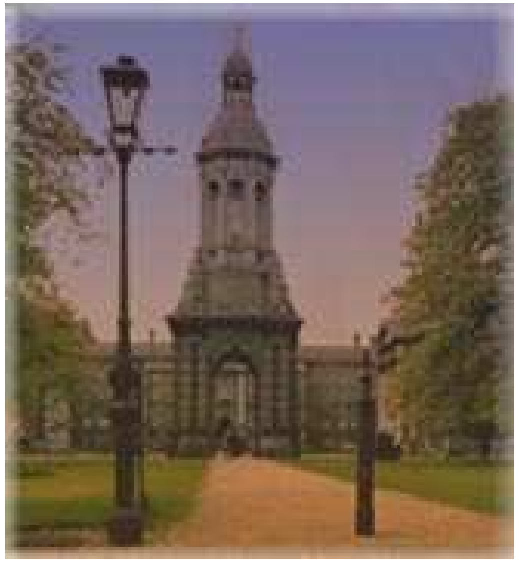
Simon Dobson

Distributed Systems Group
Department of Computer Science
Trinity College, Dublin IE

simon.dobson@cs.tcd.ie



Towards a semantics of pervasive computing

Overview

Pervasive computing is increasingly an area of research and commercial interest

Brings together disparate fields

- Languages, distributed systems, AI, vision, user interfaces, ...

Proving a little difficult to move beyond the laboratory

- Compelling real-world examples are scarce
- Difficult to make a compelling impact

Where understanding is lacking

What does a pervasive computing system *do*?

Or, put another way, how can we specify the behaviour we want in as simple and tractable way as possible?

- Easy for developers to design and analyse
- Easy for programmers to develop for
- A basis for building programming environments

In this talk we argue the case for a clearer semantic understanding, and present some very early work towards it

Structure

- What makes pervasive computing interesting (and difficult) in a programming sense?

Structure

- What makes pervasive computing interesting (and difficult) in a programming sense?
- Some “back to basics” insights

Structure

- What makes pervasive computing interesting (and difficult) in a programming sense?
- Some “back to basics” insights
- Some ideas for a *fibred categorical semantics*

Structure

- What makes pervasive computing interesting (and difficult) in a programming sense?
- Some “back to basics” insights
- Some ideas for a *fibred categorical semantics*
- Where this might all lead

Who *is* this man?

Lecturer at Trinity College Dublin

Current research interests centre on context-aware computing

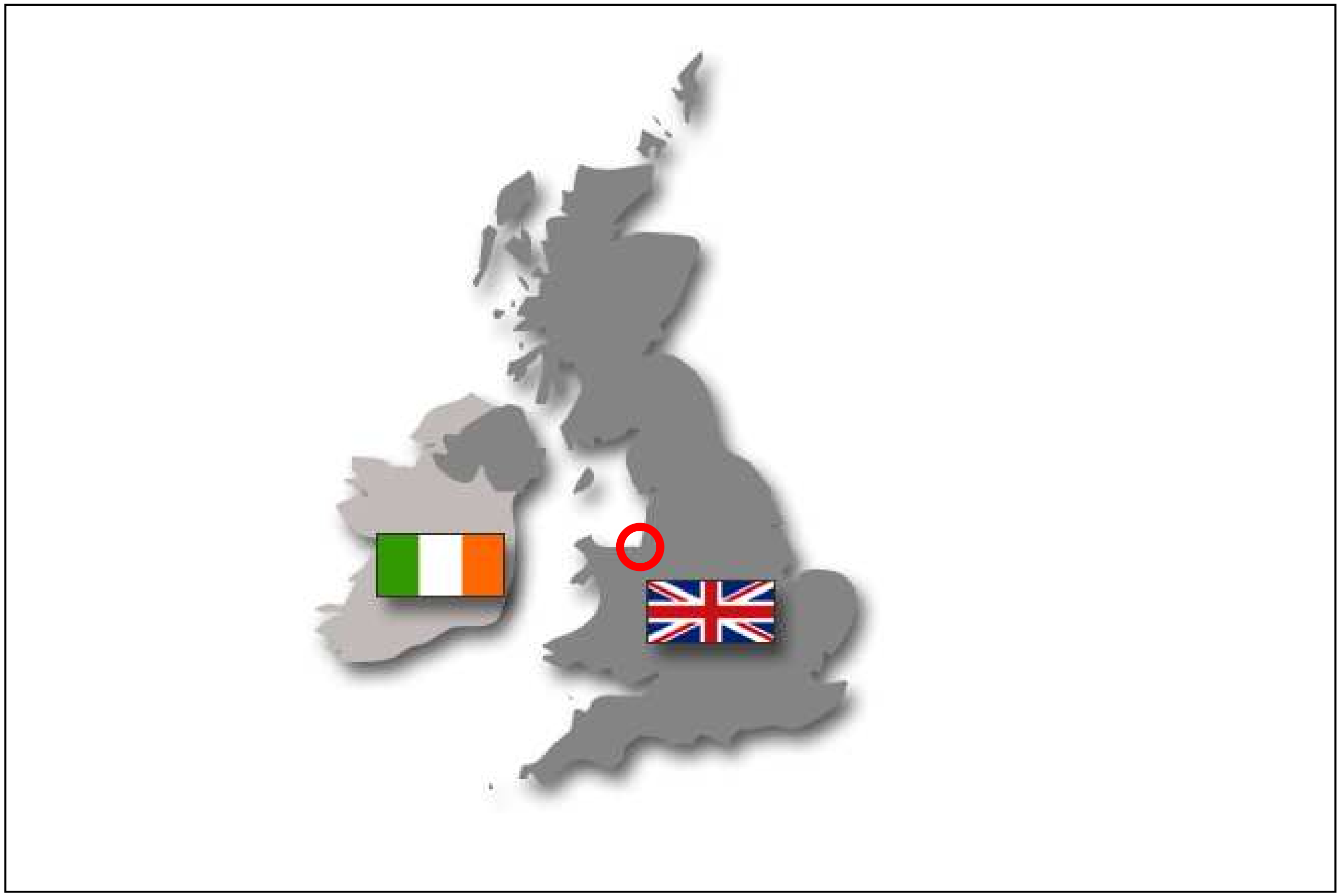
Abstract Models and reasoning (category theory, type theory)

Semi-abstract Composition, architectures

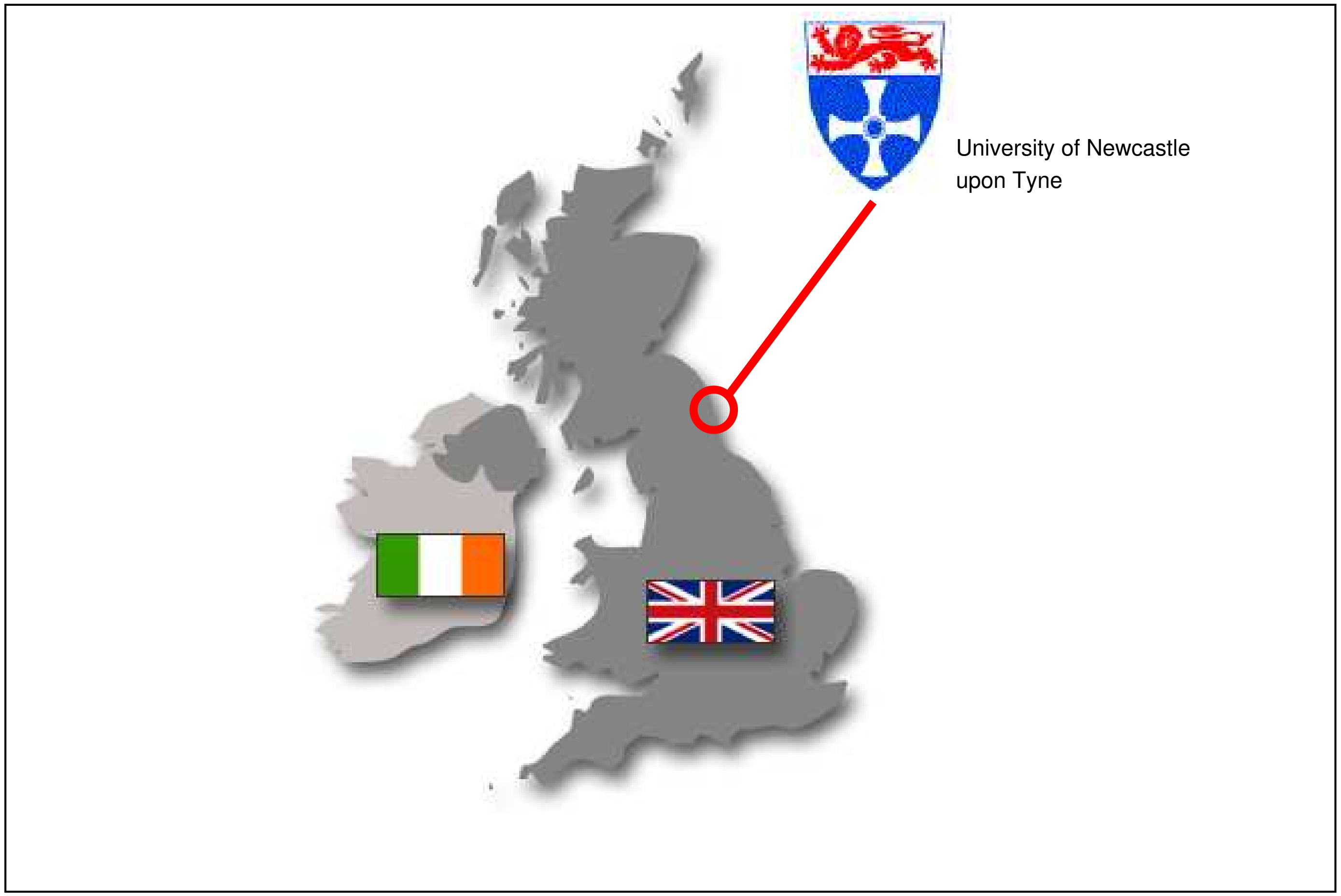
Concrete Programming platforms, smart materials

Spent two years as CEO of a start-up company

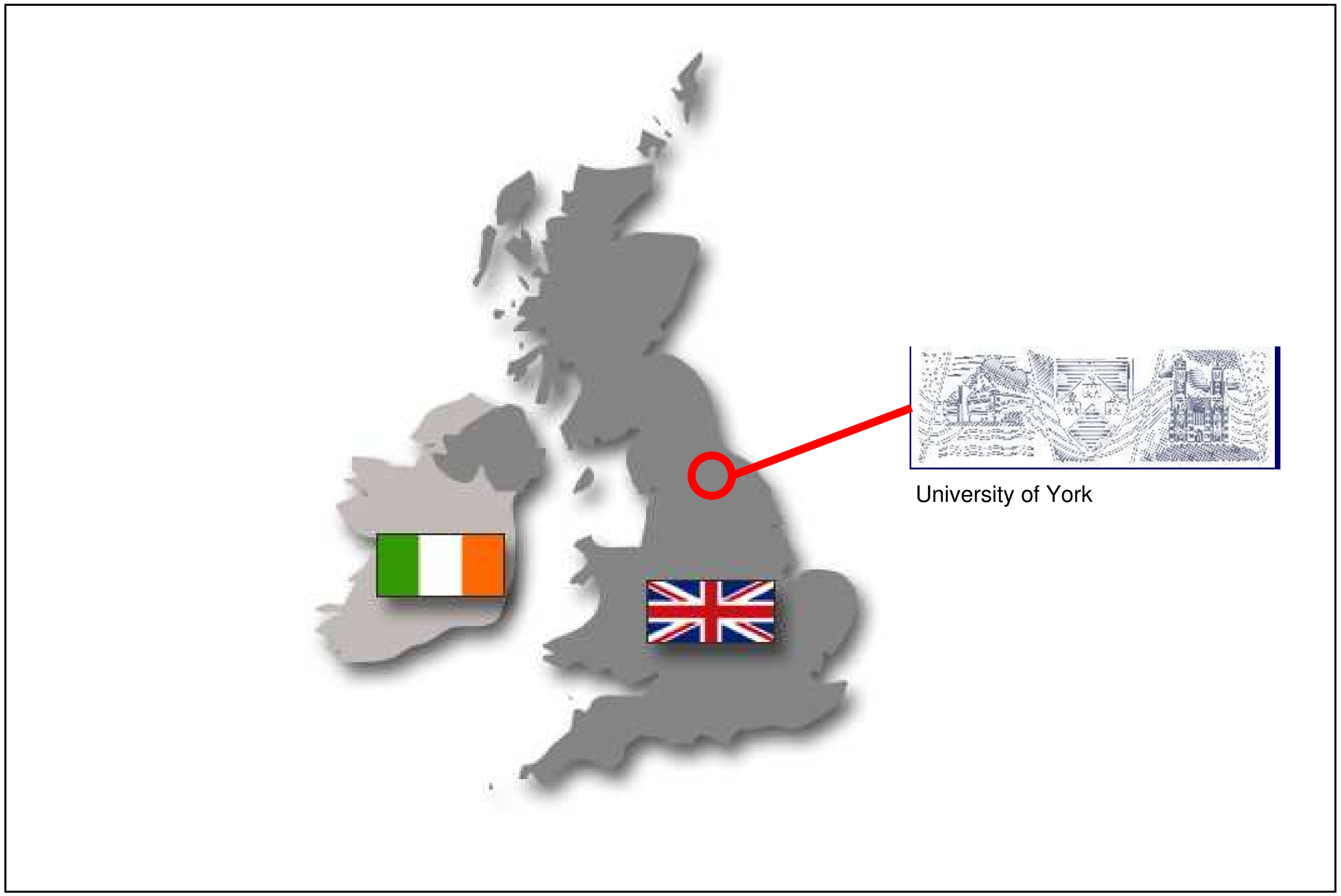
Simon’s career on one slide



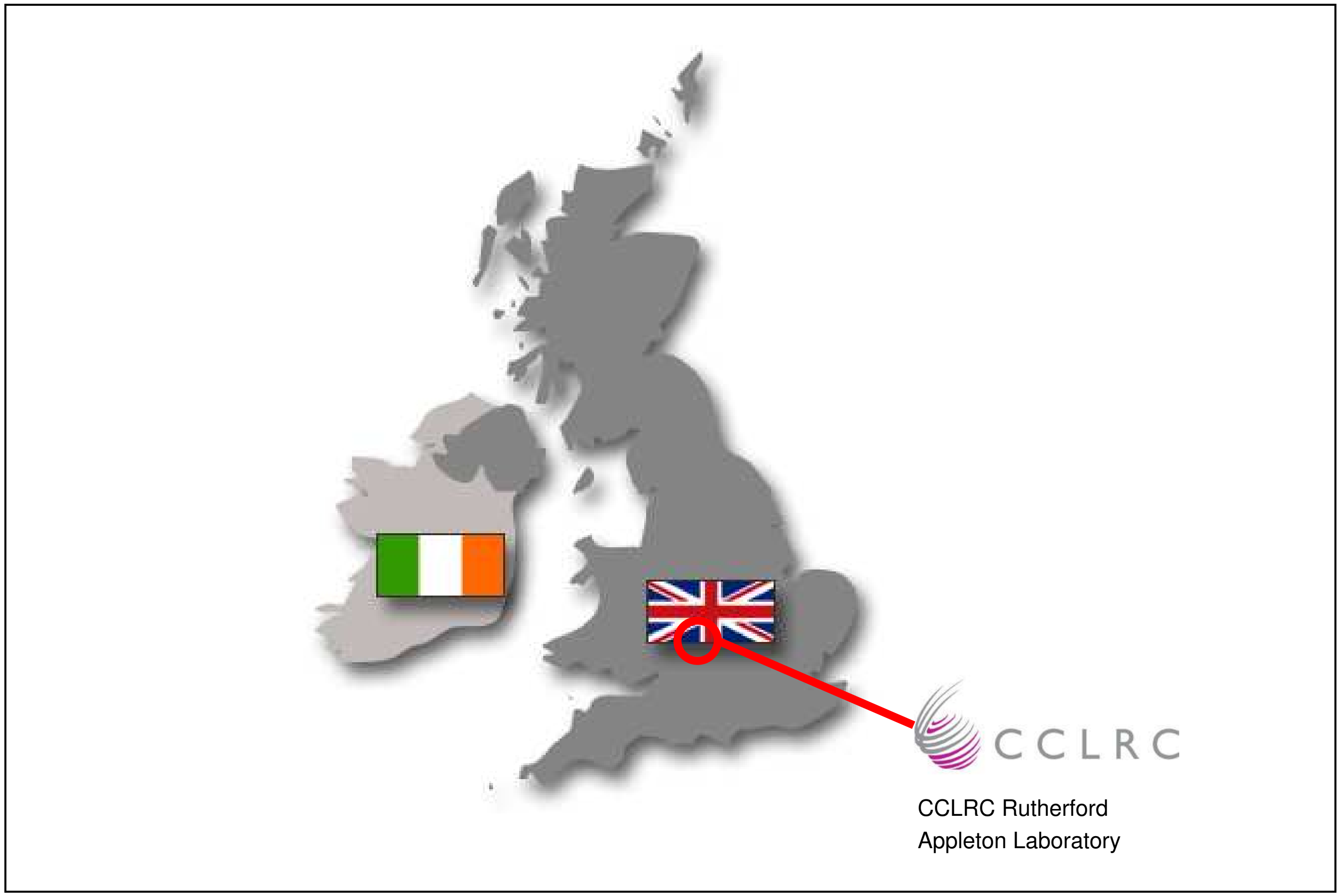
Simon's career on one slide



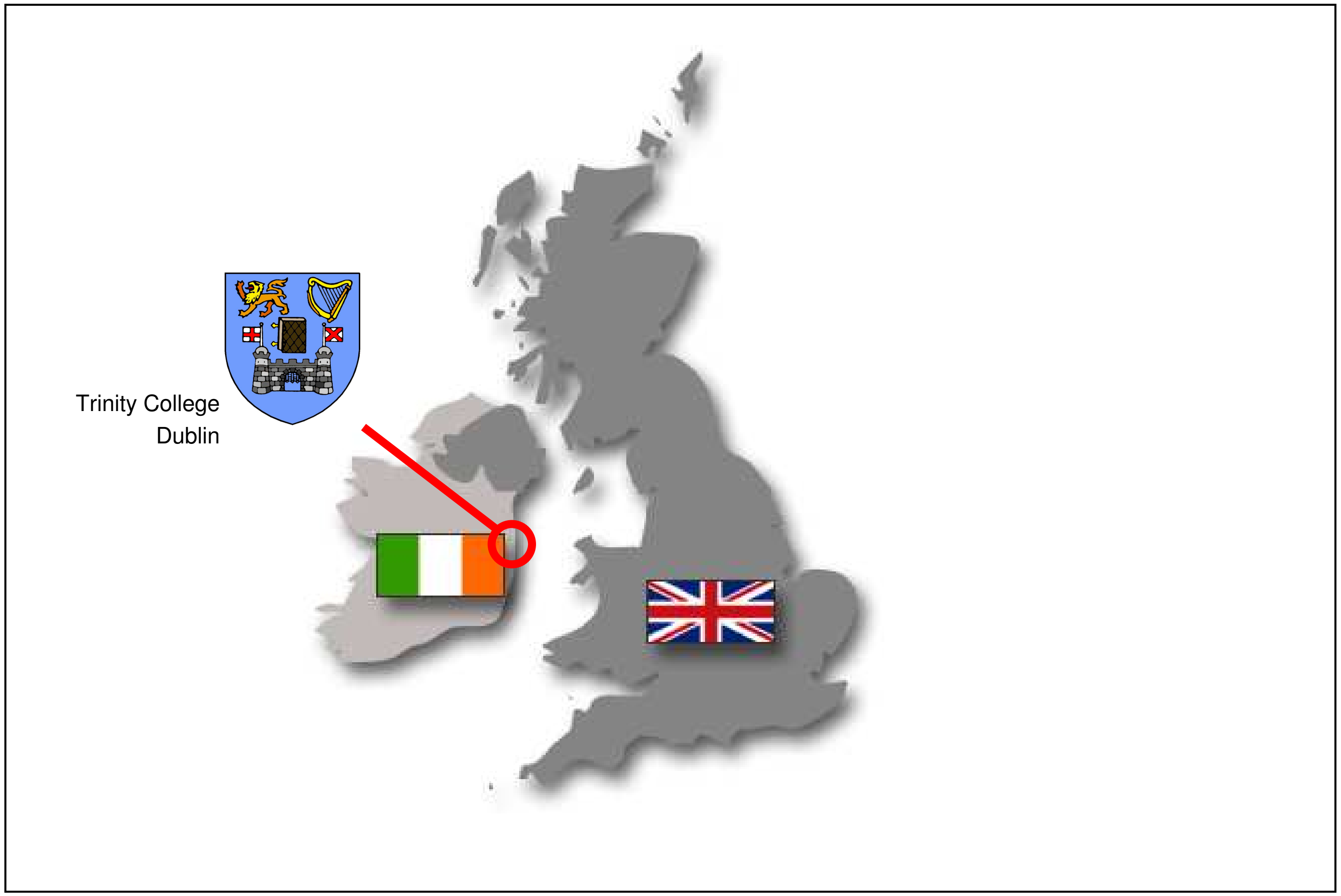
Simon’s career on one slide



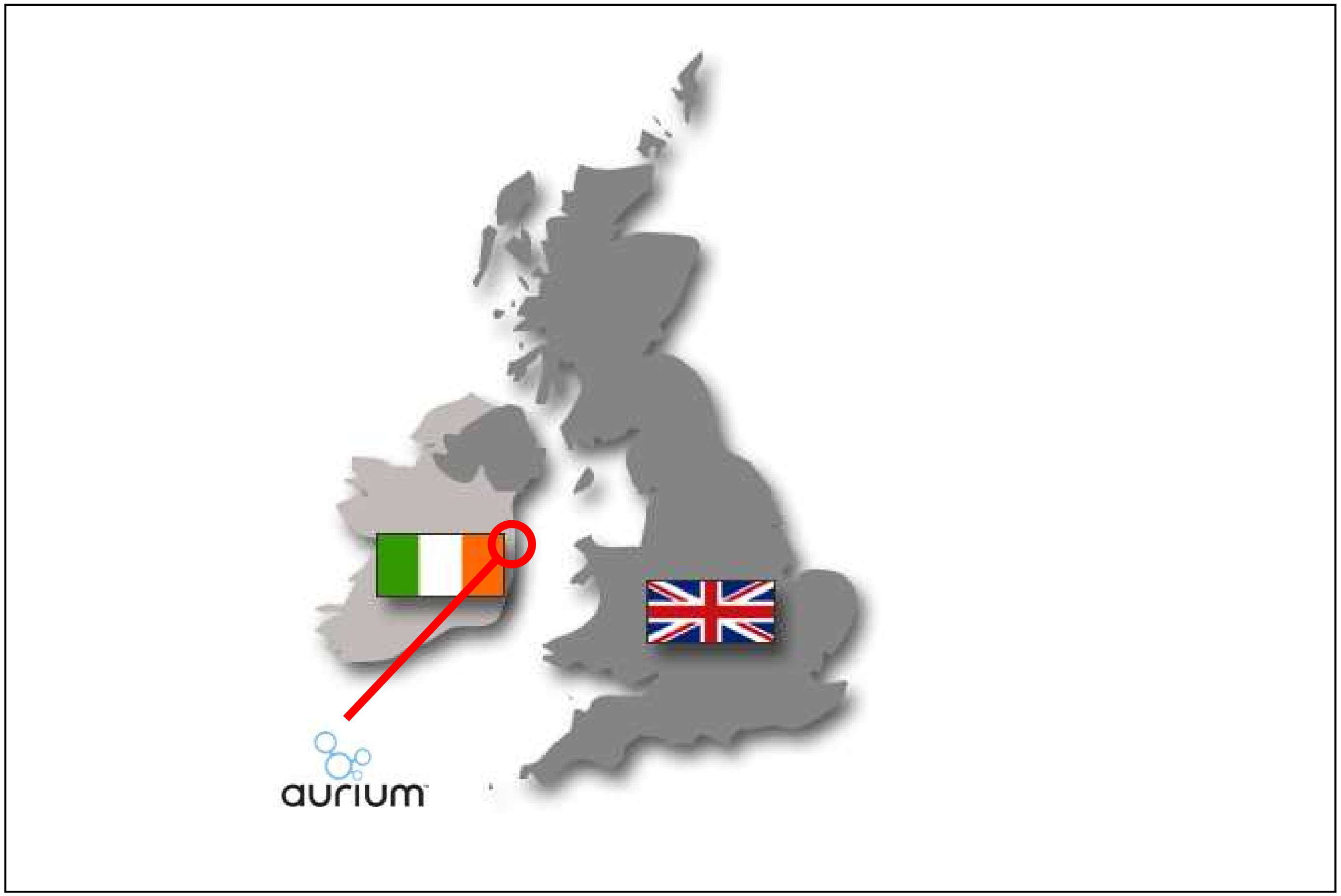
Simon's career on one slide



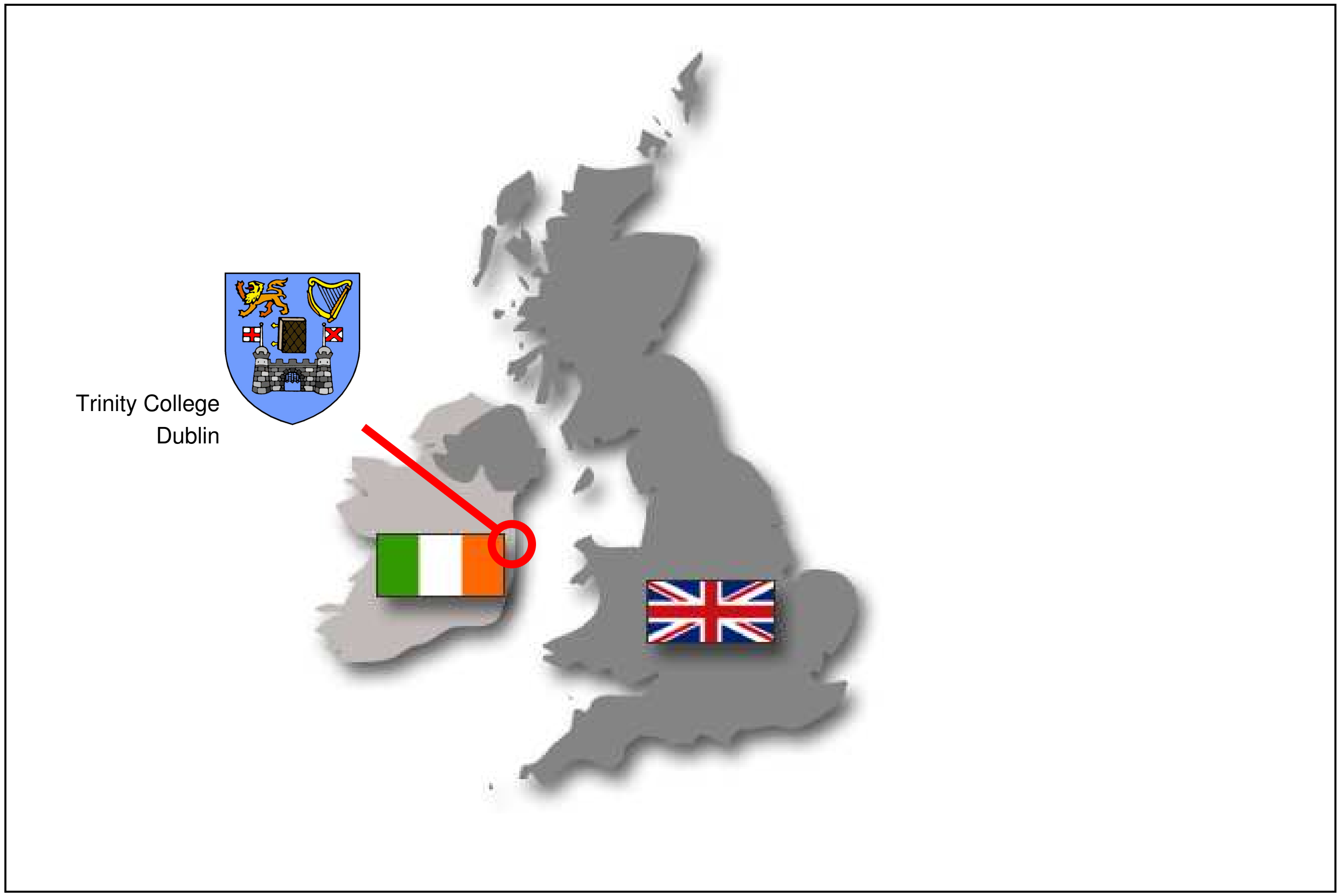
Simon’s career on one slide



Simon’s career on one slide



Simon’s career on one slide



Pervasive computing

Pervasive computing?

Computers that sense and respond to their environment

- Also called *ubiquitous computing* or *ambient systems*

Build and maintain significant amounts of *context* about how the system is being used

Reduce the cognitive load, address more applications and user communities (especially currently marginalised communities)

Characteristics

- Low-value individual services

Characteristics

- Low-value individual services
- ...but high value taken together ($\Rightarrow$ systems not applications)

Characteristics

- Low-value individual services
- ...but high value taken together ($\Rightarrow$ systems not applications)
- Substantial infrastructure generally needed

Characteristics

- Low-value individual services
- ...but high value taken together ($\Rightarrow$ systems not applications)
- Substantial infrastructure generally needed
- ...so high barriers to entry ($\Rightarrow$ low cost of development)

Characteristics

- Low-value individual services
- ...but high value taken together ($\Rightarrow$ systems not applications)
- Substantial infrastructure generally needed
- ...so high barriers to entry ($\Rightarrow$ low cost of development)
- Only a small part of the full solution

Characteristics

- Low-value individual services
- ...but high value taken together ($\Rightarrow$ systems not applications)
- Substantial infrastructure generally needed
- ...so high barriers to entry ($\Rightarrow$ low cost of development)
- Only a small part of the full solution
- ...so need to be easy to integrate ($\Rightarrow$ composition is key)

Layers of environment

Physical	location, movement
User	identity, preferences
Informational	information structures used
Temporal	workflows (explicit or (usually) implicit)
Device	characteristics and limitations
Proximate	system state is “close to” other states
Security	permissions and capabilities
Relational	between layers

Unique features – 1

A significantly different feature set than we're used to

- Existing models address *statics* (design-time)
- ...where what's now important is *dynamics*

Don't focus on what's really characteristic about pervasive computing

- Composition, behavioural variation, leverage on shared platform

Unique features – 2

There is a dynamism implicit in pervasive systems

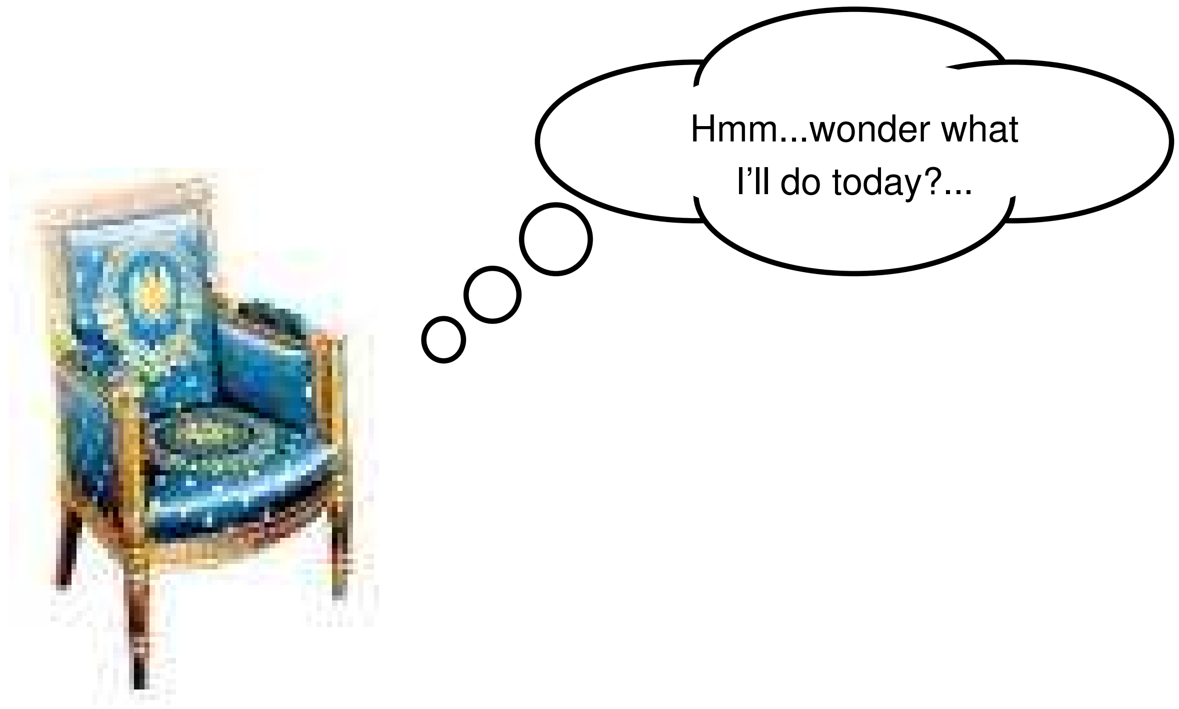
- Changing population of objects/devices/services

Key is not *what* happens, but how what happens *varies with context*

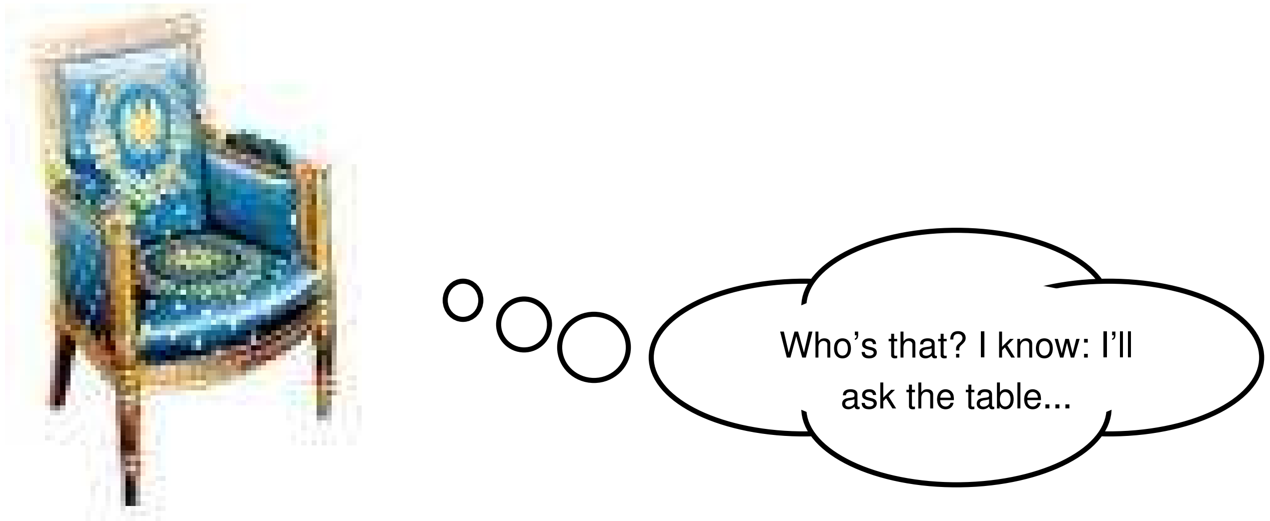
- Usually a bug – but now it's a feature...
- Stability under change
- Relationship with the wider environment

Back to basics

The need for overall control



The need for overall control



It's asking a great deal for intelligent, predictable and suitably subtle behaviour to "just emerge"

Looking for insight

A system's behaviour – even (or perhaps especially) a context-aware one – shouldn't change arbitrarily

- Change will often be "smooth"
- Sharp changes will occur at "sensible" points
- These change points will often be evident in the context

It is this structure that makes systems intelligible – there's an *underlying logic* to the environment for the user to grasp

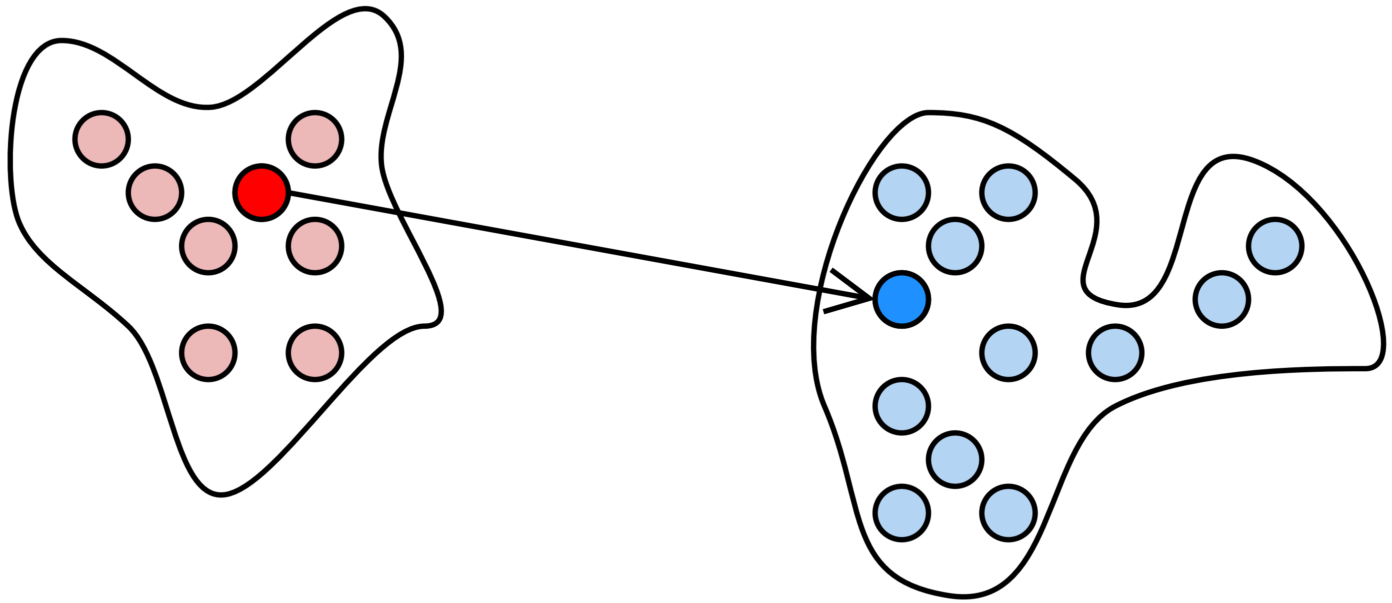
A more declarative approach

Define *what* the system should do – reduce the focus on *how* it should do it

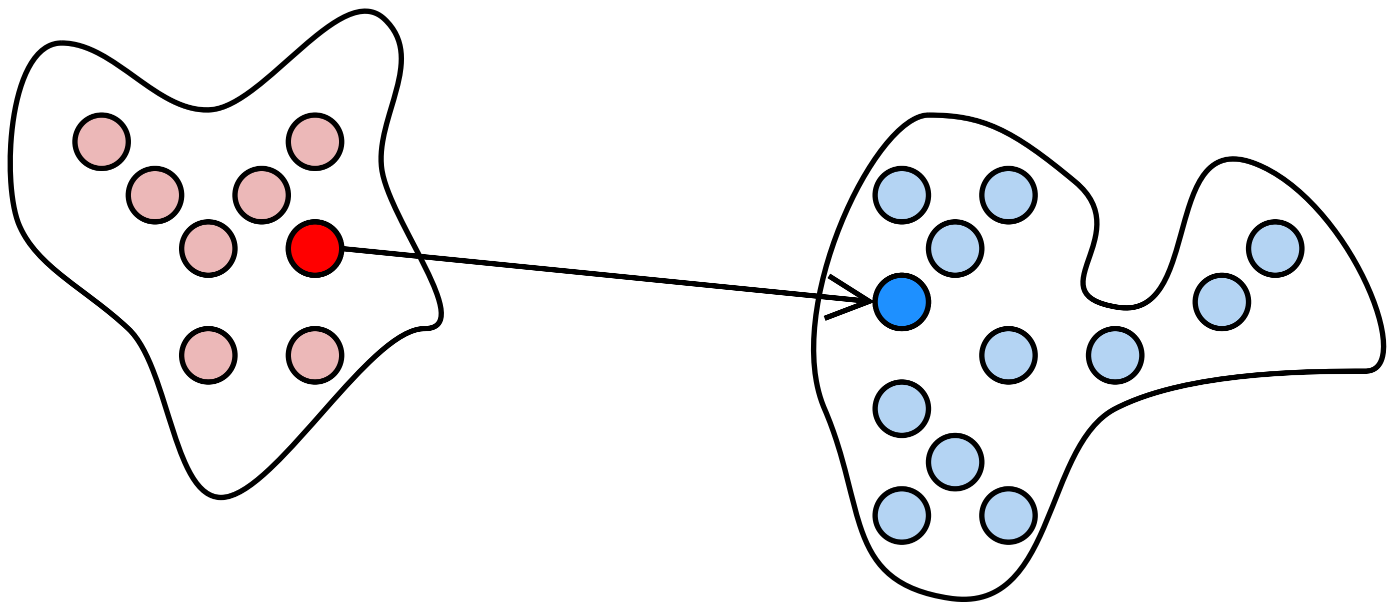
Move away from reacting to events directly and towards describing the “shape” of the responses to context

Use the structure of the context to guide the structure of the possible changes of behaviour the system makes in response to that context

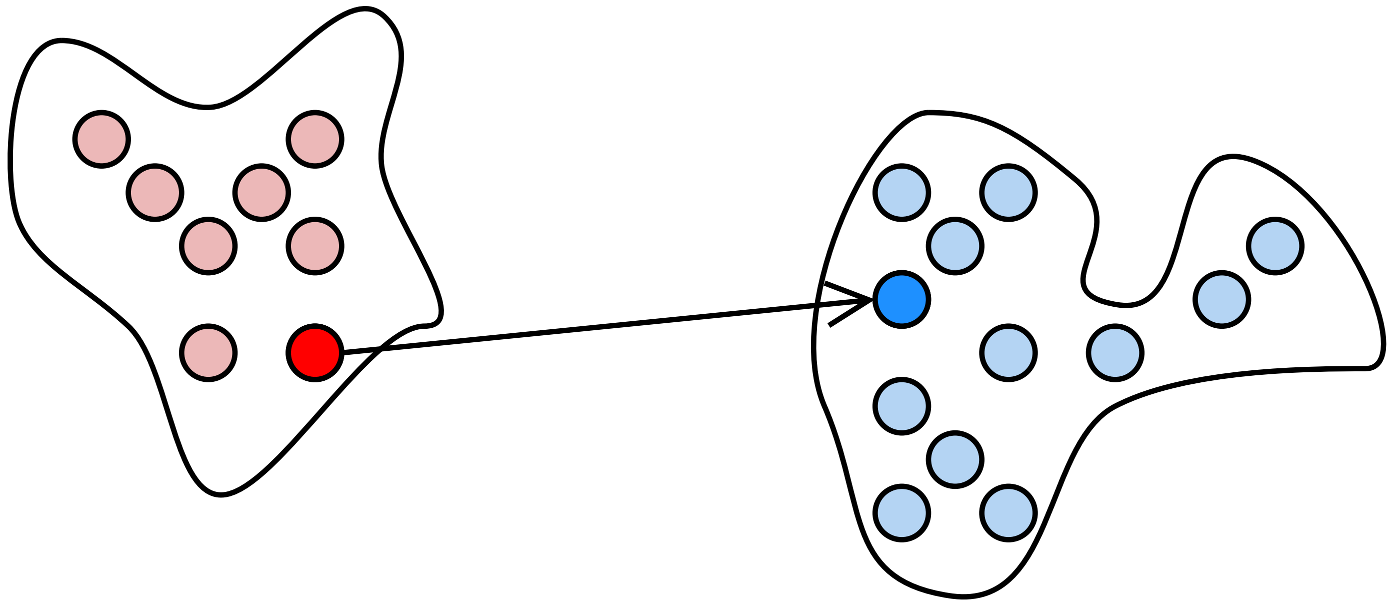
Function follows form



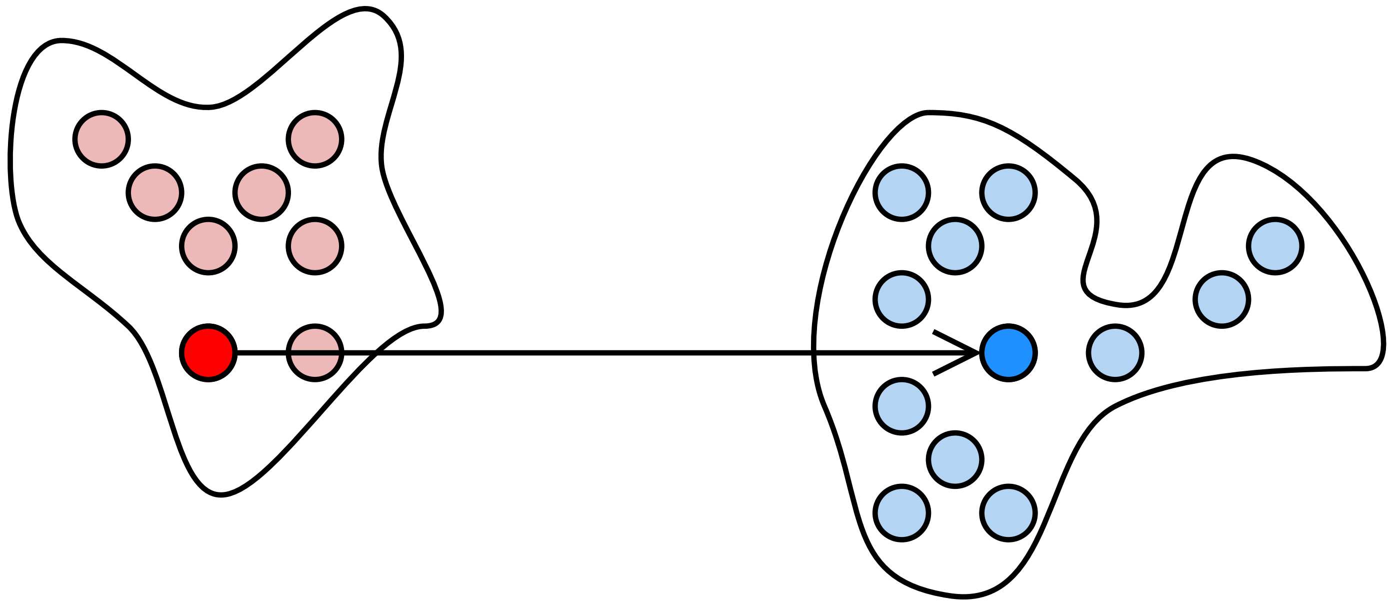
Function follows form



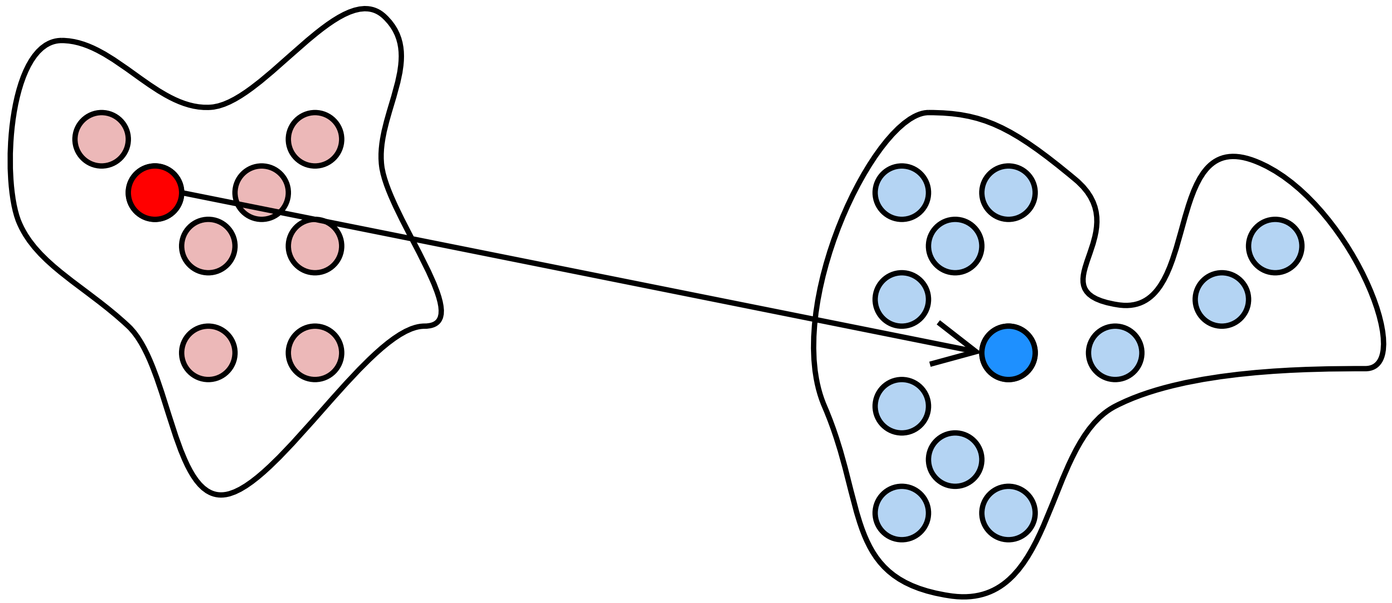
Function follows form



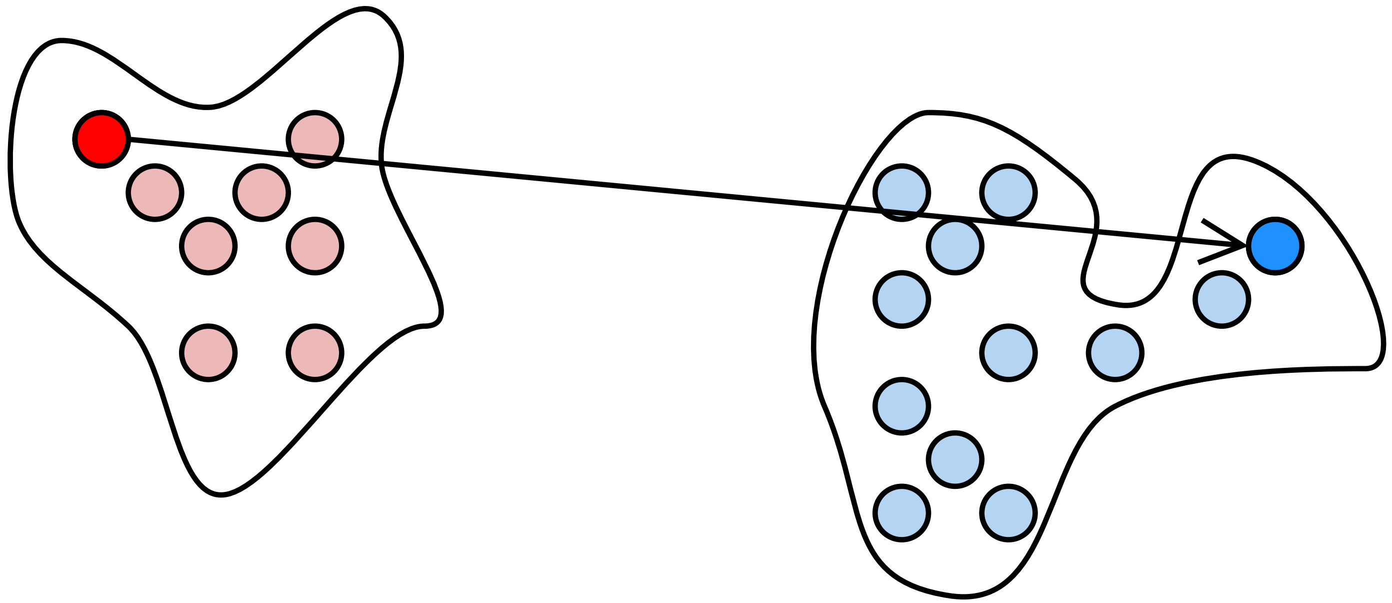
Function follows form



Function follows form



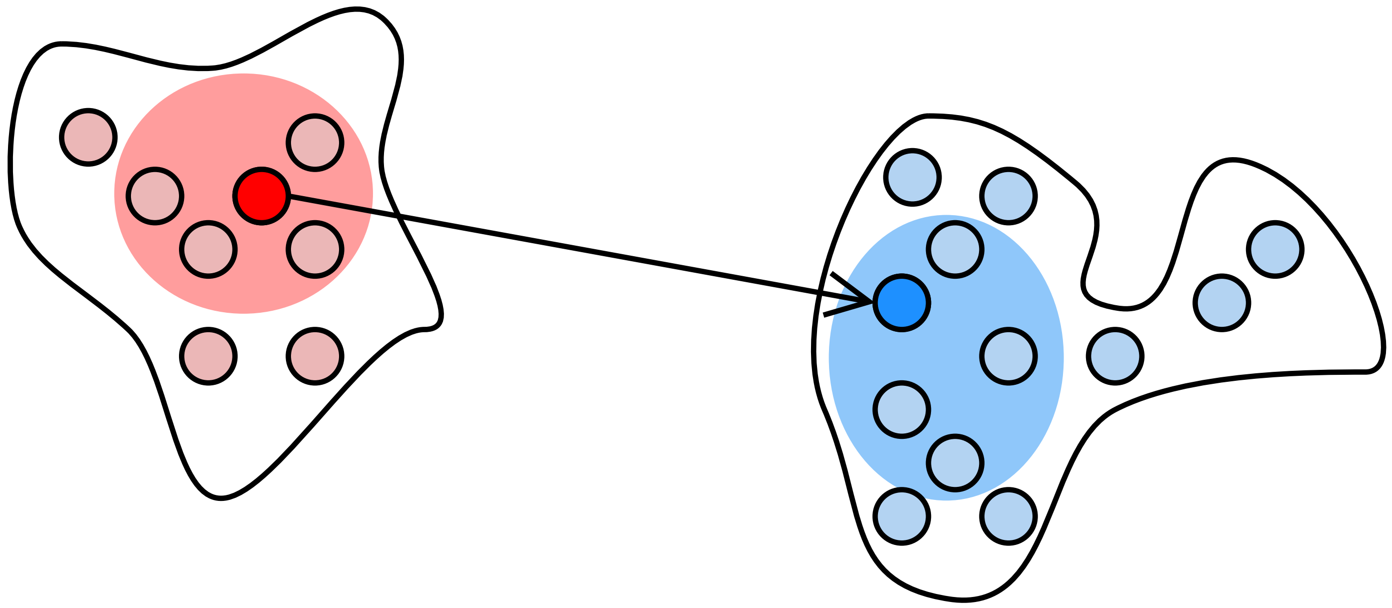
Function follows form



The range tracks the structure of the domain in some sense

Smoothness

Smoothness suggests that points “close together” will behave “alike”

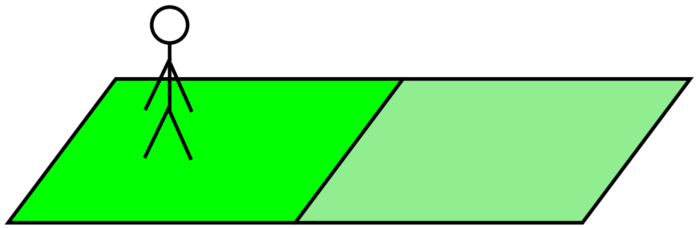


Being in a similar state gives a similar result

For pervasive computing

Capture the structures inherent in the various contextual parameters

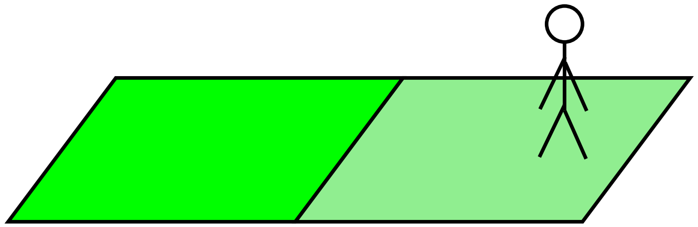
Structures that provide "hooks" for adaptability as far as users are concerned



For pervasive computing

Capture the structures inherent in the various contextual parameters

Structures that provide "hooks" for adaptability as far as users are concerned



Tie behavioural change to these structures

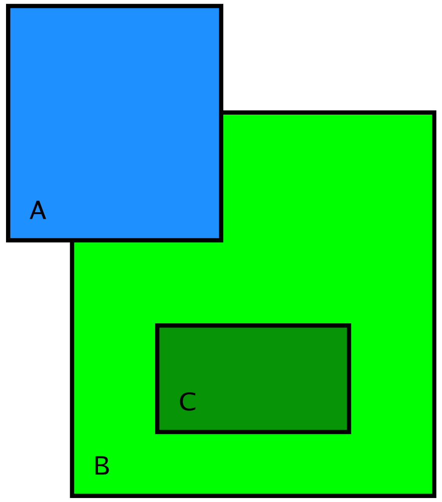
- Variation of system follows that of context
- Some *environmental reason* for a change in behaviour

Structure in context

Some contexts will be "points" – such as people's identities

Others are structured – for example location

- One space inside another
- One space overlapping another, or meeting another



Structure in behaviour

Behaviours can also be structured

Document access

- The largest set of documents someone *ever* sees
- The smallest set they *always* see

Security and permissions

- More or less capability for operations

Towards a semantics

From metaphor to semantics

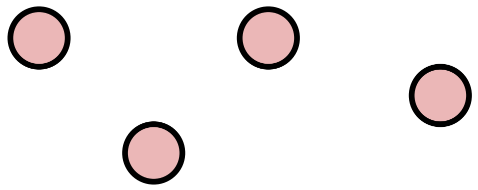
Take the observations above and turn them into a formal basis for analysis and design

- What sorts of structure does context have?
- What sort of mapping do we have to behaviour?
- How can we constrain this relationship?
- What techniques does it open up?

Basic approach

Closed-form expression of a system

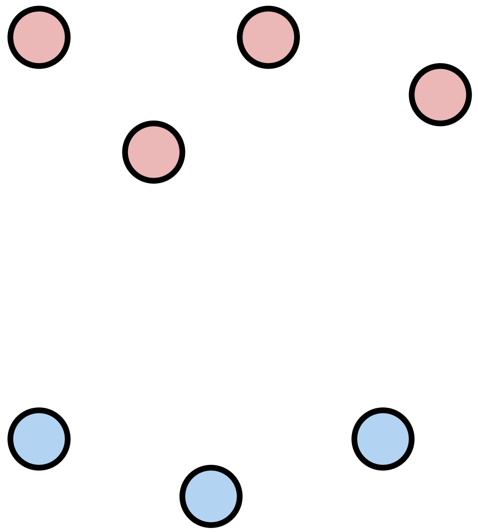
- Take a structured view of context



Basic approach

Closed-form expression of a system

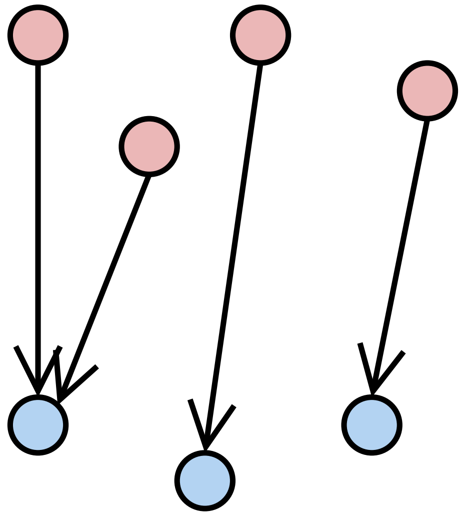
- Take a structured view of context
- ...and a structured view of behaviour



Basic approach

Closed-form expression of a system

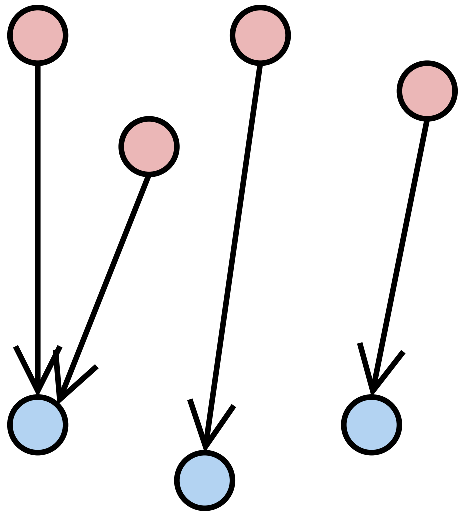
- Take a structured view of context
- ...and a structured view of behaviour
- ...and show how one relates to the other



Basic approach

Closed-form expression of a system

- Take a structured view of context
- ...and a structured view of behaviour
- ...and show how one relates to the other



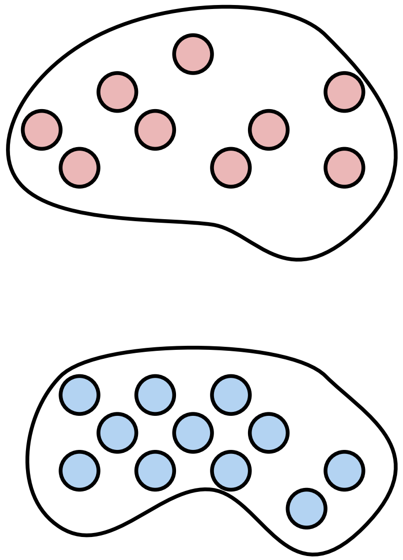
Focus on what's important – variation – and de-emphasise the details

Fibres – 1

A mapping may take several objects to a single other

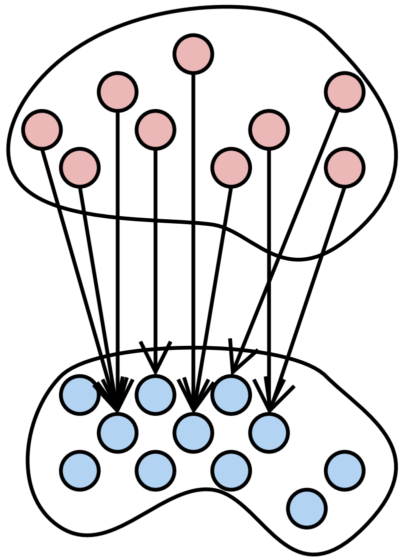
Fibres – 1

A mapping may take several objects to a single other



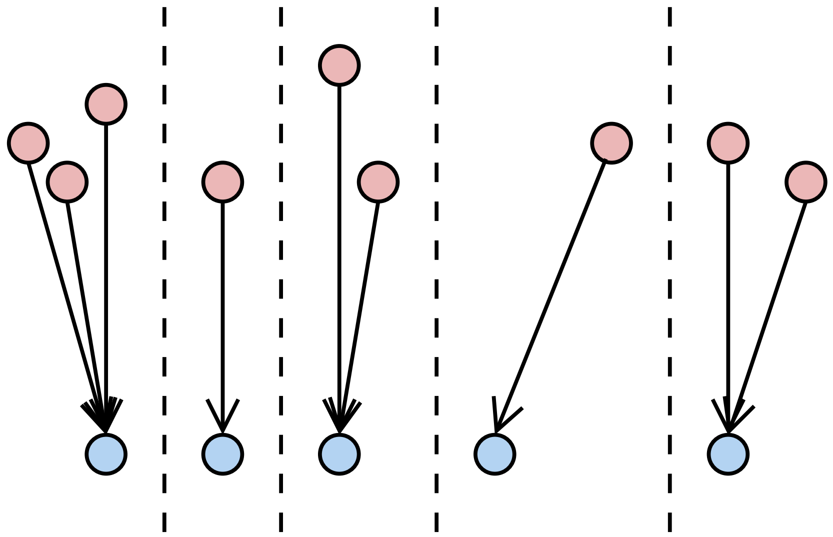
Fibres – 1

A mapping may take several objects to a single other



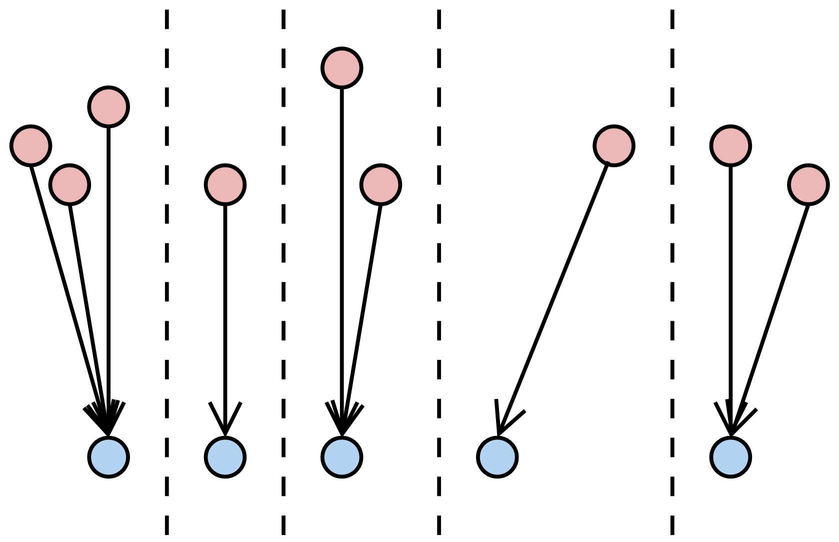
Fibres – 1

A mapping may take several objects to a single other



Fibres – 1

A mapping may take several objects to a single other



The collection of such objects is referred to as the *fibre* of the mapping *over* an object in the range

Structure in context again

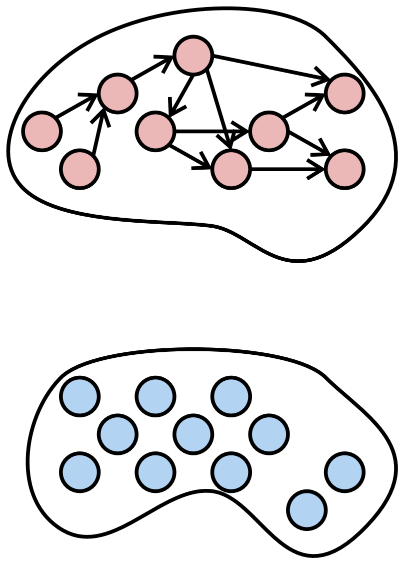
The structure of a context provides the “visible” hooks for changes in behaviour

- The user will (or should) see a change in the environment
- A logical place for behaviour to change

This structure appears semantically as structure in the contextual layer that constrains or conditions the mapping

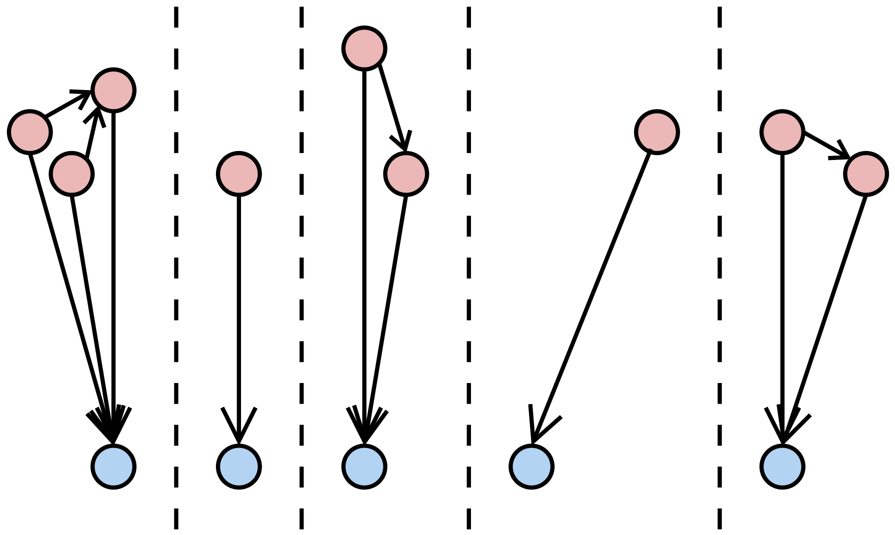
Fibres – 2

Reflect structure inside the fibres



Fibres – 2

Reflect structure inside the fibres



Each fibre is now a collection with structure, reflecting the structure of the elements “over” a particular range object

Common structures

One often finds behaviours forming a partial order

We may then have a “least” behaviour

- We refer to this as a *core* behaviour
- A behaviour appearing in *all* contexts – although the behaviour of a particular context may be bigger

Dually, we may have a “largest” behaviour

- A behavioural *extent*
- A behaviour that may be reduced in context

What this means

Moving context *inside* a fibre doesn't change behaviour

- The contexts in which the system "behaves the same"

Moving context *between* fibres causes change

If we constrain how these inter-fibre transitions occur, we can structure behavioural change to follow the underlying logic of the context

Going formal

We can translate all the diagrams in this section directly into fibred category theory

- Each context becomes a category, possibly with internal structure
- Each behaviour becomes a category too
- Variation modeling with a *contextualising functor*

All the tools of category theory are then available to study contextual systems

We're also looking at a simpler version within a single category

Categorical formulation – 1

Context modeled as a category $\mathbb{C}$; behaviour as a category $\mathbb{B}$

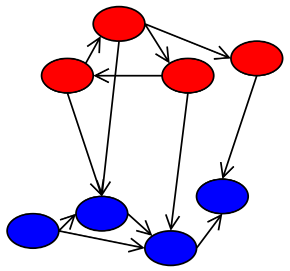
Behaviour is specified using a *contextualising functor* $\mathbf{p}$

Form fibres $\left(\begin{tikzcd}\mathbb{C} \arrow[d, "\mathbf{p}"'] \\ \mathbb{B}\end{tikzcd}\right)$ of context over behaviours

A categorical statement of what it means for a function (arrow) to vary with a context

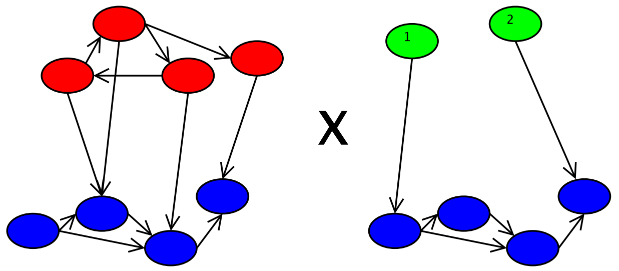
Categorical formulation – 2

Standard categorical structures provide composition



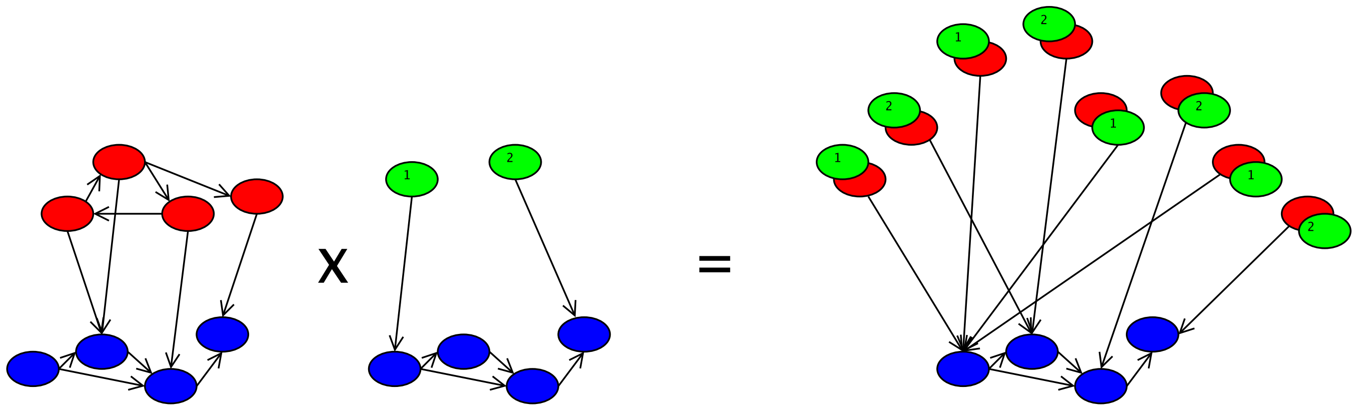
Categorical formulation – 2

Standard categorical structures provide composition



Categorical formulation – 2

Standard categorical structures provide composition

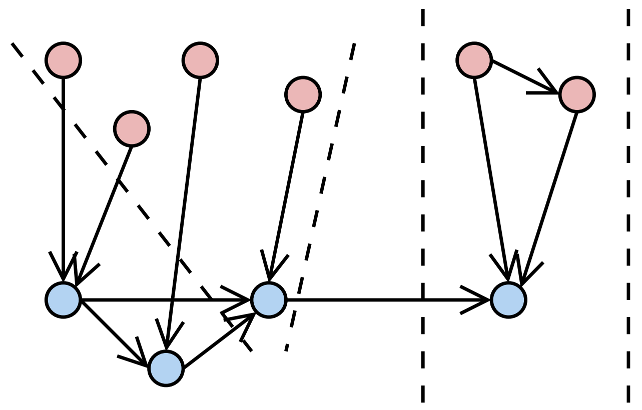


May be other fine structure, for example where one of the product elements doesn't affect behaviour

Categorical formulation – 3

Some structures appear that are more general than standard fibred category theory

- "Fibre" of a structured region of the behaviour
- Details vary while maintaining core/extent



Pros and cons

Both type theory and logic have categorical formulations

- Use tools for either to study pervasive computing
- Insights can be applied directly to languages

Not necessarily close to implementation

- Some things may be hard to model implementationally

How this might fit together as a system

Analysis

Take a system and re-cast it using categories and fibres

Make sure the inter-fibre transitions occur as expected

See what context is actually needed

- Which product terms actually induce behavioural changes

Some features appear categorically

- For example extents as relative initial objects

Design

Describe the system in terms of structured context and behaviour, with functors describing the variations

Constrain functors so that inter-fibre transitions occur at externally graspable points

- Characterise these points as categorical structures
- Abstract descriptions of (un)desirable properties for a pervasive system to have

Open questions

- What sorts of systems can we characterise this way?

Open questions

- What sorts of systems can we characterise this way?
- What is the type theory?

Open questions

- What sorts of systems can we characterise this way?
- What is the type theory?
- What is the internal logic?

Open questions

- What sorts of systems can we characterise this way?
- What is the type theory?
- What is the internal logic?
- What sorts of programming structures underlie the approach?

Open questions

- What sorts of systems can we characterise this way?
- What is the type theory?
- What is the internal logic?
- What sorts of programming structures underlie the approach?

Categorical products and sums map to interesting structures: are there any novel structures?

Can the phrase "underlying logic of the context" be more than just a soundbite?

Conclusions (such as there are)

- Behavioural variation follows contextual structure – although in quite complicated ways

Conclusions (such as there are)

- Behavioural variation follows contextual structure – although in quite complicated ways
- We have made a start in modeling these variations using notions from category theory

Conclusions (such as there are)

- Behavioural variation follows contextual structure – although in quite complicated ways
- We have made a start in modeling these variations using notions from category theory
- Categorical structures can improve analysis and may suggest new approaches

Conclusions (such as there are)

- Behavioural variation follows contextual structure – although in quite complicated ways
- We have made a start in modeling these variations using notions from category theory
- Categorical structures can improve analysis and may suggest new approaches
- Capture the underlying logic of a context and how it controls system behaviour