University of Dublin
Trinity College
Dublin 2, Ireland

Simon Dobson

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

simon,dobson@cs.tcd.ie

Applications considered harmful for ambient systems

Introduction

The idea of an application is so central to computing that we almost never think about it

- A packaged piece of functionality for deployment and marketing purposes

But are applications the right way to build the new generation of ambient systems?

We claim not

- Too much dynamism, too few certainties
- Migrate component composition to run-time using context, adapt inherently by using changing circumstances to drive selection
- Perhaps a better way to think about the problem…

Ambient systems

A system that can sense its environment and react directly to it

- Location, connectivity, proximity, …
- User, identity, tasks, preferences, permissions, …
- Processes, information, meta-data, …
- Devices, sensors, actuators, …

Generically referred to as context

The goal is seamless, "distraction-free", low-intrusion integration of information appliances and services into everyday activities

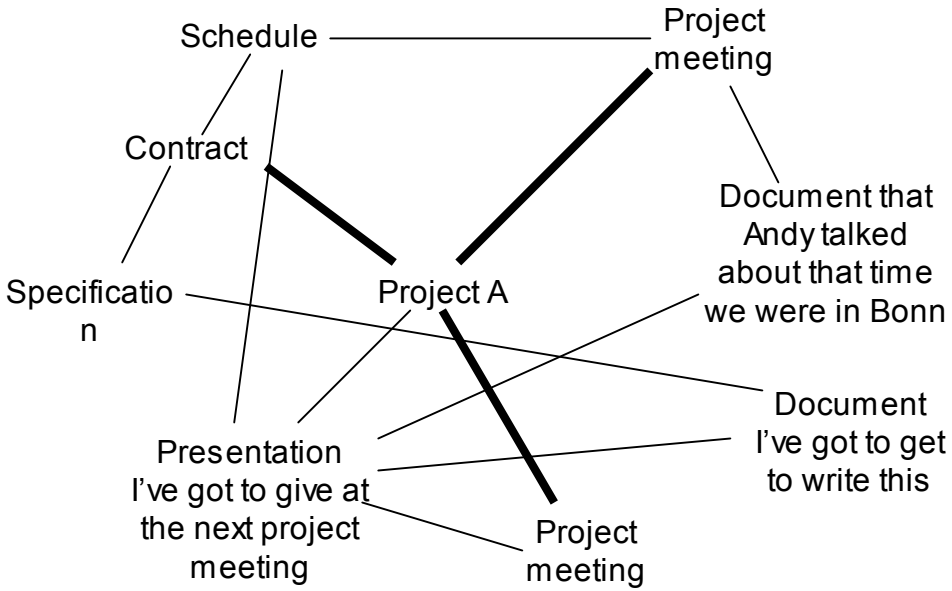
- Smart devices, smart buildings, smart systems
- Pre-empting user needs
- Wider participation by diverse communities in the information revolution – elderly, disabled, busy, …

The same, but different…

How developers see information:

C:\projects\A
- Contract
  - Specification
  - Schedule
- Minutes
  - Bonn-1Jul03
    - change-proposal.doc
  - Dublin-1Oct03
    - rebuttal.ppt
- Repository
  - change-proposal.doc
  - audit.xls
  - threats-from-lawyers-1.doc

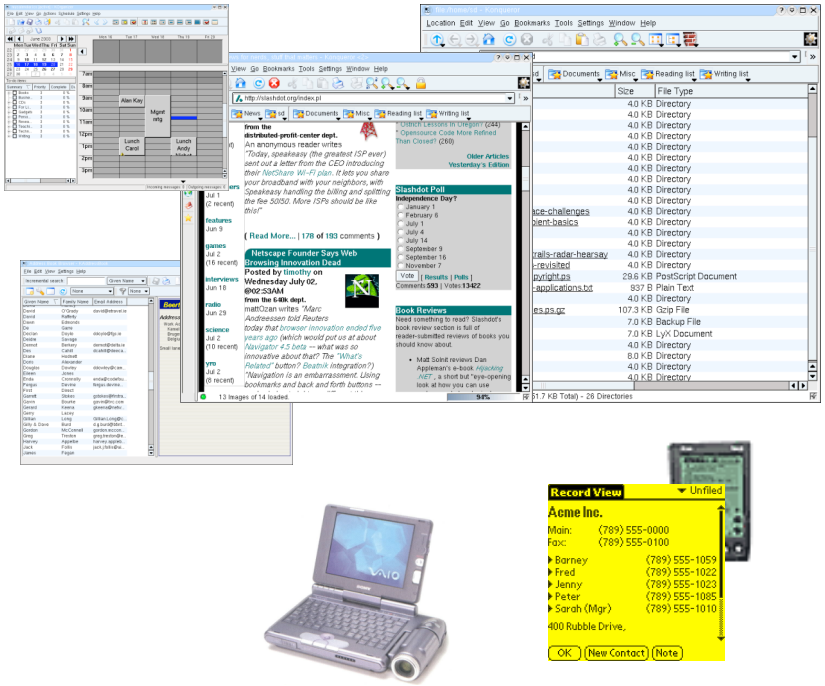
How users see information:



..and I'm in a hotel in Athlone with no network connection back home...

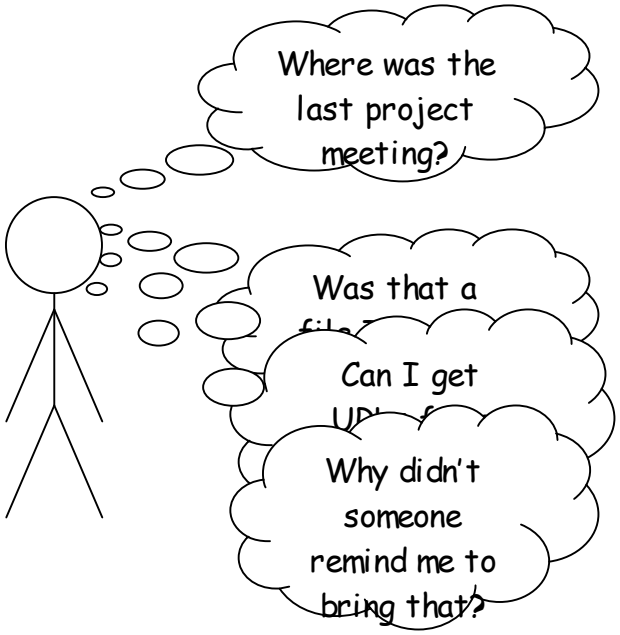
And what happens next is…

Developer:



Identify information → build tools

User:



Use tools → lose information

The problem

An application distils a view of some information, and provides an access mechanism

- meetings = diary, documents = files, people = address book

The problem is, there is no canonical structure for information that will be universally useful

- Between people, between organisations, …

You even take a different view at different times

- meetings = diary
- …until the week before, meetings = to-do list and address book
- …and the week after, meetings = files

You do tasks, not applications, and your software should support this

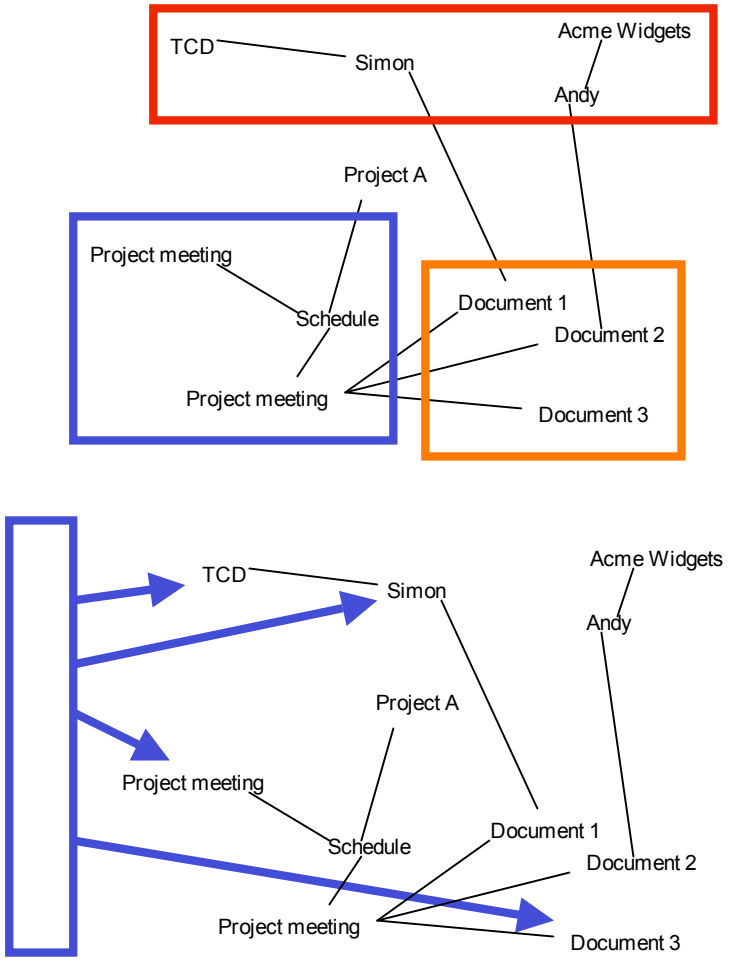
A possible solution: look the other way

Tools break the concept graph into sub-graphs, and wrap each one up in a tool

- Objects at their worst
- Rigid, lose cross-links
- Prevent re-interpretation, force a single presentation mechanism

Instead, why not project different views from a common knowledge base?

- Let the task decide



Using the knowledge

Suppose our shared conceptual structure is modelled using a “semantic network” approach such as RDF

- Information is well-typed, relationships semantically articulated

Suppose we also have a set of components that provide the “projections” onto the shared model

- A “view the documents needed for the next most urgent task in the schedule” component – high-level, user-centric

We can use the same model to drive the selection of components and their combination

- Context-driven application assembly – the “application” is the set of components that are “best” given the current state of the world
- Use information about use to drive use of information

Context-driven assembly

Collect into the context all the personal information and task descriptions

- Task description describes the relevance and importance of information as the process progresses
- Can make reference to other contextual items, i.e. appointments

Also include functional descriptions of components

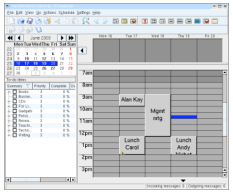
- “I arrange meetings”, “I find people”, …
- Relevance function from environment to component – can’t use a complex web browser on the move

Use relevance/importance to select components to present, and present them automatically

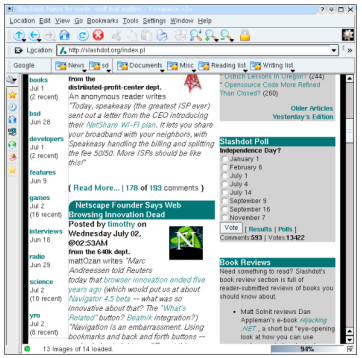
- The set of components is the application

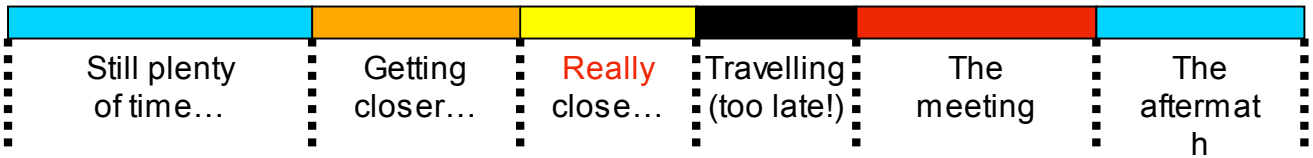
Example – the project manager

Although we're still using components, we're assembling an application – at run-time – that suits the state of the process.



One way to do this is that the application is the desktop: you use what's put there, at least as a first cut





Conclusion

It's harmful even to think of applications for ambient systems – the ideas just don't fit together properly

- Need a more intrinsically dynamic idea of what "system" means

We can't build Word or GIMP like this

- …and that might be a good thing…

But we can potentially build a highly adaptive ambient systems platform

- Components providing views/methods onto a shared model
- Assemble the "application" at run-time, changing with the changing context of the user (connectivity, location, device, free time, …)
- Don't build an application and adapt it: build a uniform component library and adaptively assemble it, driven by context