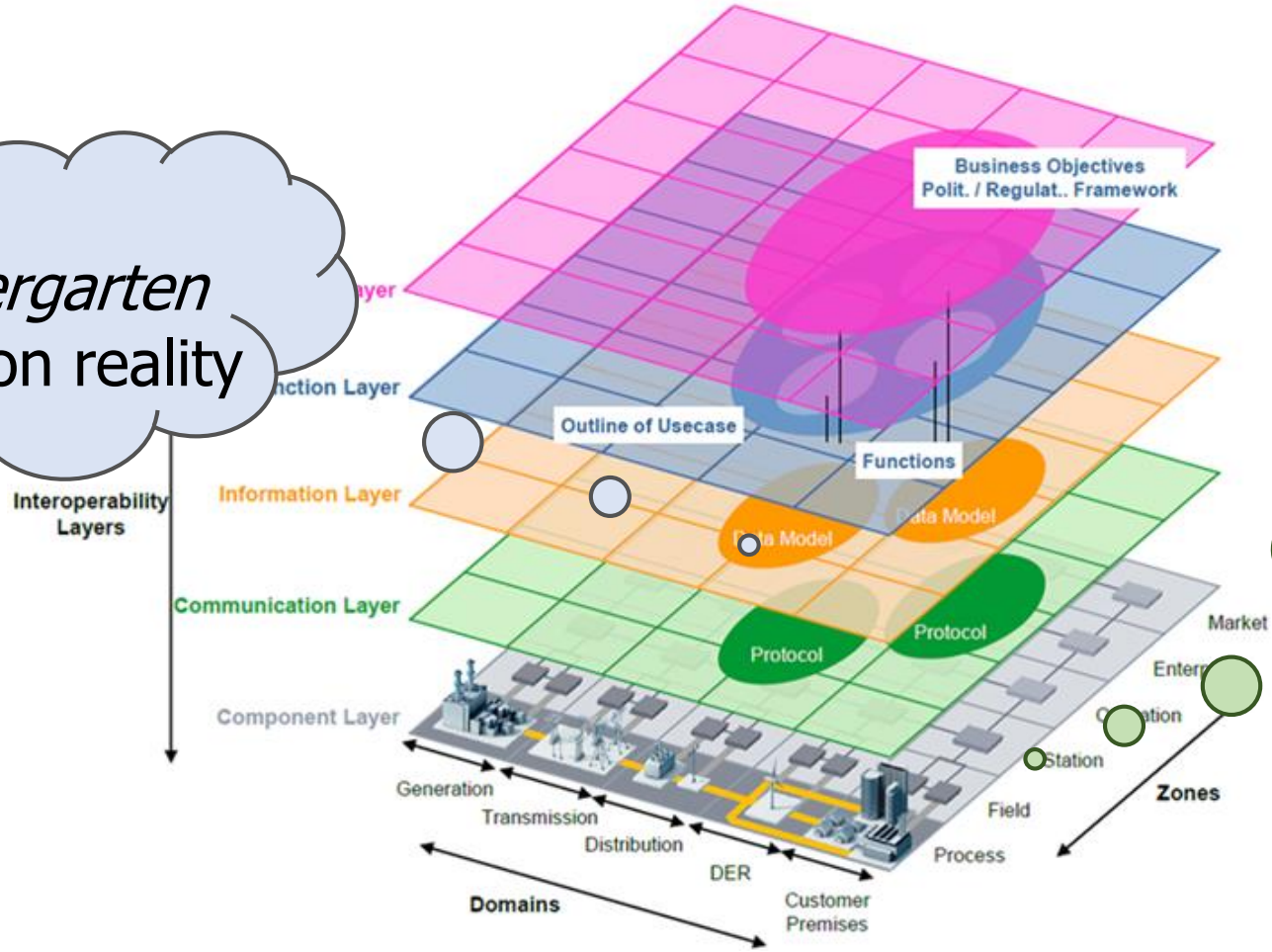
IT architectures <–> reality



Smart Grid Architecture Model (SGAM)

IT architectures <–> reality

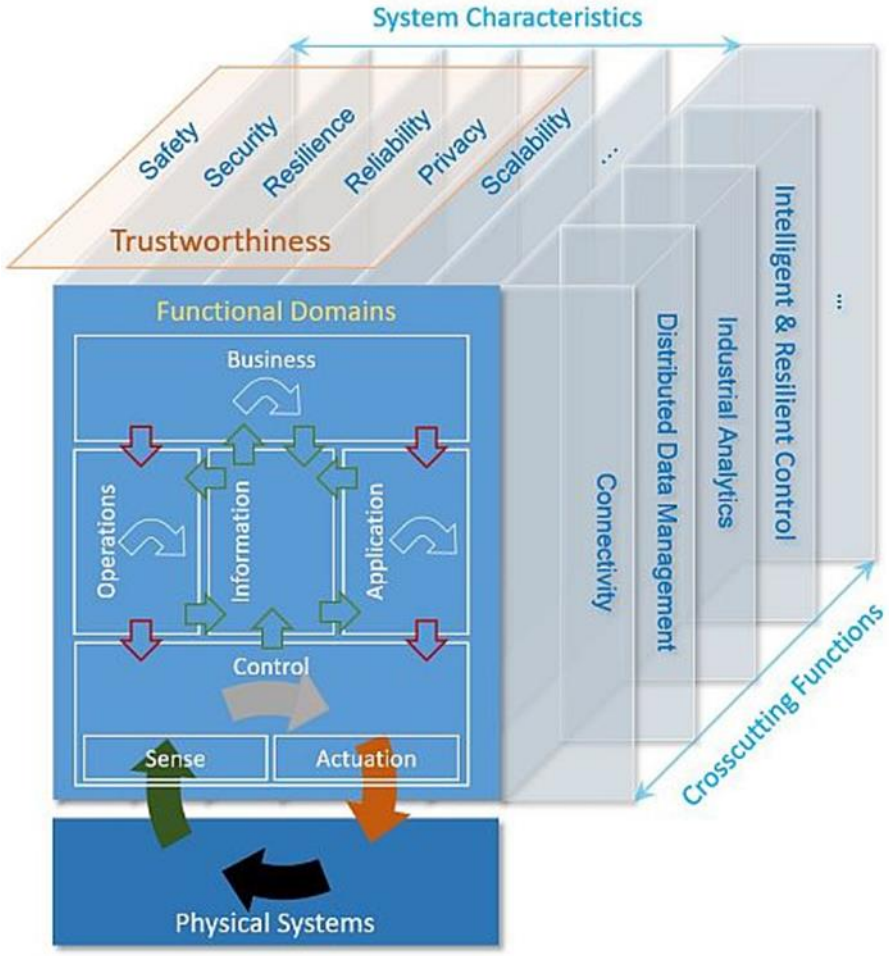
Kindergarten views on reality



Smart Grid Archi (SGAM)

Layer for Reality

IT architectures <–> reality

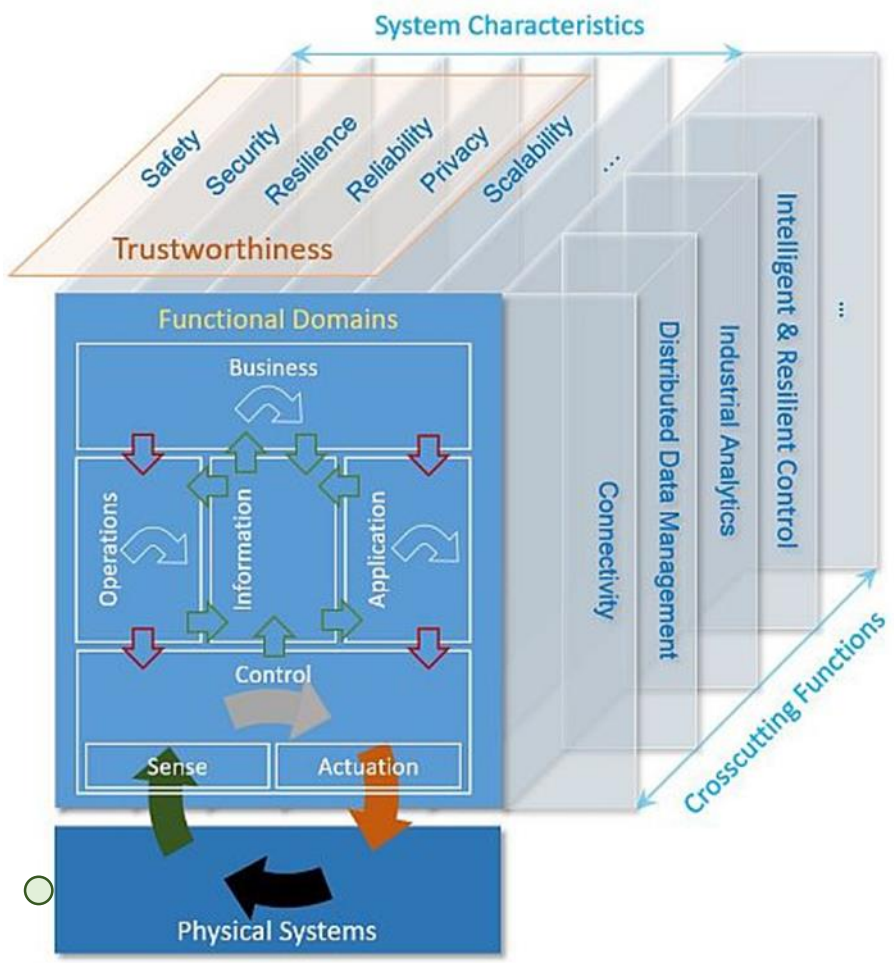


Industrial Internet of Things

IT architectures <–> reality

Only 1 box for reality!

Owned by control.



Industrial Internet of Things

IT-centric architectures

- Cover the ICT
  - E.g. communication protocols, connectivity
- Cover the business/user (wish lists, rules, policies…)
  - I.e. what the system should do, not what it will do
- Reveal how/what of reality is 'perceived by the S/W'
  - E.g. a single 'physical systems' box without any details/structure
- Suffer from data model lock-in (cf. legacy frustration)
- …

IT-centric architectures

Reflection services for reality are 'lacking'.

Type information:

dimensions, response times,
commands, diagnostics, power,
flow rates …

Instance information:

state (open, closed, unknown),
time and numbers of switches since … ,
URI of neighboring device instances …

https://commons.wikimedia.org/wiki/File:Pl_control_valve.jpg#/media/File:Pl_control_valve.jpg

IT-centric architectures

Are missing:

- **Allocation services for real-world resources**

  E.g. A car parking space can only hold a single vehicle at any given instance in time.

- Without allocation services, applications need to ‘know each other’.

- Without a single source of truth for the resource, problematic ‘coupling issues’ will arise.

IT-centric architectures

Do **not** provide / mandate / cover :

- **State** modeling / information for reality
  - On-going and past activities
- **Future** states and trajectories
  - Possible/feasible? A good or bad idea?
  - Future interactions to be expected?
- **What will happen**, when given 'how the system will attempt to achieve *what should happen'*

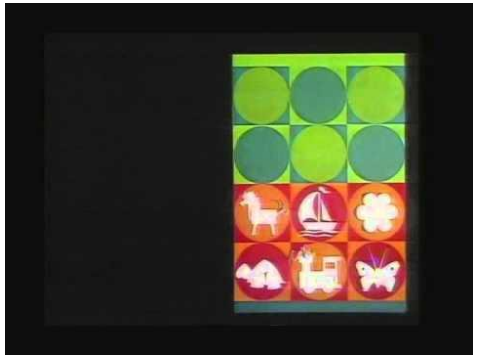
Warning

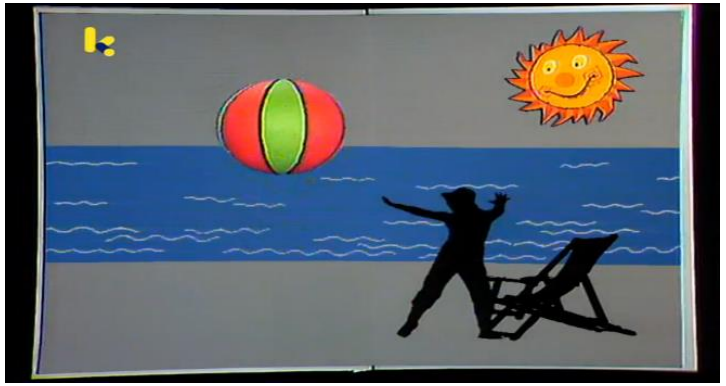
We are leaving the Internet !

Moving into the real world

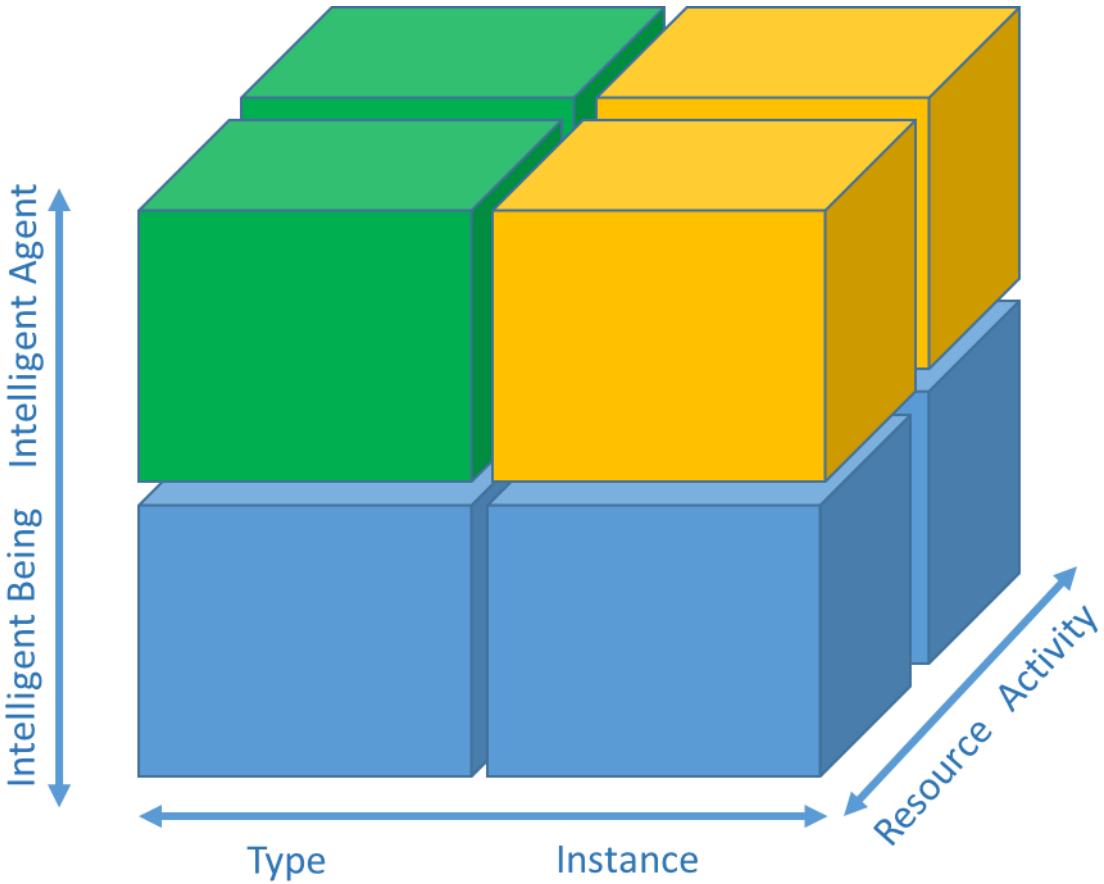
But your IT background will be accounted for

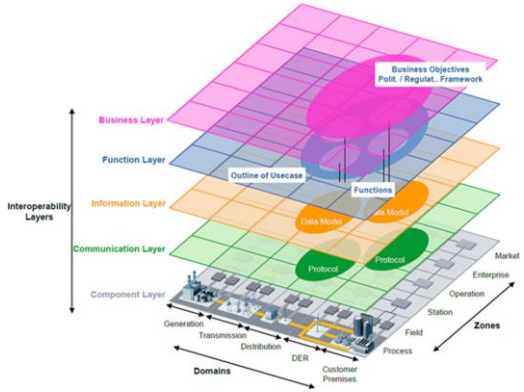
Don’t panic. We will stick to the *media guidelines* for the intended audience ;-)

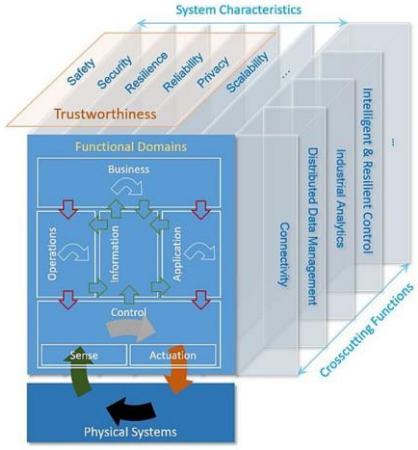




Reality first, second and third





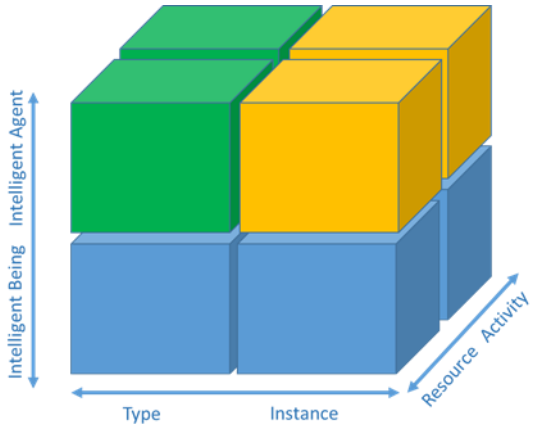


ARTI architecture – reality-centric

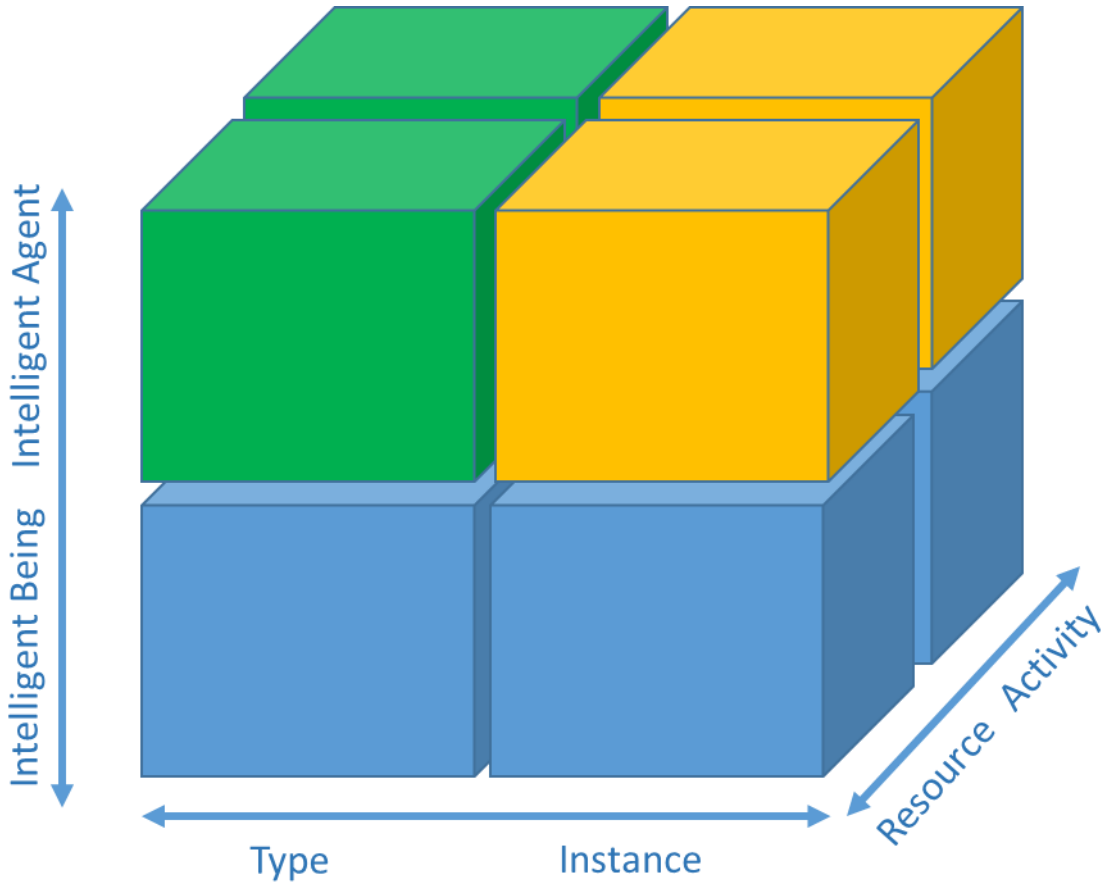
ARTI – Application domain

Activities executing on Resources

- E.g. Google maps
- E.g. not Facebook



Reality first, second and third



ARTI architecture – reality-centric

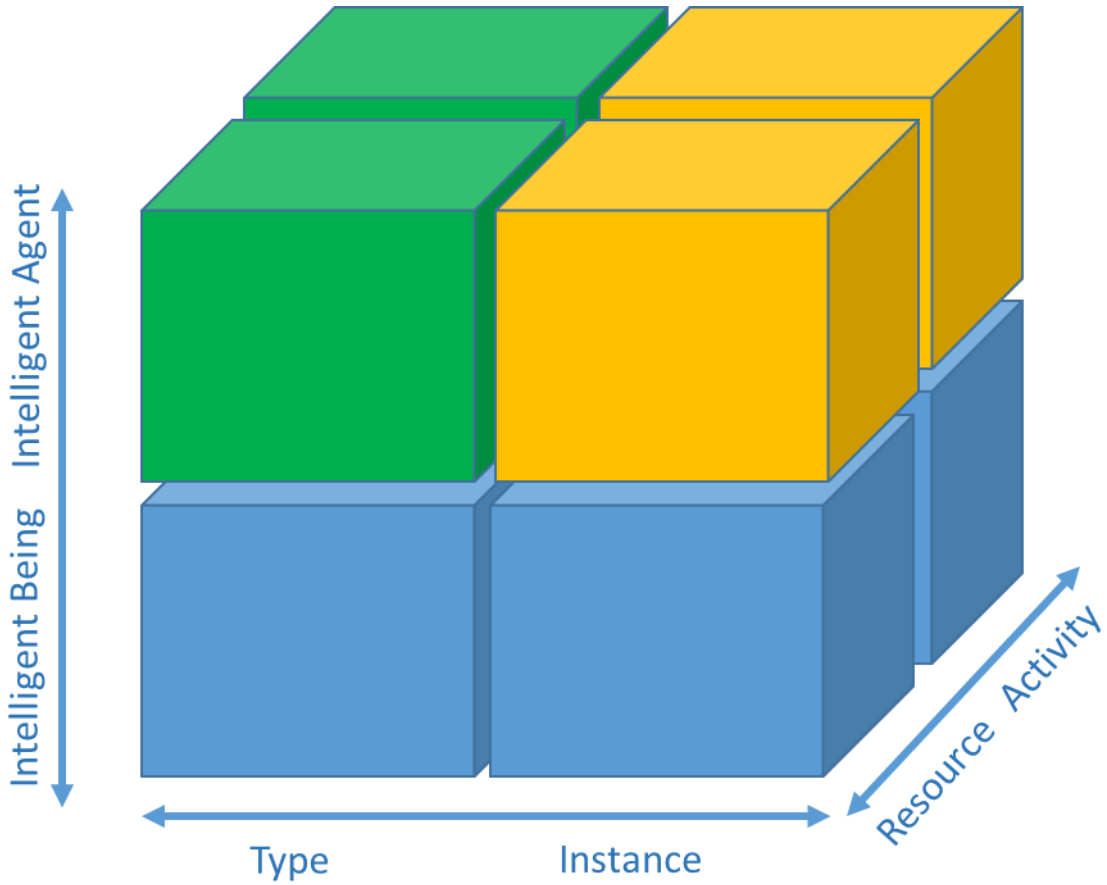
Resources

- Parking space
- Road segment
- Roundabout

Activities

- Driving from A to B
- Parking from dusk till dawn
- Crashing into a tree

Reality first, second and third



ARTI architecture – reality-centric

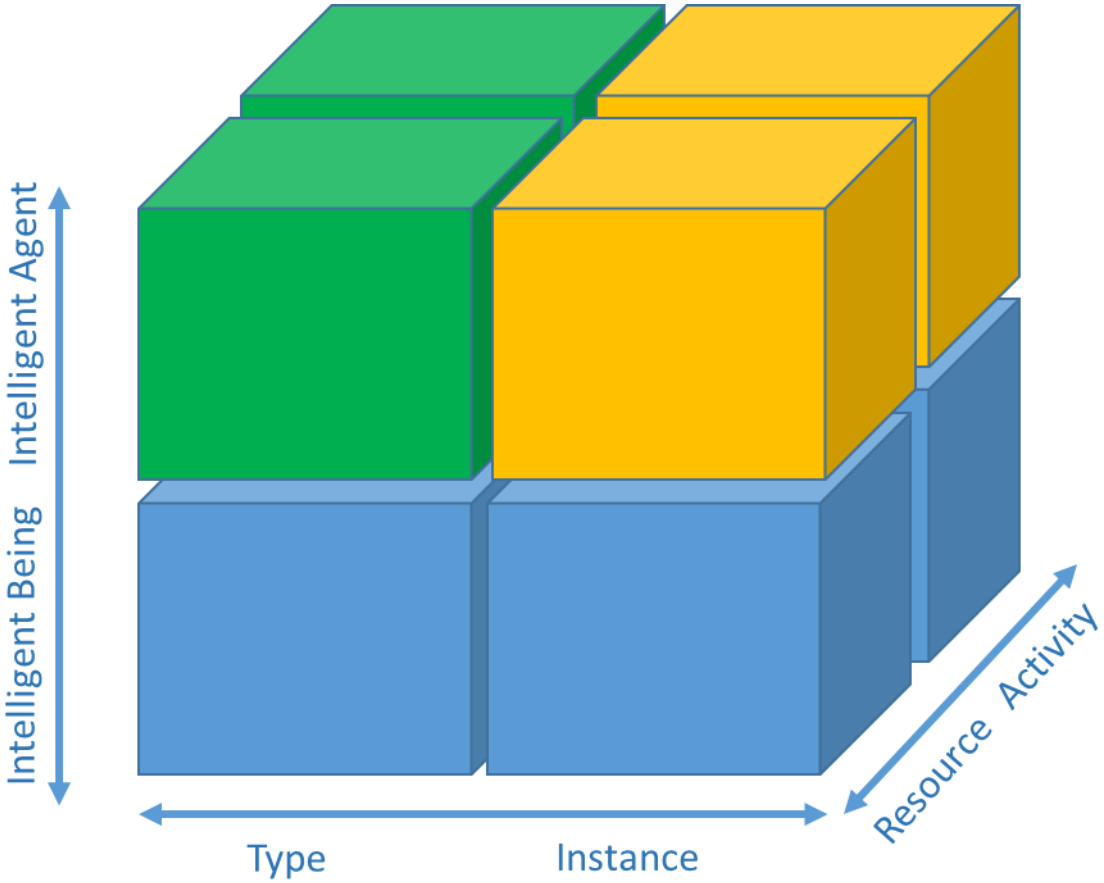
Type

- Ford Fiesta
- Routes from A to B

Instance

- Your Ford Fiesta on the parking lot, half empty fuel tank, …
- Your spouse halfway between A and B taking the scenic route

Reality first, second and third



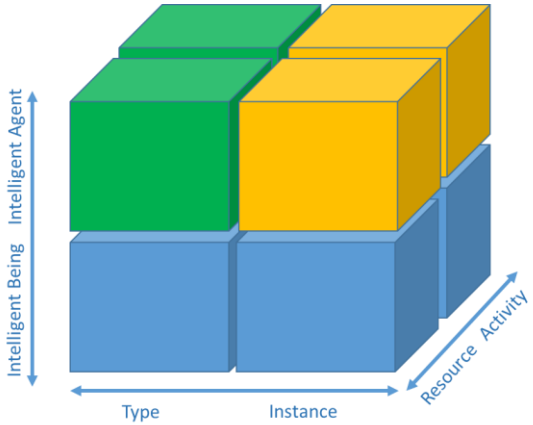
ARTI architecture – reality-centric

Intelligent agent
- Trajectory generator

Intelligent being
- Road segment reflection

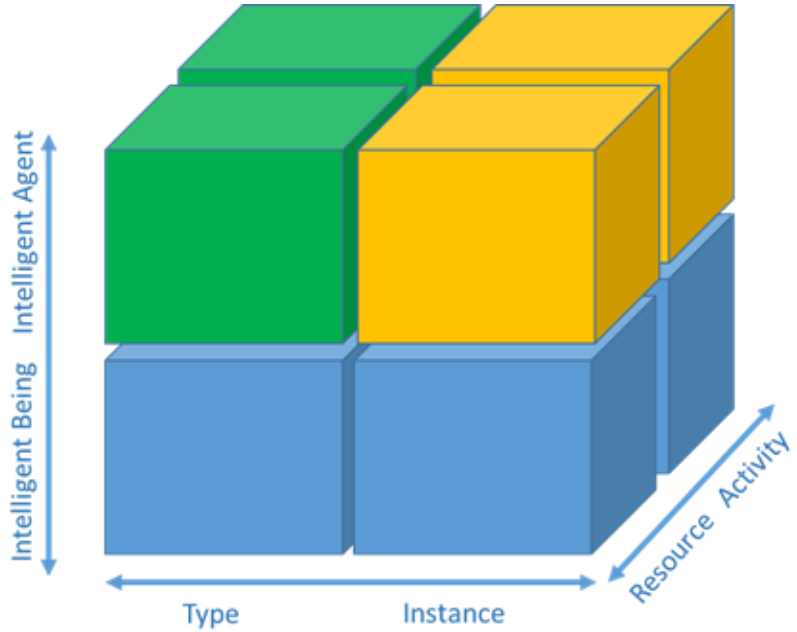
ARTI – Looking at reality

- Perceive the world-of-interest in a specific manner
- Distinguishing:
  - Resources – Activities
  - Types – Instances
  - Reality-reflecting (blue) – Decision-making (yellow and green)



ARTI – Usage

- Map* your software components/modules on ARTI:
  - Which cubes are involved in a single software module?
  - Which colors are present in a single software module?



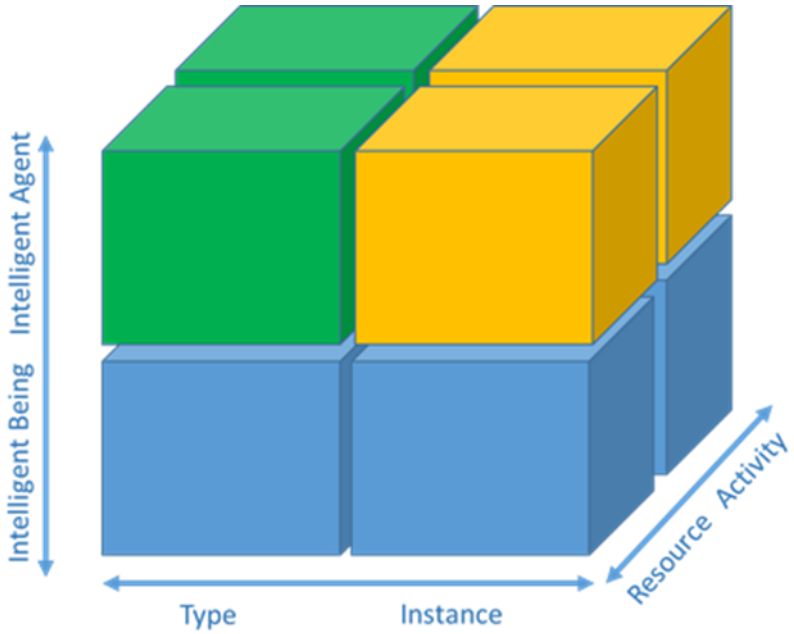
*Note that ARTI is complementary to SGAM, Industrial IoT …

ARTI – Usage

Map your software on ARTI:

E.g. a road map fits in the resource-related blue cubes

- Map legend fits in …
- Map updating software and organization fits in …



ARTI – Usage

Map your software on ARTI:

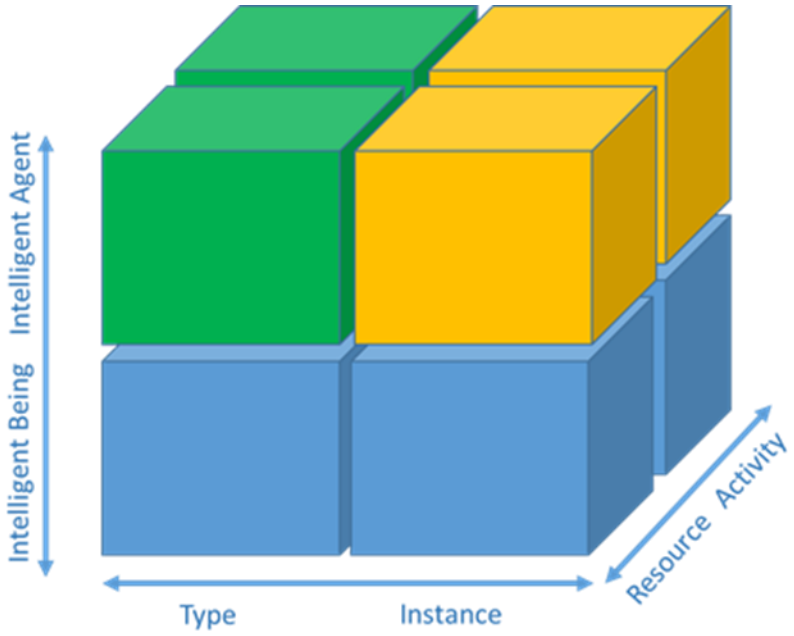
Examples (continued):

- Routing algorithms fit

  in the green cubes

- Route descriptions span

  blue and yellow cubes



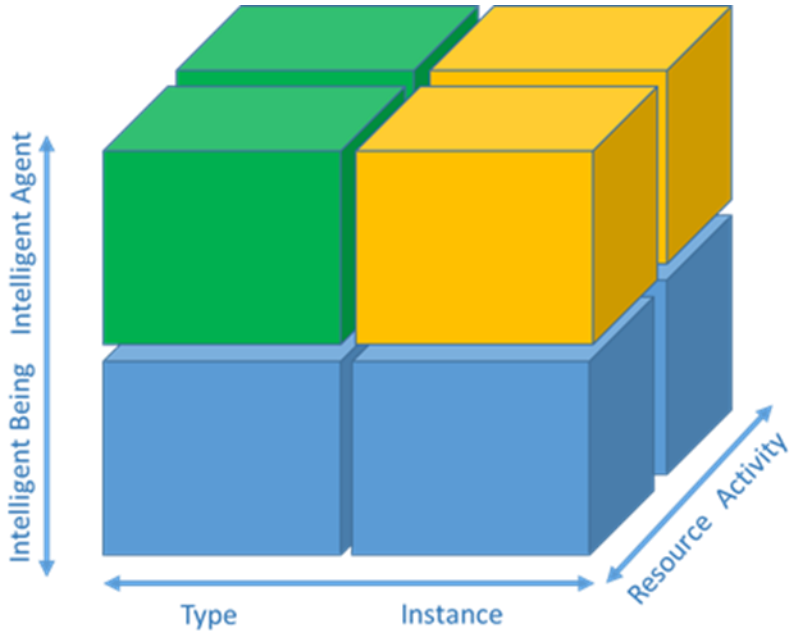
- Navigation systems span all 8 cubes (note: not monolithic)

ARTI – Usage

Map your software components/modules on ARTI:

Crossing cube borders has user mass related issues.

Mixing with yellow has in-depth interoperability issues.



User mass

- Does user mass matter for IT companies?

Google

Apple

IBM

Intel

ARM

Amazon

***

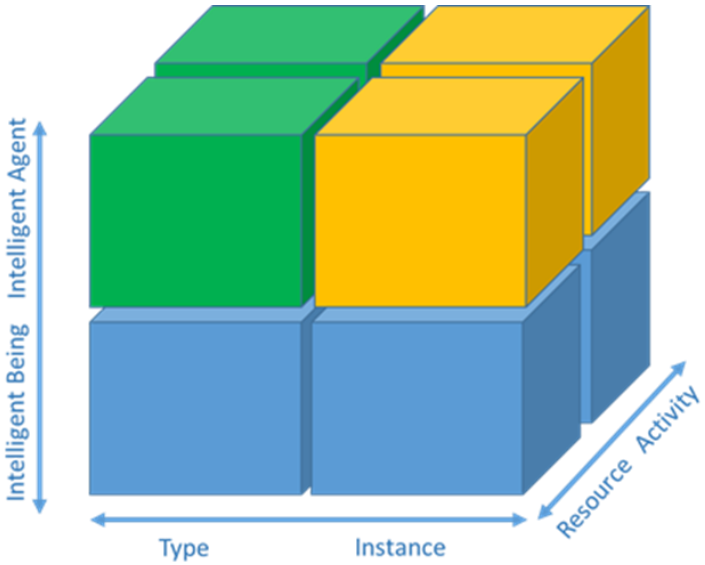
>>> Increasing returns

ARTI – User mass

Crossing cube borders has user mass related issues.

Important for the blue cubes

a. Types need experts
b. Instances need managers
c. Resources & Activities



Cross a border >>> intersection of the potential user masses

>>> order(s) of magnitude smaller, possibly

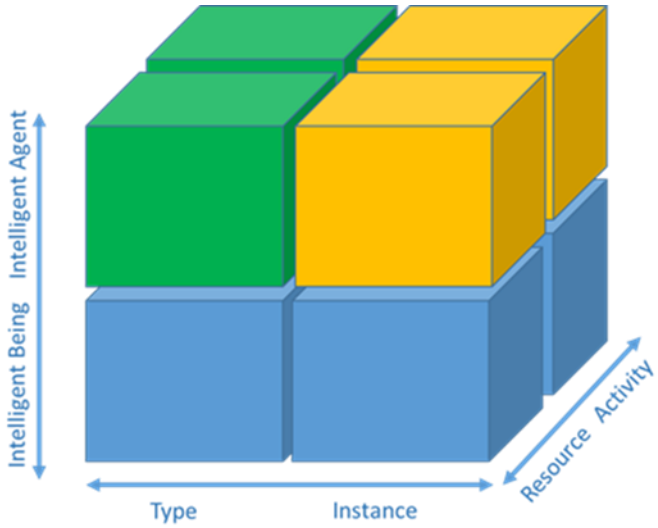
ARTI – User mass

Not an issue for the green cubes (= software libraries)

I.e. they do not cross borders

Less of an issue for yellow cubes (= beyond repair)

I.e. installation & time dependent



In-depth interoperability

https://commons.wikimedia.org/wiki/File:Caleb_Mendez_Soccer_09.jpg#/media/File:Caleb_Mendez_Soccer_09.jpg

https://commons.wikimedia.org/wiki/File:Arian_Foster_fumble.jpg#/media/File:Arian_Foster_fumble.jpg

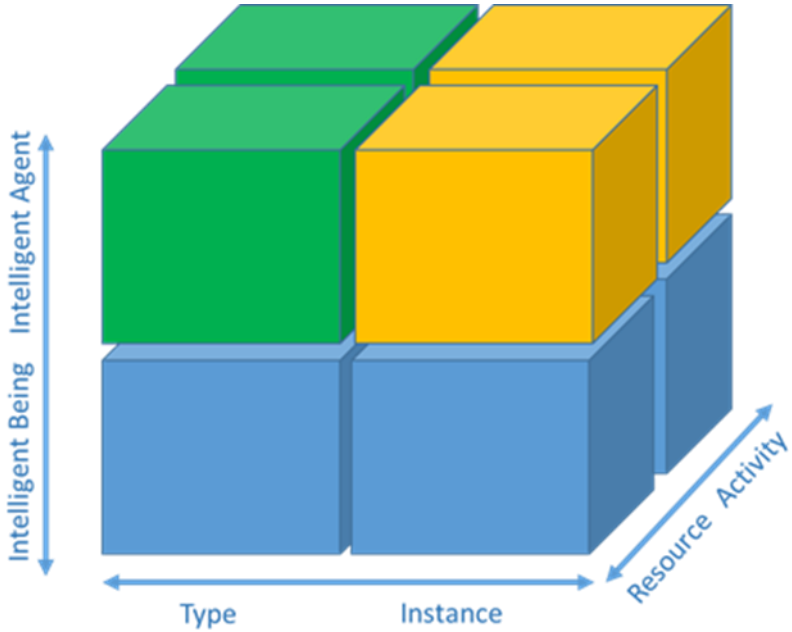
ARTI – In-depth interoperability

- Effective interoperability matters for IT companies?

  Mixing with yellow has in-depth interoperability issues.

Yellow cubes represent all (real-world) design choices.

These choices add limitations that are non-existent in the world-of-interest.

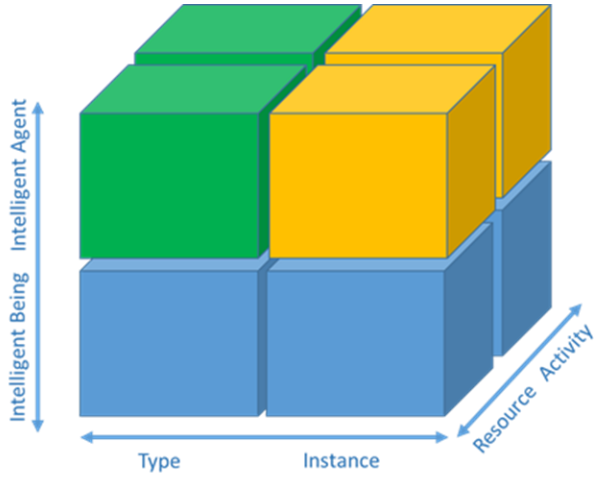


ARTI – In-depth interoperability

- Effective interoperability matters for IT companies?

  Mixing with yellow has in-depth interoperability issues.

  - Yellow cubes determine system performance, functionality.
  - Avoidable s/w conflicts reside within the yellow cubes, and only in the yellow cubes.



Software modules spanning a yellow cube become yellow.
Modules depending on a specific yellow module are yellow.

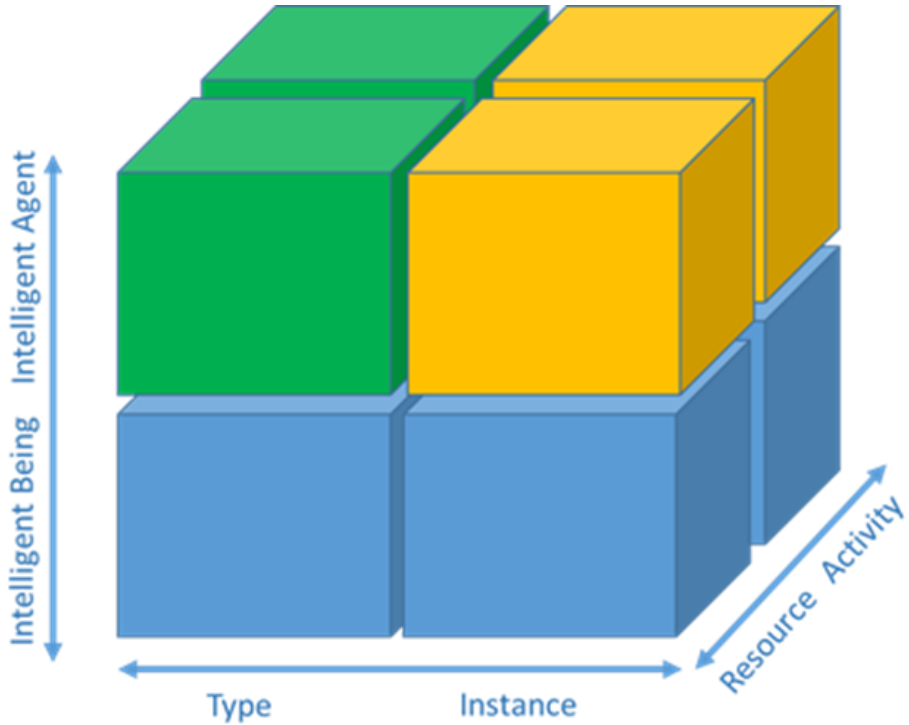
Interoperability – Remarks (1/2)

- Green = technology
  - When causing a conflict:

    replace.

- Blue = decision-free

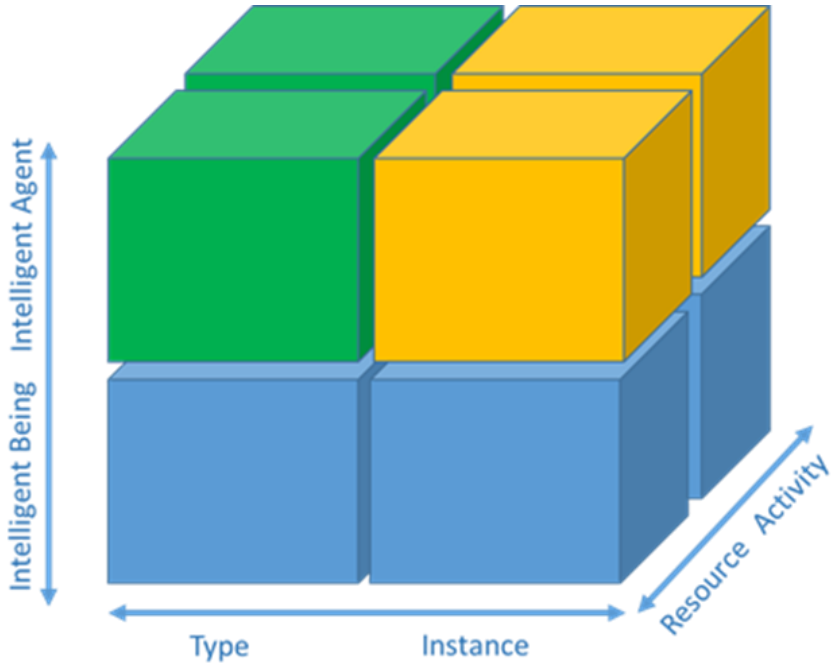
  reality reflection



  - When involved in a conflict: change reality and track.
  - When involved in a conflict: improve, expand, …

Interoperability – Remarks (2/2)

- Yellow = the choices
  - When involved in a conflict:
    adapt (some) yellow modules.
  - **Residence of evil;**
    here interoperability is lost
  - Determines how (well) the user requirements are addressed
    but should come last and must be minimized.



Concluding remarks – Let's get real

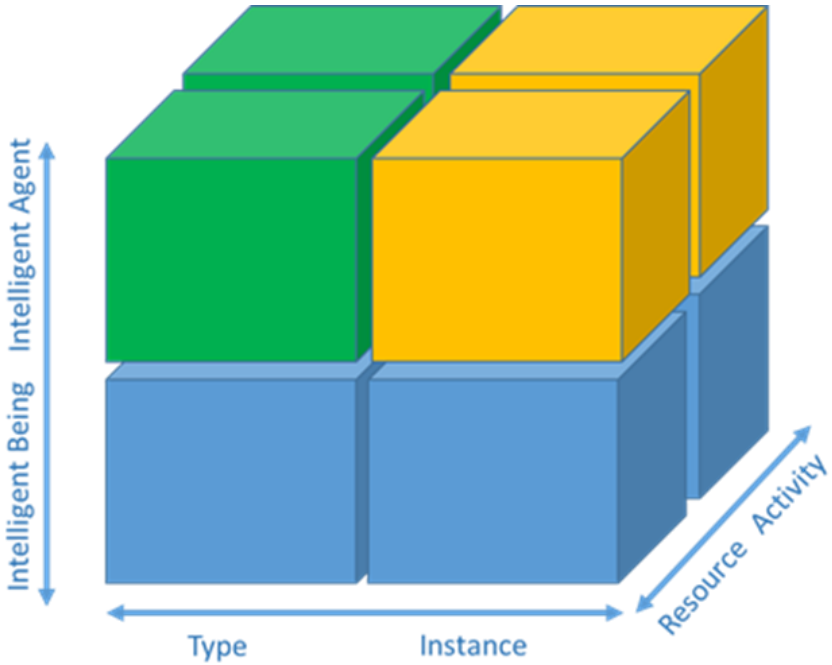
- Reality represents more value than IT
  - At least by one order for 'construction / development'
  - At least by two orders for 'operations / recurring costs'
    - **Architecture defined in terms of reality / world-of-interest**
- In depth interoperability
  - Avoid loss of value where it counts (i.e. reality)
  - Blue preferred over yellow, selecting green

Concluding remarks – Let’s get real

- Modularity
  - Potential user mass is key
  - Avoid hard-wired aggregation across (blue) cube borders
  - Dynamic / time-varying aggregation mirroring reality

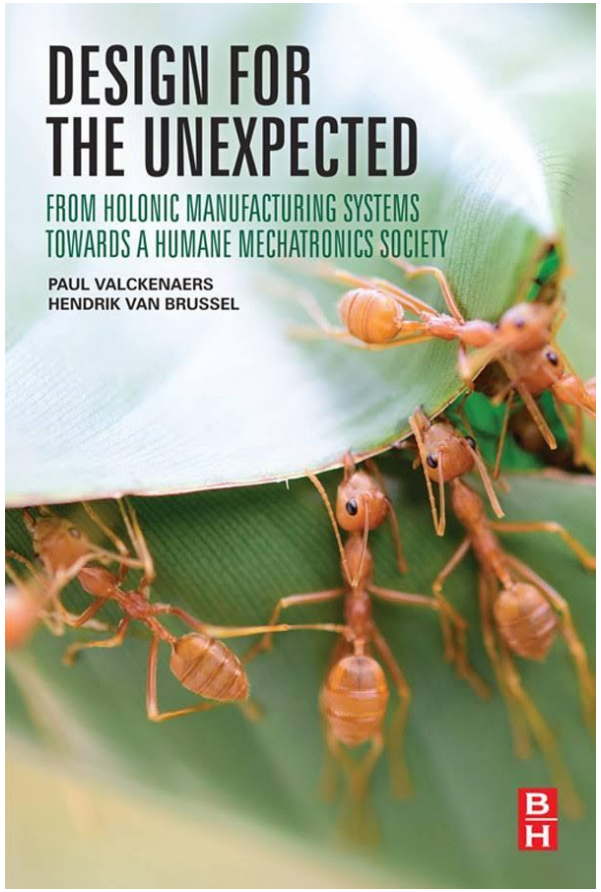
ARTI - Teaser

- Advanced and complete reflection (self-knowledge)
- Reflect yellow in blue



- What kind of service may emerges from this?

Further reading



Design for the Unexpected

P. Valckenaers and H. Van Brussel

Butterworth-Heinemann

18th November 2015

www.sciencedirect.com/science/book/9780128036624

Teaser answer: Predicting the unexpected!