A taxonomy for thinking about location in pervasive computing

Simon Dobson

Department of Computer Science, Trinity College, Dublin IE
simon.dobson@cs.tcd.ie

Abstract. Virtually all pervasive computing systems use location as a major factor in controlling their behaviour. Simple representations do not however account for the richness in humans' perception of location which – if leveraged – can greatly improve applications' ability to use and reason about location, allowing

1 Introduction

Virtually all pervasive computing systems use some form of location for affecting the system's behaviour. Location-based services are available commercially, albeit in a primitive form, from many mobile telecommunications providers.

Despite this, location is a remarkably subtle concept to reason with. There are a huge number of possible answers to what is superficially a simple question: each kind of answer reveals something about the way in which we conceptualise location and any services based upon it.

This may seem like a trivial observation, but it cuts to the heart of the development of programming environments for pervasive computing. Software design is the process of mapping concepts to programming structures, whether one is developing object models or complete programming languages. A system is easy to program when the developer can express his concepts directly or with minimal mental gymnastics; conversely, the developer takes on a considerable load when using a system without the requisite concepts. Handling location is a basic requirement for many pervasive computing systems: but what *kind* of location will we provide in a programming environment? If we choose the "wrong" representation we will end up with an awkward (or unusable) environment.

In this paper we try to exhaust the ways one may answer a simple location question in a pervasive computing system, to develop a taxonomy of answers that an application might want to receive. We deliberately stray beyond the bounds of traditional location systems and the information typically at hand in order to see how a "really pervasive" computing system might view location. We analyse the space of answers to see what conclusions can be drawn for the development of semantic models and programming environments.

2 Thinking about location

When we speak of location we typically mean determining where some person or artefact is located in the real world. This can be used for a range of applications including adapting behaviour, controlling appliances, surveillance and health care[7].

Location sensor systems typically work in one of four ways[4]:

1. By directly tracking an item, for example using image processing over video images
2. By having an infrastructure that can track transponders attached to devices, for example as used in [3]
3. By having a device track something in the environment, for example the GPS satellites or local fixed transponders such as "crickets"[6]
4. By inference from actions, such as a user logging-in at a computer console or turning on the lights in a room

Each approach has its strengths; all suffer from noise, occlusion and missed events. Rather than engaging in a critique of particular sensor technologies, we can instead start from the other end of the development spectrum and ask two questions: what are the conceptual models that a developer might use to reason about applications? and to what extent can these models be realised directly within a development environment?

Suppose for a moment that we are trying to locate our friend Waldo without the use of any significant information technology – for example by ringing his home or office and asking where he is. We can imagine a whole host of ways in which this question might be answered. Each is a *plausible* and *correct* answer to the location question, and – assuming a certain degree of optimality in human languages – should identify a conceptual "space" in which humans reason about location.

2.1 Absolute location

Geography, maps and geometry provide an idealised view of space, and it is natural that this should be the most obvious way to talk about location in computer systems.

At $53°4'N, 1°17'W$ (*absolute position*) Absolute physical location systems typically use the Global Positioning System (GPS), a constellation of satellites used the triangulate a receiver on the ground. The result is a latitude/longitude position accurate to a few metres.

GPS location has no immediate connection with the real world, implying that a detailed map needs to be constructed of the features of interest. Behaviour is unlikely to be conditioned by co-ordinates *per se*, but rather by what (else) is at these co-ordinates.

GPS is both not precise enough and too precise. Imprecision comes from the natural error in locating using satellites and commercial-quality receivers. However, location to with a few metres is often actually *too* precise in that one must always work in terms of what is "near" the person rather than what is exactly "at their position".

Suppose I am in a room – GPS is often unusable indoors, but this doesn't affect the argument – and the location system has correctly ascertained this. I then move closer to the door but remain inside the room. Errors in location may place me outside; "nearness" calculations may determine I am nearer the corridor; and both will cause the system to locate me outside the room.

A GPS system is no panacea, and must be used with care. GPS hardware is relatively large, has relatively large power requirements, and must be combined with considerable information that is hard to gather and maintain.

2.2 Structured naming

We tend to assign names to anything perceived to have a unique identity, and spaces are no different. We can use these names to refer to locations, effectively providing the location system with a map based on human-meaningful names.

In OR3.16 (*named space*) Most "structured" spaces have sub-spaces with names – offices, streets, squares, the penalty area etc – which relate directly to a user's conception of both the layout and usage of space.

Using these terms within a location system has the huge advantage that any behaviour that a user expresses will probably be couched in terms of the space names, and so can be translated directly into the internal representation used by the location system.

Logical naming is particularly well-suited to systems that infer location from other clues such as use of a computer keyboard. The location of the clue can typically be expressed quite neatly using a logical space name.

There are of course some unstructured spaces that do not have meaningful names, and it is probably better to resist the temptation to invent them.

In a conference room (*named class*) This is a variation which identifies a *class* of spaces rather than a space itself. Such names are generally functional (as

in this case), although it is conceivable that some other naming scheme might be used ("in a red room"?).

This is an example of an *uncertain* location, in the sense that the possible locations not only have physical extent (which is true for most cases) but typically have *disconnected* physical extent.

2.3 Relative naming

By "relative" naming we mean spaces identified by their relationship to some other object, space, location or artefact.

In his office (*subject static space*) This answer relates a person's location to a place they have a know relationship to. The difference between this and literal naming is that Waldo may move his office to another space. This introduces another problem of the stability of location references over time, which we discuss more fully in section 3.6.

In his car (*subject dynamic space*) This is superficially the same as the previous case, but the space is different. From one perspective a car is a space that a person or artefact occupies; however, it is a space that follows a path and has a dynamic interconnection with other spaces. So is it a location? – few would argue for a definite "no", but it is clearly a location of a different order to others. We return to this point in section 3.5.

In Widget and Sons' offices (*related space*) There is a variation on the theme of a related space where the space is related to some other entity, not to the user. In this case the location system needs to be able to navigate the relationship graph to locate the entity the space relates to. One could reasonably assume that there will typically be a relationship between Waldo and this entity.

With Willard (*related association*) Both the previous answers use spaces as their referent, which are at least reasonably fixed. However, a common answer might be that Waldo is "with Willard". If Willard's location is known, then so is Waldo's; of Willard's location is not known, then neither is Waldo's – although we know that they are together, so locating one locates the other.

There is an obvious recursion if one asks where Willard is and receives the answer "with Waldo". However, a sufficiently rich set of possible location approaches should reduce the possibility of this happening in practice.

2.4 Approximate answers

As mentioned above, no location system is completely precise – nor would we generally want it to be. However, while many systems make the *pretence* of precision, others expose the uncertainty of their answers.

At 1000 he will be ... (*in the future*) Suppose it is 0900. We do not know Waldo's location *now*, but we know what it *will be* in the near future.

This is important, as many applications will ask for location in order to prepare for a future event, and so this answer may be completely adequate: if the application is trying to arrange things for Waldo's 1000 meeting, then it is probably not germane that he is currently on a particular street. Indeed, this points to a weakness in many services conceived as location-based: it is not the exact current location that counts, but the *next relevant location for the application.* We return to this point later.

At 0800 he was ... (*in the past*) The dual of where Waldo *will* be is where he *was*. Again, some applications may be adequately served by knowing someone's recent location.

We might make educated guesses about Waldo's range of possible locations based on how far he can have travelled since his sighting. The further we get from 0800, of course, the less reliance can be placed on this method.

Another subtlety is whether the time refers to the time he *began* to be at the location, or the *last* time he was there.

Near/Within ... metres of ... (*in vicinity*) One can make imprecision explicit by stating the "error bars" of the location statement.

"Near" is a highly subjective term, but might typically be interpreted as meaning "close enough to still be of relevance". Providing a harder bound, such as a radius (as is done with location using cellphones), can be more useful but can also be distracting.

Between ... and ... (*on path*) Suppose we have access to Waldo's schedule, and so know where he was at 0800 and where he will be at 1000. At 0930 we can reasonably infer that he is between these two locations.

The interest of this form of answer is that it locates someone on a *path* rather than in a *place.*

His badge/phone was last seen at ... (*by proxy*) It is worth making explicit that most location systems do not see Waldo, but rather see some artefact closely associated with Waldo. The assumption – which may be false – is that the artefact will never stray far from Waldo.

That this assumption of flawed is obvious: Waldo may have leant his cellphone to a friend, or may have been robbed of his badge. This entwines the location problem with an identity problem – which can be just as subtle. In applications that use artefacts as a surrogate for a person the identity is often split between "something you carry" and "something you know" – ATM cards are the most familiar example.

2.5 Task-based answers

Yet another form of location is task-based: we capture someone's location by capturing the task they are engaged in. The assumption is that the task somehow relates to the location, and this remains true even when the task is not concretely located in space.

Meeting Widget and Sons (*task*) This form of answer essentially dodges the question: instead of answering with *where* Waldo is, it answers with *what* he is doing. There may or may not be a location associated with this task.

However, few people would argue that this answer is *incorrect*, even though it does not actually refer to location at all. This is another example of location being tied-up with other aspects of context.

At this time he is usually ... (*by default*) Absent any other information we may use default logic (in the formal or informal sense) to locate Waldo. People are far more regular than is generally realised – experiments reported in [2] show a frightening regularity in some specific cases – so the use of defaults can be very powerful.

2.6 Negative answers

The final class of approaches inverts the problem. Rather than stating where someone is (with some degree of precision) they instead narrow down the possible locations by removing some.

Not ... (*by negation*) Perhaps the least expected form of answer – and the most confusing from a computer science perspective – would be to answer a question of where someone *is* with an answer about where they *aren't*: surely this doesn't constrain the possible locations enough to be of any use at all. This turns out not to be the case.

We encountered this answer in designing a system for a user with a mild physical disability, where action could be triggered by knowing that the individual had left home (to go to work), without actually being able to locate them otherwise. This style of response can be much easier to generate than any of the others, as it is inherently limited to a small scale. Nevertheless, many systems that are are conceived as location-dependent may actually be "non-location"-dependent, in the sense that they behave according to someone's non-presence in a location regardless of their actual location elsewhere.

Out/on holiday (*non-located task*) This is similar to the task-centric answers (section 2.5) but with a twist: while those answered specified location in terms of task, this form does not necessarily have an associated location (or least not one that is known),

However, even with such limited information one may conclude something about where Waldo *isn't*, with some degree of precision or confidence. One can therefore also regard this answer as a variation on the one above.

No idea (*unknown*) We (finally!) come to the possibility of someone being un-locatable by any means.

Most designers would expect this answer some (if not most) of the time. However, it should be clear from the foregoing that it can be made almost arbitrarily unlikely in practice by combining fragments of knowledge from other sources. Some of these may have little obvious relevance to location but – with enough information and (admittedly uncertain) reasoning – can be used to contribute to at least some form of answer.

3 Discussion

The full taxonomy of 17 answers is shown diagrammatically in figure 1. The diversity of answers suggests that modeling location in a system that aspires to generality will be a major challenge. In this section we try to tease-out some of the underlying forces at work to see what (if any) conclusions can be drawn for systems development.

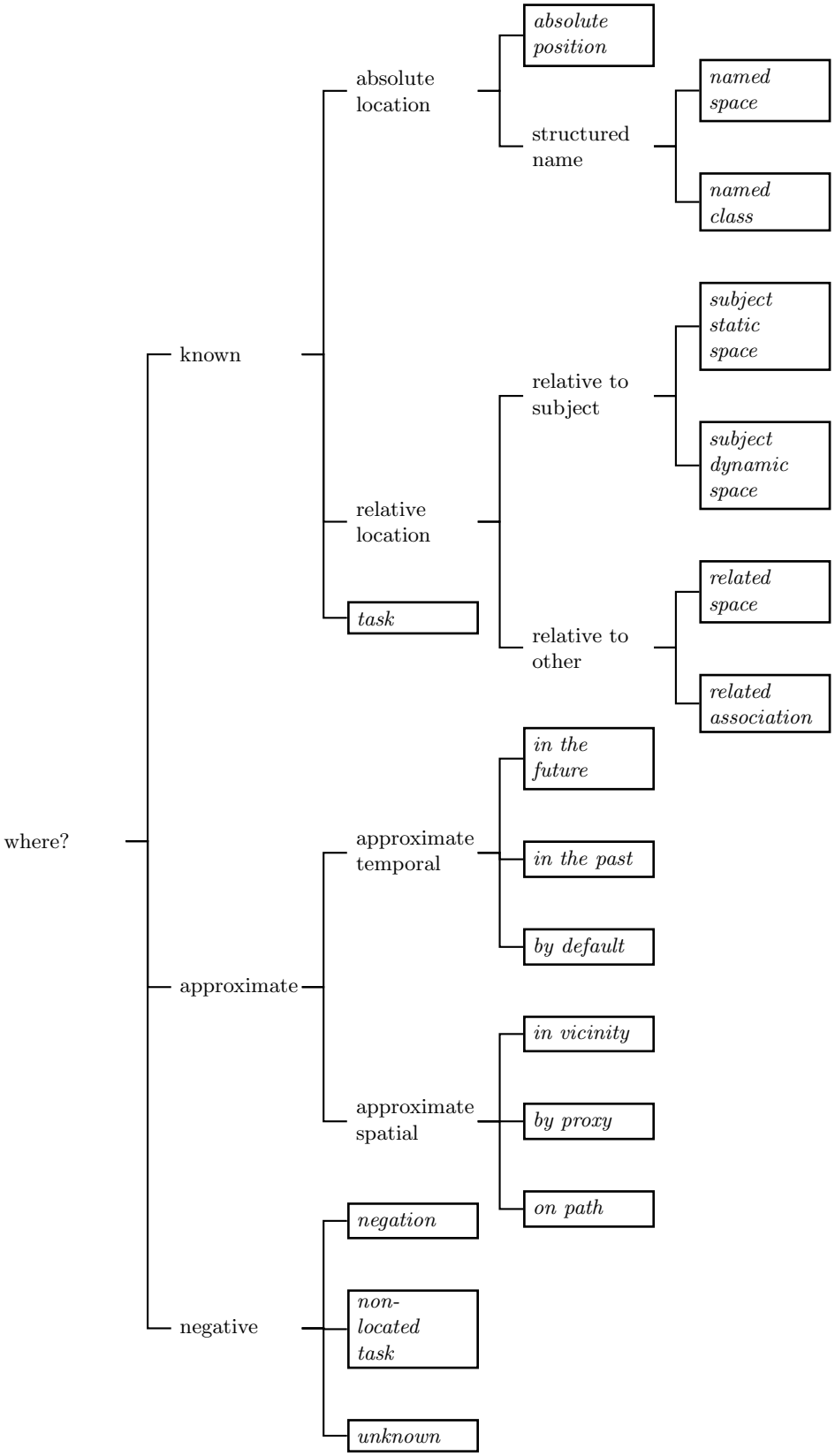

Fig. 1. A taxonomy of location in pervasive computing

3.1 More information than we think

Perhaps the most surprising result of the foregoing is that there is a huge potential well of information about location. "Potential" is the operative word, of course: it requires information to be held in machine-readable form in an accessible location and kept up-to-date. However, it is also clear that one could in principle build a location-based service without *any* dedicated location hardware at all.

More importantly, maintaining a richly-connected context opens up the possibility of getting multiple answers to the same question. If these agree, we have increased our confidence and reduced the effects of noise; if they disagree, we can use this disagreement constructively to make a decision informed by probabilities.

Finally, even the most ambitious pervasive computing can realistically be expected to have "gaps" in its sensor coverage, whether permanent or through partial failures. One could address this by adding more sensors, but this may not be possible and may be an unnecessary expense for many applications which, when thought of correctly, do not actually need good physically-based location at all. Having multiple sources of information can act to plug these gaps quite effectively.

3.2 Is the application location-based anyway?

We mentioned earlier than most pervasive computing applications have at least some component of location-awareness, and that location is a fundamental part of a context model. However, the requirements that applications place on their location-awareness varies rather more than might be expected.

Some applications undoubtedly require real-world co-ordinates, either directly or to drive a logical naming scheme. However, as the examples we used above have shown, one can conceive of a host of applications whose requirements are rather different. Behavioural variation may bind to a dynamic space such as a car, or to a task, or to someone not being somewhere – none of which necessitates GPS levels of precision.

We gave an example of a location-based service that was actually "non-location"-based. We conjecture that many other services – perhaps a majority – can be conceived in similar ways by taking a broader view of location.

3.3 The uses of inference

Clearly a complete model might also maintain several different models of the space it was interested in: physical, ownership, names, etc. This is the essence

of mapping and Geographical Information Systems (GIS)[5]. The mapping between layers can be seen as a mapping between semantic domains, for example from physical co-ordinates to a well-known name, membership of a class, and its ownership.

Many of the approaches identified above specify location relative to some other phenomenon such as a person or task. This suggests that a maximally flexible pervasive location system would function as part of a larger contextual system where it could acquire information from other layers of the contextual model, using "layer" to refer to a single aspect of context. The relationships between location layers and other contextual layers are clearly very rich, but it is this richness that is potentially the strength of pervasive computing.

Making use of this rich interconnection requires significant reasoning and inference over fundamentally uncertain data, informed by knowledge of what each layer means and what default behaviours can be expected. This is rather more information than is usual at present: most diaries, for example, are free-form, do not typically contain regular tasks such as lunch and coffee, and so could not be used reliably to determine a user's task. Viewing location – and context in general – in this way suggests possible improvements in application design for use in such systems in the future.

3.4 Space and extent

Several approaches discussed above do not tie location down precisely, but instead reduce the possible spaces a person might be in. This is often sufficient for applications that react to broader contexts rather than narrowly to location, even if they are generally considered location-based services.

Two answers are particularly interesting. The named class answer (section 2.2) identifies a set of spaces sharing a common (usually functional) class. The path answer (section 2.4) identifies a path through space that the user is following (called a *trail* by some groups). In both cases location is widened to encompass a "bounding box" – possibly not contiguous in space – that can be usefully applied to pervasive computing.

3.5 Dynamic reconfiguration

The topology of spaces can also change. A car or train is a classic example – a "space that moves", and which has a dynamic connection with other spaces while at the same time having a distinct identity of its own[1].If we admit "mobile" spaces to a location model, we introduce dynamism into the map in terms of physical location and the accessibility of spaces from one another.

[1] There are also examples of this dynamism in buildings – see [1]

One could of course simply remove the notion of a car as a space and only allow "static" spaces, but this has two disadvantages. Firstly it is an unnatural and somewhat arbitrary decision to allow one kind of named space but not another. Secondly (and more importantly) there are useful behaviours a system might take when a person is in this space – switching a cellphone to hands-free, for example. These activities emphatically bind to the mobile space, not to the succession of static spaces the user may occupy.

3.6 The effects of time

Location changes with time. When we ask someone's location we might aspire to receive an up-to-date answer, but will often have to cope with an answer that refers to a time in the past (or indeed in the future, as in section 2.4).

Which is better: a precise answer that is fifteen minutes old, or an imprecise answer reflecting what is happening now? One cannot answer this question *a priori*, and it is easy to come up with pathological applications if either one is chosen without the other.

Any location statement is therefore time-bounded, in the sense that it "ages" to a point when it is virtually useless. Just as one can make over-strong assumptions about the precision of an answer, one must be equally careful about an answer's timeliness..

We can even devise scenarios in which we do not *want* up-to-the-minute location. Suppose we are writing a restaurant guide, and Waldo asks about for a recommendation while on the train to his 1000 meeting. We do not care that the train is in (for example) Co Laoise: what we care about is that the meeting – the next time Waldo is in a position to use a restaurant – is in Co Cork, and this should be the location we use. In other words, the answer we want is intimately connected to the application we are building and the relationships with other contextual layers.

Time has another, more insidious effect. Space is not fully stable, and especially the names and uses of spaces change over time, as do the exact bounds of spaces and their relationships to people and each other.

A GPS location reference is "temporally stable" in the sense that it its referent will not change through time. The name of a room, however – and even more Waldo's office – *will* change over time. If one were to archive a temporally unstable location reference, its referent might have changed when one later examined it. In the presence of rich interconnections and inference this might cause a cascade of problems, for example inferring that Waldo could see into Willard's office from his own *then* simply because he can *now*.

4 Conclusion

> If you only have one representation of something, you have a poor one – Marvin Minsky

We have identified 17 different, plausible and potentially correct ways to answer what appears to be one of the simplest questions one might ask of a pervasive computing system: someone's location. This multiplicity of possible ways to conceptualise location gives rise to a multiplicity of possible representations and programming interfaces.

While location is undoubtedly a subtle issue, it is unlikely that it is unique among contextual parameters – it seems implausible that all other context one might account for will be trivial. This suggests that the problem of representing contextual information within a pervasive computing system is somewhat more challenging than might have been thought.

Moreover it is clear that there is no single canonical representation of location that will be uniformly useful for all possible applications. If one can constrain the application domain sufficiently then it may be possible to extract a "best" answer; however, as pervasive computing spreads it is inevitable that applications will need to be composed together unexpectedly, so the prospects for finding a single answer that will work across the expected spectrum of applications is not good. We are therefore drawn to the conclusion that multiple simultaneous representations of location – the bugbear of traditional systems analysis – are all but inevitable for pervasive computing applications. And we doubt that location is uniquely subtle in this respect.

We can easily find applications which "prefer" certain answer formats to others, and these preferences are often conditioned by the *use the application will make* of the answer it receives. The restaurant guide mentioned in section 3.6 is a good example of where a GPS location alone would be essentially useless. Location must therefore be taken holistically as a layer of context linked to other layers – and once again we doubt location is unique in this.

We conclude that the focus of context-aware computing is not the individual layers but the concurrent, consistent, synchronised relationships between them, and especially in matching the application's view of contextual information to its conceptual view of the application domain. This is a major challenge for systems design, but one that – if addressed properly – will do much to improve both the usability and compositionality of pervasive computing systems.

References

1. Michael Fox and Bryant Yeh. Intelligent kinetic systems in architecture. In Paddy Nixon, Gerard Lacey, and Simon Dobson, editors, *Managing interactions in smart environments*, pages 91–103. Springer-Verlag, 2002.

2. Richard Greenane. Managing interactions in smart environments. Master's thesis, Department of Computer Science, Trinity College Dublin, 2002.
3. Andy Harter, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.
4. Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.
5. Paul Longley, Michael Goodchild, David Maguire, and David Rhind. *Geographic information systems and science*. Wiley, 2001.
6. Nissanka Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket location-support system. In *Proceedings of MOBICOM 2000*. 2000.
7. Debashis Saha and Amitava Mukherjee. Pervasive computing: a paradigm for the 21st century. *IEEE Computer*, 36(3):25–31, March 2003.