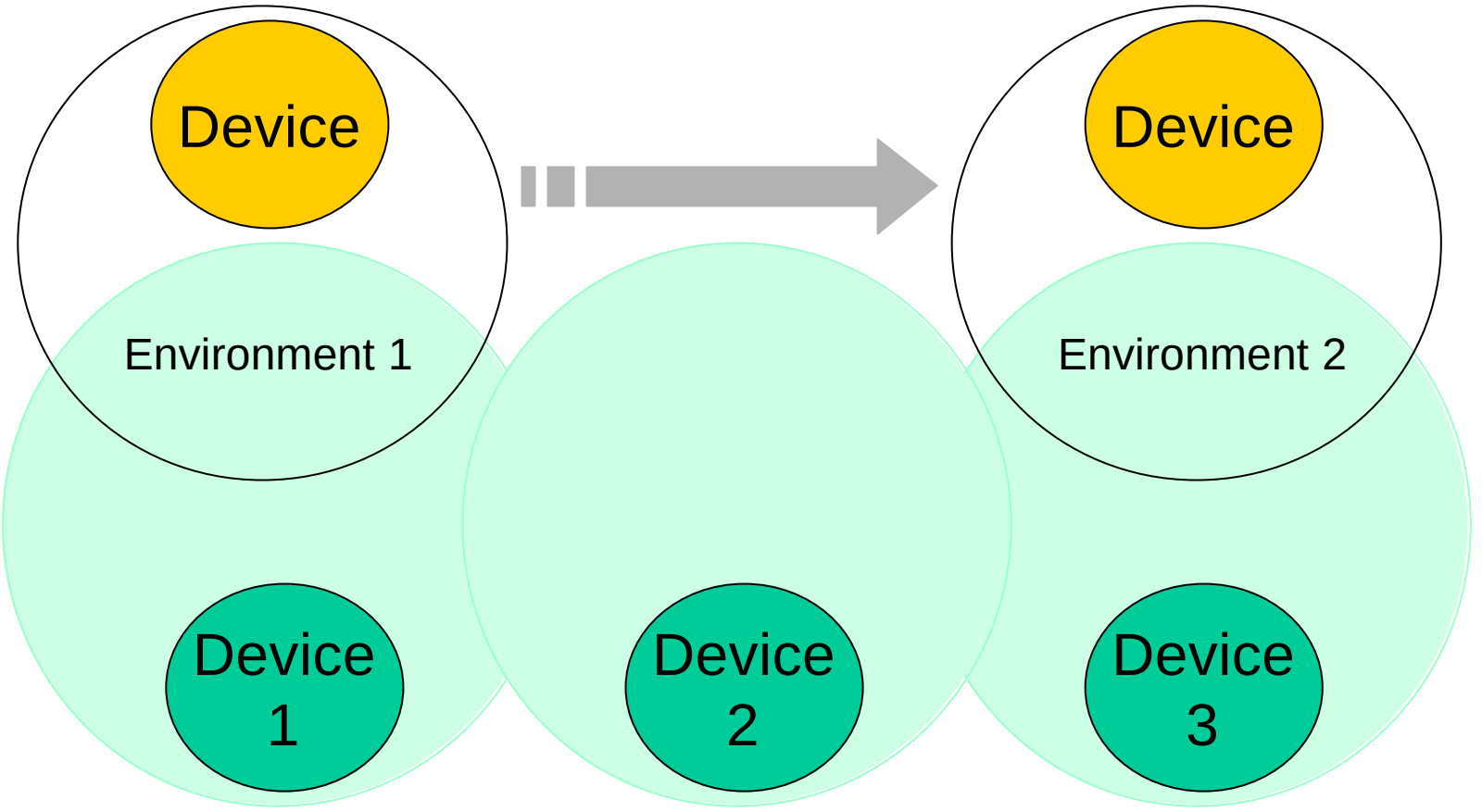
Agent Oriented Engineering

Three major technological waves:

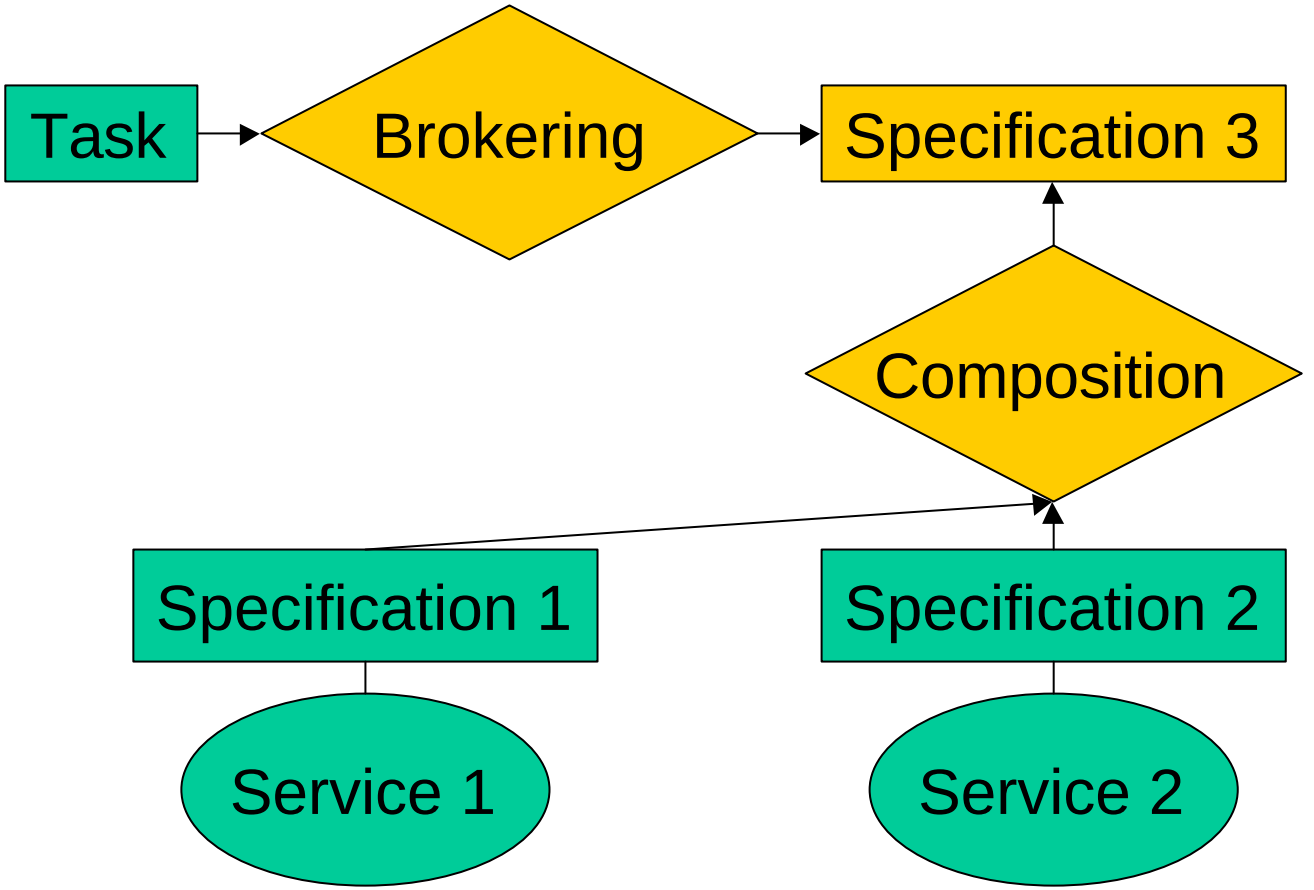
- Ubiquitous computing
- The Semantic Web
- Computational Grids

All these are (partly) agent architectures.

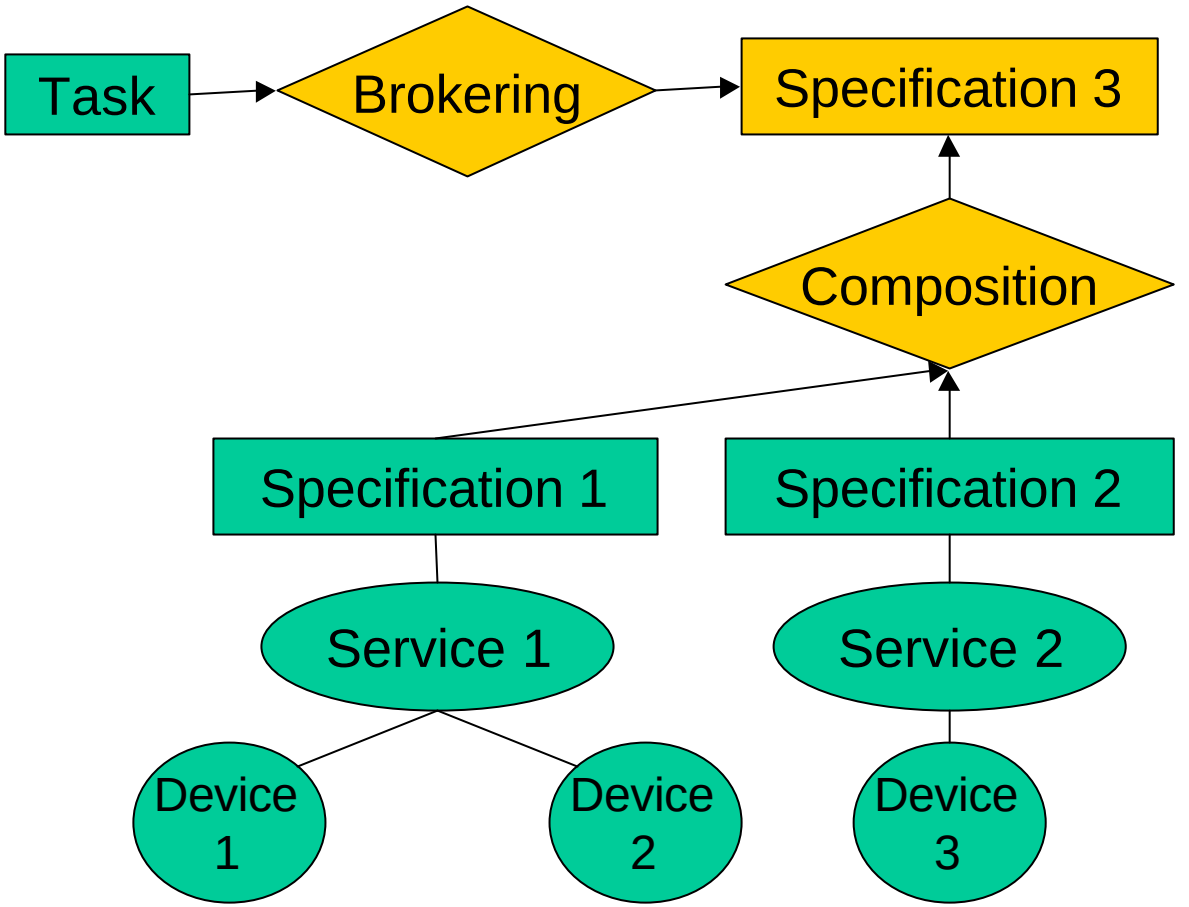
Ubiquitous Computing



Semantic Web



Computational Grid



Why Are These Similar?

- All assume millions of components.
- All want to minimise standardisation of components themselves.
- All assume autonomous components.
- All need standardisation of component interaction.
- All need opportunistic interaction.

Why Do They Look Different?

- Differing engineering traditions:
  - Ubiquitous: Communications
  - Semantic Web: Knowledge engineering
  - Computational Grid: Supercomputing
- Differing design priorities:
  - Ubiquitous: Opportunistic interaction
  - Semantic Web: Evolution from Web
  - Computational Grid: Reliability and performance

Why Should I Believe You Built a Well Engineered System?

- You can prove it is good from analysis of its structure.
- You used a trusted design process.
- You are a trusted engineer.

The Dilemma

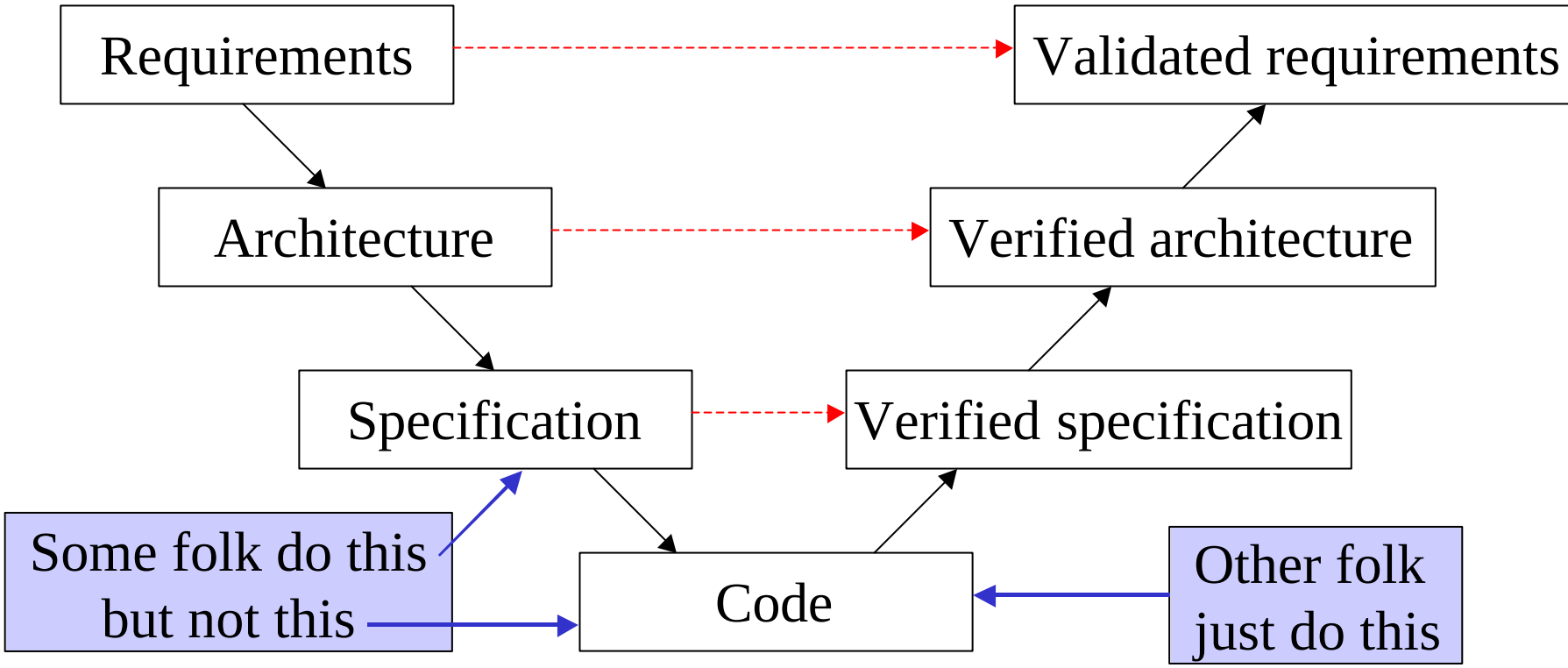
"What is particularly impressive is the way that scientists are now undaunted by important complex phenomena...The emerging field of e-science should transform this kind of work...One of the pilot e-science projects is to develop a digital mammographic archive, together with an intelligent medical decision support system for breast cancer diagnosis and treatment....So the surgeon in the operating room will be able to pull up a high-resolution mammogram to identify exactly where the tumour can be found."

Tony Blair, Speech to Royal Society, 23rd May 2002

“Design and Development: Software Architecture Design… Artificial Intelligence…NR [Not Recommended]”

IEC 61508 standard for safety-related software

Agents and S.E. Lifecycles



Issue 1: Social Protocols

- Naïve view is that, since agents operate autonomously they can be designed autonomously.
- Impractical – consider auctions.
- So need a separable definition of social protocol (or social norm or institution).

Issue 2: Specification Level

- Which aspects do we specify (*e.g.* knowledge, beliefs, temporal constraints,…)?
- Do these refine to code (*e.g.* institutions to object classes and FSMs)?

Issue 3: Spec. versus Deploy

A. Most agent formal specs don't execute.
B. Most executable code is in Java *et.al.*

- Hard to get from A to B.
- Options include:
  - Simulation
  - Constructive proof
  - Model checking

Issue 4: Aggregate Behaviours

- Naïve view of multi-agent design:
  - Build individual agents reliably.
  - Let them communicate via a dependable protocol
  - Now the multi-agent system is dependable
- What is the alternative?
  - Ignore the issue
  - Standardise
  - Build predictive models