A simple semantic model of adaptive pervasive systems

Simon Dobson and Juan Ye

Systems Research Group
School of Computer Science and Informatics
UCD Dublin Belfield, Dublin 4, Ireland

http://www.ucd.ie/csi

{simon.dobson,juan.ye}@ucd.ie

Overview

We are seeing an increasing interest in building *adaptive* computing systems of various kinds

- Respond to mobility and the increase in wireless devices
- Self-manage and -configure to improve plug-and-play construction
- Personalise interactions to individuals' needs

A severe increase in software complexity

- Hard to grasp what systems will – or should – do in a given situation
- Hard to compose components, too much "big bang" development

We need new models of what it means to be a system

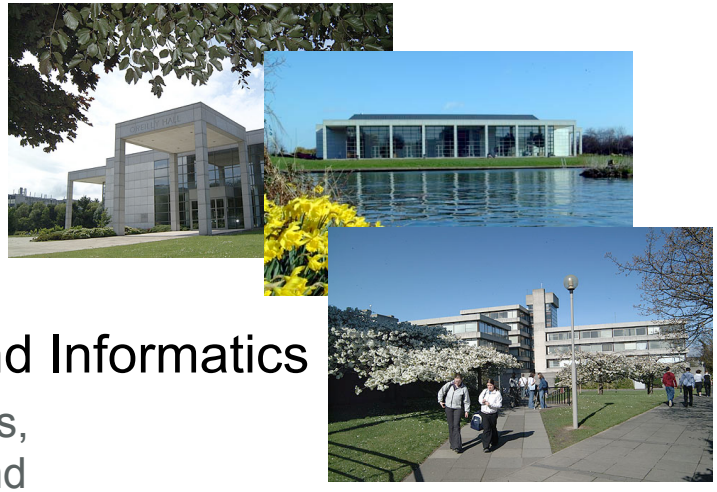
- Design, analysis, development based on appropriate formal models
- Improve our ability to tackle the next generation of big systems

My aim: to introduce the intuitive ideas and basic mathematics of a semantics for adaptive computing

UCD CSI SRG

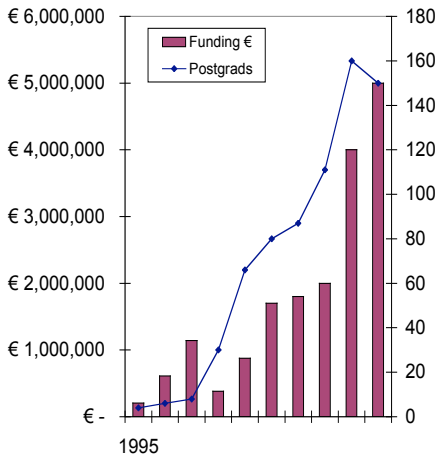
UCD is Ireland's largest university

- 22,000 students of whom about 6000 are on MSc/PhD courses
- Dedicated campus in the south suburbs of Dublin



The School of Computer Science and Informatics

- 30 academics, around 150 PhD students, around 150 international papers/year and €4M/year research funding
- Major research themes in intelligent systems, autonomic and pervasive computing, distributed systems, computational science



Systems Research Group

- Four academic staff including one full professor
- Extensive facilities including the largest sensor and location-aware systems test-bed in Ireland

Adaptive computing

A computer system whose behaviour adapts to its surroundings

- Location, user, role, people being together, preferences, …
- Network load, isochrony constraints, wireless sensing, …

How might we tackle these issues?

- Take an existing application
- …put it together with some sensors
- …decide how the sensors affect the behaviour we want to see
- …add `if` statements *etc* so that the behaviour changes in response to events
- …see what happens

How do we know what sensors we'll have? What happens when they fail or change?

Do we know *all* the places behaviour should change?

Do we even know what each individual sensor is telling is about the *user*?

How will we know when we've got it right?

Major benefits, major challenges

Why adaptivity is great for the user/designer

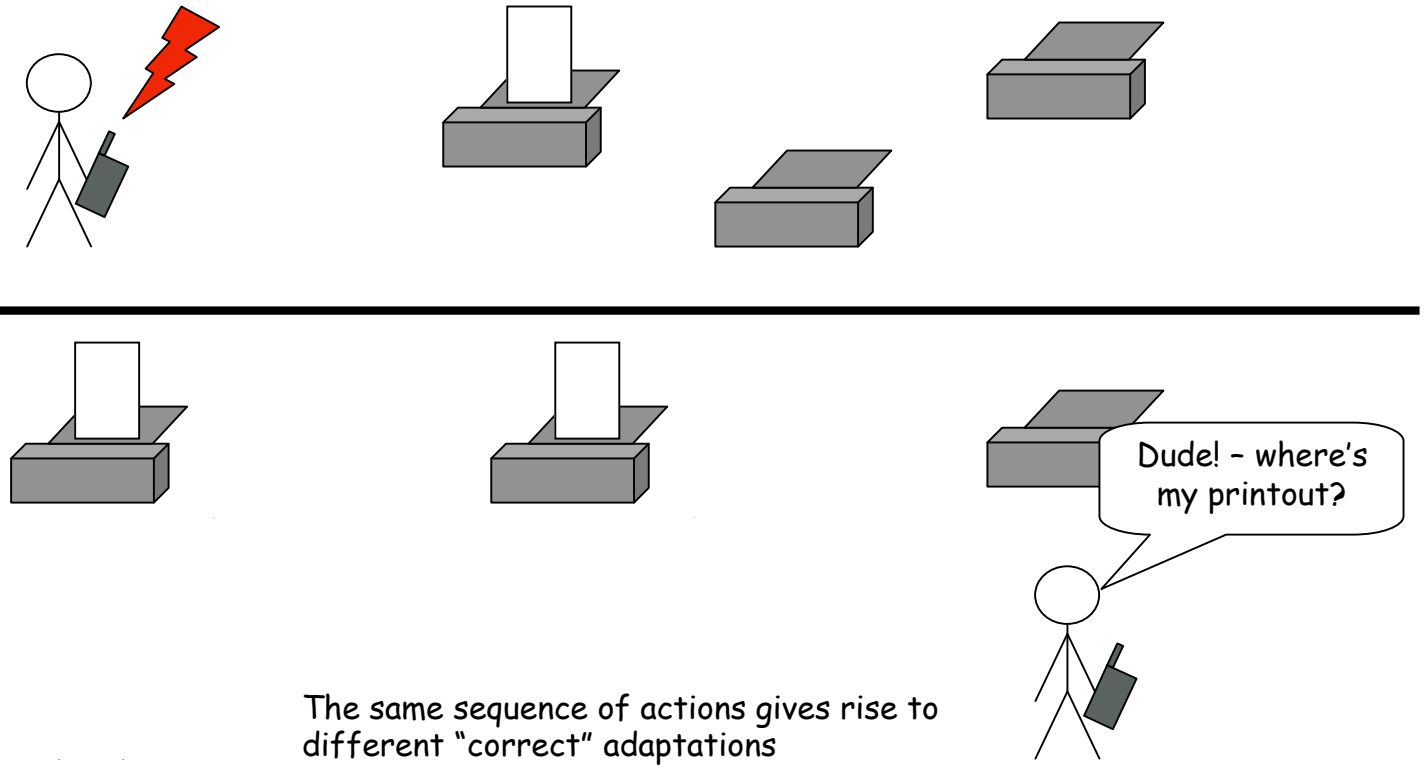
- ✓Offer consistent (and hopefully better) levels of service
- ✓Even-out changes from the environment
- ✓Reduce explicit user interaction, more intuitive and predictive behaviour
- ✓Reach out to new applications, marginalised communities, …

Why adaptivity sucks for the designer/programmer

- ×Getting an application right *once* is hard enough
- ×…and now we have to get all the adaptive behaviours right too
- ×…*and* ensure that we select the right adaptation at the right time
- ×…in the face of low-quality, inaccurate and often completely misleading information about user needs and environment

Example

The “Dude! – where’s my printout?” problem…



The same sequence of actions gives rise to different “correct” adaptations

What's happening

We have a sequence of action that a user performs

- The usual model of an interactive system
- Constraints on the order of actions, some may be disallowed, …

We can automate some of these actions – the selection of the printer – based on sensed location

- Common example from the literature

But the choice turns out not to be stable under motion on the part of the user

- User moves to places where the choice changes
- …so making the right choice at each individual point is *wrong* overall from the point of view of the complete interaction

Scrutability

The there's a term for this in usability engineering: *scrutability*

- The ability of a user to form a coherent mental model of how a system behaves simply from observing it over time
- The nearest printer application is essentially *inscrutable* – you'll never be able to predict what it'll do

We can also draw a more formal distinction between the two kinds of behaviour

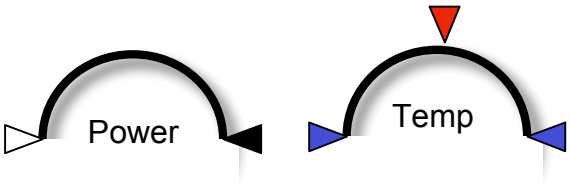
- A behaviour is *point correct* when it does "the right thing" in a given context
- A behaviour is *process correct* when it does the right thing over the length of an interaction
- A sequence of point-correct behaviours may not be process-correct

Another example: my shower

How does this device work?

Two knobs, both of which rotate, not labeled

Two knobs, both of which rotate, labeled "hot" and "cold"



The need for a semantic model

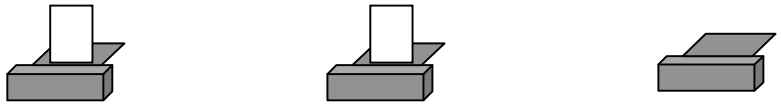
These are all good arguments for a formal semantic model

- Separate from the details of an individual computation or system architecture
- Designed to be mathematically tractable to aid analysis
- Able to drive design to improve system properties *a priori*

Improve our ability to manage the complexities of adaptation over the long term

- Next-generation systems
- Composition and dynamism

…and its as well to admit that our current state-of-the-art can't quite address these processes completely yet



Capture the process, relate the adaptations we make to more than just the sum of the sensor observations we make

Semantics in the small

There are many demantic theories

- Denotational, operational, game, action, …
- Process calculi, Petri nets, dataflow, …

Some even target mobile computing, which has some simple adaptive requirements

- Cardelli and Gordon's ambient calculus
- Jensen and Milner's bigraphs

All focus on the details of a process: specifying the complete details of everything

- …and often can't abstract away from the low-level synchronisation
- Lose the interesting bits in the noise of the detail

Semantics in the large

We have proposed that it's better to work on a semantics in the large

- Focus on the "shape" of adaptation rather than the detail
- Allow (assume) the details are handled by some other means

Guarantee large-scale properties

- …at the expense of fine guarantees

Dobson and Nixon. More principled design of pervasive computing systems. Proceedings of EHCI/DSVIS. 2004.

Three main components

- The possible *behaviours* the system may exhibit
- A model of the *environment* or context the system is to adapt to
- A *mapping* by which environmental changes drive adaptation

Category theory

We've chosen to work within a framework of category theory

- Generalise away from sets
- Strong notions of typing, composition and common structure
- Links with other potentially useful areas such a topology

In what follows we'll work exclusively in the category of sets and total functions

- Objects are sets; morphisms are total functions

Formalising interactive behaviour

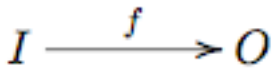
We need a model that can capture the interactive behaviour, the context that affects it, and the structures involved

- A complete field in itself, of course…

However, for our purposes we can abstract away from all the detail of exactly *what* happens and concern ourselves with how what happens *changes*

- An object representing the users' actions
- An object representing the visible behaviour
- A mapping from one to the other

User actions are taken from a pool of possibilities – we say nothing here about sequencing etc



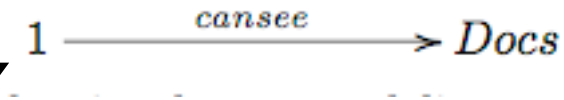
Examples

Input is a URL web query; behaviour is a document; map is what document is served

Every URI in this set

…maps to a document from this set

Input is a single element (“give me the information”); behaviour is a set of documents; map is what documents can be seen



The “terminal” object that carries no information

This second example is a simplification of the first that really allows a strong focus on the variation in behaviour

Contextualisation

We want the behaviour we see to change according to the context in which the interaction occurs

Represent context as an object and combine it with the users' actions

$$C \times I \xrightarrow{f'} O$$

We don't widen the choice of possible behaviours, although the new arrow may "choose" some that weren't chosen before

Might choose from several different contexts

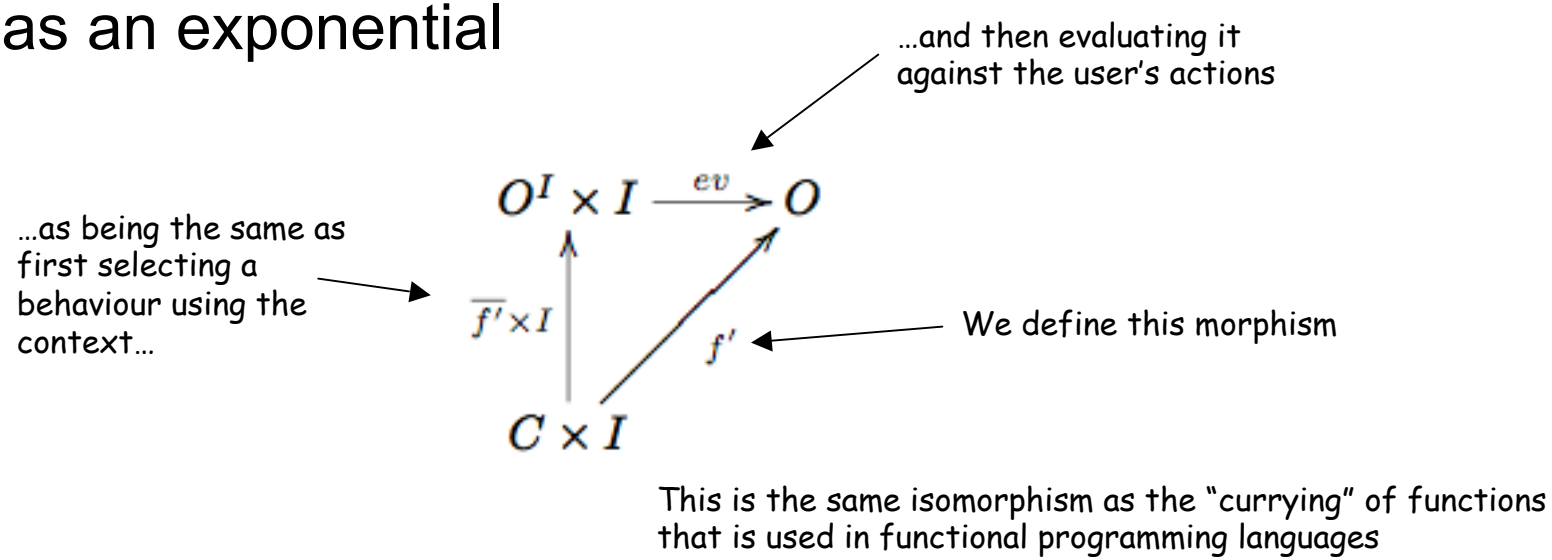
- Location, identity, …
- Different kinds of sensor data, results of inference, sensor fusion, …

How does behavour change?

We can view this situation in several ways

- As a map from two inputs to a behaviour
- As a map from an input to a map from the other input to a behaviour

These two views are isomorphic, and can be modeled as an exponential

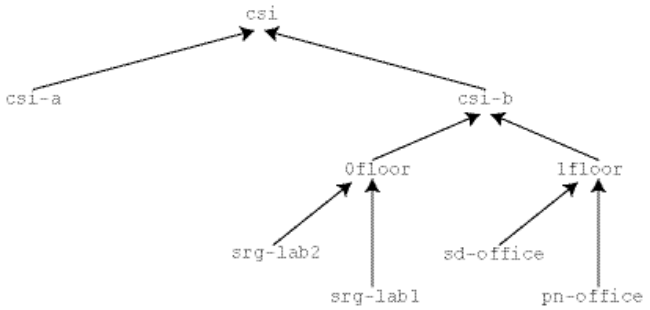


This is the same isomorphism as the "currying" of functions that is used in functional programming languages

Context has structure

Context isn't without structure: a good example is a user's location

- One model is to use named spaces
- But the spaces lie in a relationship with each other, for example a hierarchy in which spaces can live inside another



It is this structure that provides scrutability

- Users perceive the structure in context – people have a strong notion of space, for example
- If a system adapts, it's reasonable for them to expect the variation to follow the structure they perceive in the environment

Modeling structure

Structure can be captured generally as a relation on the context

A relation is a sub-set of all the possible pairs taken from a set, with aRb iff (a, b) ∈ R

$$r : \underline{R} \rightarrowtail R \times R$$

Often find the relations are less than fully general

- Typically find semi- or full lattices, for which monotone maps preserve the important structures

A similar approach can be used for behaviour

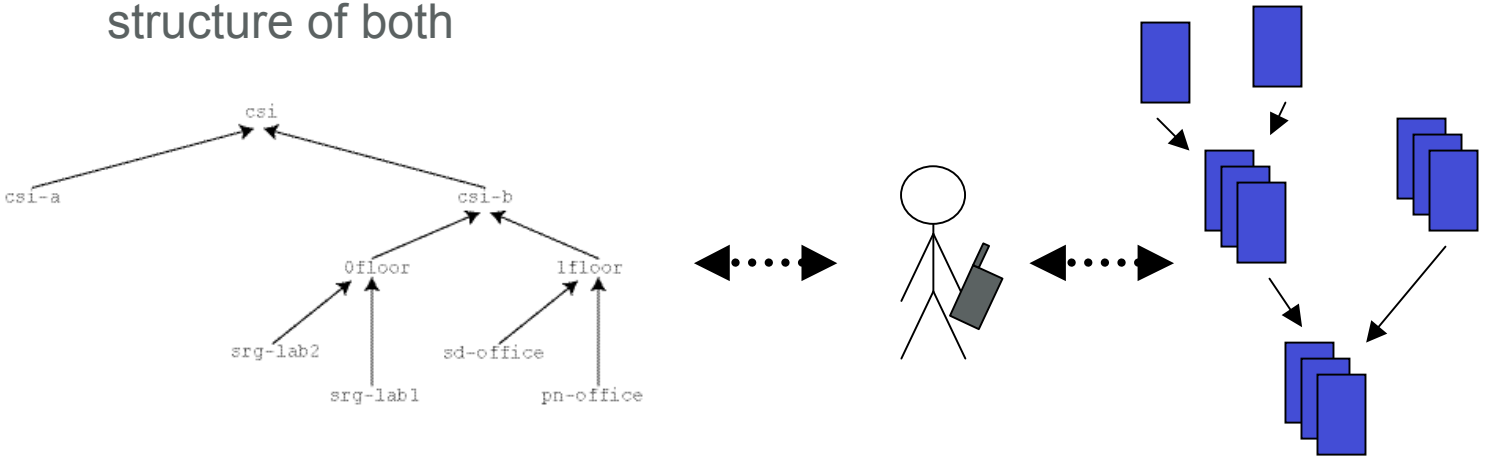
- Again find lattices – especially when taking about information delivery, which often can be modeled as sets of documents ordered by inclusion

Since all relations have inverses, we can use either as appropriate

Leveraging structures

So we can now see intuitively what we want to study formally

- There is structure in context and structure in behaviour, and adaptation needs to map one to the other in a way the respects the structure of both



The essential semantic problem is to capture this in a mathematically tractable way

Contextualising behaviour

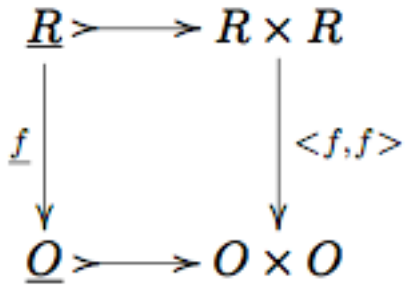
Contextualising a behaviour means describing its variation in a way that is structure-preserving

- Structures in the context must be respected in the (changes in the) behaviour

$$a \bar{R} b \Rightarrow \underline{f}(a) S \underline{f}(b)$$

Monotone maps between context and behaviour (as relations) ensures that this happens

- A context that is “smaller” than another will give rise to a behaviour that is “smaller” than the other context



Selecting a behaviour

The process is therefore

1. Determine the scope of all possible behaviours, represented by inputs, outputs and maps between them
2. Determine the structure over the possible observed outputs
3. Develop a context model, with its structure
4. Contextualise by adding a new input (the context) that acts to select a map which is then used to determine the response to inputs
5. Specify the monotone map from the context to the behaviour such that the structures on both sides are respected

This provides an analytic “closed form” for the system as it adapts

- A *behavioural envelope*the system will remain within, although having freedom to adapt within it subject to structure

Maintenance and limitation

Most systems will have additional structure, and we need to capture this categorically too

Continuing the wireless document example

- There may be a set of documents that are always visible no matter how someone moves
- There may be a set of documents that provides an upper bound on the documents a person can see, regardless of where they move

More generally

- A behaviour that is “as large” as any other for every stimulus
- A behaviour that is “no larger” than any other for every stimulus

We refer to this sort of behaviour as a *core*

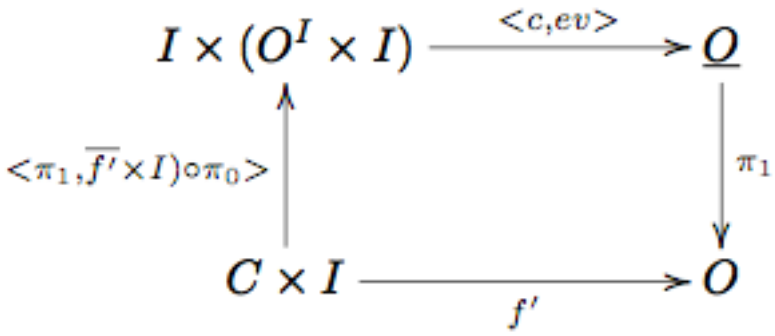
…and to this as an *extent*

Cores and extents – 1

For a behaviour to be a core, it has to be related to *any* other behaviour ever selected

$$I \times I \xrightarrow{<c,f>} \underline{O}$$

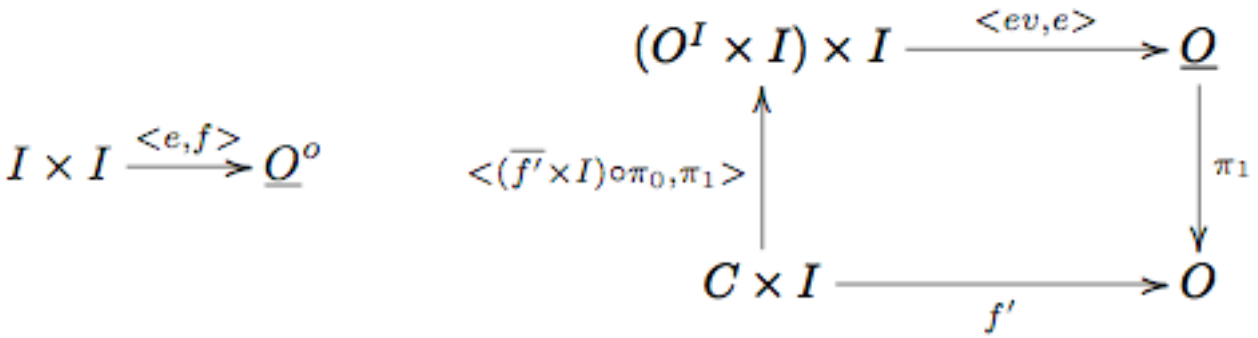
We can factor this into the selection process



Core and extents – 2

Extents are just cores of the inverse relation

- Must have any selected behaviour related to it



Cores and extents provide lower and upper bounds on behaviour, that is then guaranteed throughout an interaction

Composing context – 1

The foregoing will work as a model of context, but does not accurately reflect how we *want* to build adaptive systems

- It's a model of how we currently do things – all aspects specified in one go
- …but we need a more compositional approach where we can deal with each aspect independently

Important for a number of reasons

- Design complexity – reduce the things to bear in mind simultaneously
- Dynamic composition – people just turn up with mobile devices
- Extensibility – add new sensors, remove failed ones, …

Composing context – 2

If we have two contexts – say a person's location as a named space and their identity in an organisatonal chart – then we might specify their effect on adaptation separately and then combine them

$$(P \times N) \times I \rightarrow O$$

$$(P + N) \times I \rightarrow O$$

- Product – we know a person's location and their identity
- Co-product – we know one or the other

There are several ways of performing this composition

Composing context – 3

Consider the case where

- An individual's rights to see documents depend on their organisational affiliation – "outsiders" see less than "insiders"
- All people entering a building must be able to access the safety card at all times

An extent specified by a person's affiliation

A core for location-dependent behaviour

When we combine these we need to decide on priorities

- Should we let someone see more because they're in the building?
- …or less because they work for another firm?

The choice depends on the application, but this sort of analysis makes things clear and explicit

When things stay the same

When we move around a building we expect the system to adapt to us – but not *all* movements result in change

- Only “significant” changes, where the context “really” changes, should give rise to behavioural change
- So one can predict change by observing the environment, and reason back from an adaptation to some model of the environment

Even in pervasive systems, interaction *isn’t* seamless – and indeed shouldn’t be

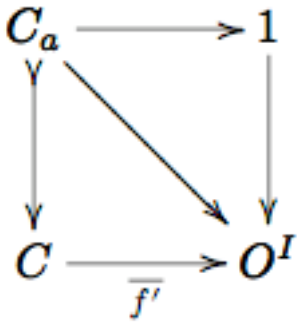
- The “seams” describe the change points
- Make sure that adaptations occur only at these boundaries

Coutaz *et alia*. Context is key. Comm ACM **48**(3). 2005.

Fibre structure

An adaptive behaviour gives rise to a *fibre* structure over its adaptations

- A part of the context which selects the same behaviour



Moving between fibres will give rise to adaptive changes; staying within a fibre is invisible

- Equivalence classes of contexts

Next steps

What we have now is a very simple model of context and contextualised behaviour

- Captures simple case studies
- Simple analyses

A number of weaknesses

- No real understanding of complex time-dependent processes
- Need stronger notions of composition and construction

May need a richer model

- Category of relations and monotone maps rather than sets?
- An algebra for doing the compositions more cleanly?
- Contextualisation as a functor might add power?
- Is scrutability an adjoint? – we strongly suspect it might be, and this would make a core principle of user interfaces emerge directly from the semantics, which would be a good thing

Conclusions

In order to develop complex adaptive systems we need to be able to describe the context we work in, the behaviours we want and the interactions between them in a well-founded way

- Find structures in the context and behaviour
- Respect these structures through adaptive mappings
- Build complex behaviours by composing simpler ones

In this way we'll hopefully be able both to address more complex overall systems and improve their construction and evolution over time