"Devices Are People Too" Using Process Patterns to Elicit Security Requirements in Novel Domains: A Ubiquitous Healthcare Example

Yang Liu, John A. Clark, and Susan Stepney

Department of Computer Science, University of York, UK

Abstract. We present a set of process patterns that can be used to as a systematic way of analysing security requirements in a novel domain, taking ubiquitous healthcare as an example.

Keywords: Ubiquitous systems, security requirements, process patterns.

1 Background

Conventional computer technology designed for office use is inadequate for use in a hospital setting. Characteristics of medical work are fundamentally different from those of typical office work [Bardram 2003]: extreme mobility, *ad hoc* collaboration, interruptions, high degree of communication, etc. In a hospital setting, pervasive computing elements will be embedded in things such as instruments, tablet blister packs, bottles, wheelchairs, badges and staff uniforms. Security is a difficult issue in current healthcare systems, e.g. [Anderson 1996a, Anderson 1996b, Vaclav 1998], and traditional security models are very hard to apply to the new kinds of pervasive systems. Worse, for pervasive healthcare systems it is not yet clear what the security requirements even are.

When such systems become established we will be faced with all the old problems of traditional IT systems, plus many more due to the characteristics of ubiquitous computing (pervasive access, wireless communications, significant mobility, etc). What should the security requirements be for such systems? How should a healthcare system designer go about determining such requirements? We do not know: there is no systematic approach or guidelines to tackle the pressing concerns for healthcare systems incorporating pervasive technologies.

We propose the use of **process patterns** to guide the elicitation of new security requirements of novel pervasive computing application domains. We have developed useful patterns and anti-patterns that characterise issues, problems and some policy strategies, in a manner inspired by software engineering patterns work. Several such patterns are presented in this paper.

2 A Way Forward - Process Patterns

Patterns [Alexander *et al*. 1977] capture expert knowledge about commonly occurring problems and situations, and their solutions, expressed in ways that (should) assist

D. Hutter and M. Ullmann (Eds.): SPC 2005, LNCS 3450, pp. 31–45, 2005.

less experienced users to solve problems. As such, individual patterns may sometimes appear relatively banal, especially to those expert users. However, the purpose of patterns is to codify good engineering experience, and the cumulative effect of a whole Pattern Language can exceed the sum of its individual patterns.

Patterns are used in software engineering to document and promote best practice, and to share expertise. Existing patterns cover a range of issues, from coding standards [Beck 1997], through program design [Gamma *et al*. 1995], to domain analysis [Fowler 1997] and formal specification [Stepney *et al*, 2003], and meta-concerns such as team structures and project management [Coplien 1995]. The pattern concept has been extended with antipatterns, illustrating developmental pitfalls and their avoidance or recovery [Brown et al. 1998].

Our eventual goal here is the creation of a catalogue of specific policy requirement patterns for ubiquitous systems. System developers would browse such a catalogue to determine which patterns were suited to their particular needs. Suitable patterns could then simply be incorporated into security policies. The catalogue could be viewed as an analogue of the security patterns catalogue from the Open Group [Open Group 2002], but addressing higher level (policy) concerns.

However, much work is needed before such a catalogue of specific patterns can even be started. There are no available patterns for security *requirements*, and little in the way of security requirements for ubiquitous healthcare systems. The state of practice is that we do not yet know what the regularly occurring problems are, let alone their solutions. A catalogue today could provide patterns that are only provisional solutions (at a policy/requirements level) to problems we *believe* will occur. Also, whatever security catalogue becomes available, new systems will inevitably raise concerns that are not covered by it. What then should developers do? Can we help them to create new requirements and requirements patterns? What is needed now is a process for *creating* the catalogue in the first instance: *process patterns* for eliciting novel requirements.

We have developed an initial version of such a process pattern catalogue. The patterns it includes should be useful not only in the ubiquitous healthcare domain, but in a wider set of novel application domains. Indeed, the generation of this catalogue itself is a novel domain, and the patterns can be used at a meta-level: we applied the patterns we were developing to their own development.

Here we present an outline of some of the patterns in our catalogue that are most specific to ubiquitous healthcare, and how we applied them, both to get more specific patterns, and in the generation of the process patterns themselves. The full catalogue can be found in [Liu 2004].

3 Structure of the Patterns

Each pattern given in the text below is presented in a common four-part structure:

- **Name**. This provides an easily remembered and intuitive name for the pattern.
- **Intent**. A summary of what the pattern provides, what task it aims to facilitate.
- **Motivation**. A brief rationale for why the pattern is needed.
- **Solution**. This provides a description of how the intent of the pattern can be realised. There may be several such solutions.

In the text below each major pattern is presented in a section on its own, with the name of the pattern indicated in the section header in quotes. The first such pattern is the "Getting Started" Pattern.

4 "Getting Started" Pattern

Intent
To get started in an unfamiliar domain, where requirements are not fully understood.

Motivation
Security is a difficult issue in current healthcare systems; the security issues emerging from pervasive computing will be even more complicated and unpredictable. We have no *systematic* approach or guidelines to tackle the such systems. However, we do have some generic requirements elicitation techniques at our disposal that should be part of any requirements elicitation process.

Solution1: Literature review. Get up to speed by reading the existing literature.

Solution2: Stakeholder interview. Use the classic requirements elicitation technique of identifying and interviewing all the relevant stakeholders.

Solution3: Security experts. Discuss requirements with general security experts, to get overall requirements for secure systems, in order to adapt them to the specific domain.

Solution 4: Brainstorms. This potentially very useful technique can help to highlight big issues and requirements arising from the earlier reviews and discussions. It can also identify issues missed or not in the scope of other techniques.

Solution 5: Scenarios are an important classic technique, and form a specific process pattern of their own (see later).

Origin and Application of the "Getting Started" Pattern

Literature review. The current literature highlights ubiquitous healthcare environments comprising a mixture of participants, including people, devices, other infrastructure etc. The distinction between people and devices is somewhat blurred, with some similarities between device and personal concepts. This suggests that a deliberate blurring of the two could provide inspiration for pattern generation operated by equating analogous concepts between the two domains. If we have a requirement for one this immediately suggests a possible requirement for the other. This eventually led to formulation of the "**Devices are People too**" pattern.

Stakeholder interviews. We interviewed two senior Hospital staff responsible for security issues of the current IT systems. Various security issues were raised during this meeting. Of particular note was the "erroneous mapping" issue between the virtual (computer) model of the system and what is actually happening in the real

physical world. This led us to consider a much more general issue of mapping between the two worlds, and to the "**Model out of sync**" antipattern.

Security issues of portable devices is another point exposed by this interview. The security requirements of the mixture of participants involved in ubiquitous healthcare environment are major concerns, especially embedded devices with sensors that may exchange information with other agents (people or devices). The security requirements of devices need to be considered as analogous to those of people, incorporated into "**Devices are People too**".

Security expert discussions. For this research, two of the authors who had extensive experience of industrial security (Clark and Stepney) acted as the security experts. The literature survey highlighted the importance of privacy as an issue. Agents in ubiquitous systems require privacy, but also want services to be provided. There are obvious conflicts to be addressed. We chose to consider conflicting goals of a service requester and those of a service provider under a variety of different scenarios (not discussed further in this paper; see [Liu 2004] for details).

Brainstorms. We know that all stakeholders should be able to "buy in" to ubiquitous system policies and should be treated appropriately. So we considered some specific requirements, and asked what would happen if we replaced the stakeholder subject to it by another. We used this to help generate "**Substitution**" as a means of generating new requirements from old. In those cases where substitution does produce a further acceptable requirement, then the old and new requirements can be regarded as specific instances of a higher level one. We used this to help generate "**Generalisation**", to create a more widely applicable or abstract version of a specific requirement.

5 "Build Scenario" Pattern

Intent

To systematically identify the situations an agent will find itself in and consider the security issues that apply.

Motivation

Determining requirements is hard. Often important requirements are missed or are inaccurate. It is very easy to miss subtleties that apply to specific circumstances (or even appreciate what special circumstances there are – see "**Consider Modes**")

Experience is a great educator: one of the best ways to appreciate a need is to be faced with that need. We need to pre-empt the experience we will have when the system is built.

In normal requirements engineering, a *scenario* can be used to capture a particular transaction in a concrete way (and then be abstracted to a more general Use Case [Cockburn 2001] [Adolph *et al* 2003], for example). Traditional Use Case scenarios tend to focus on functionality, but scenarios are also excellent ways to focus attention on non-functional issues, such as security [19]. Scenarios allow us to 'walk through' situations that an agent may encounter, at early stages in the requirements process.

On walking through situations, we can see what encounters are generated, and so can consider what we need to do in those circumstances.

For many agents in the system we can develop a scenario of the whole of their activity over time, not just a single transaction. Thus, the working life of a person is a scenario, with an entry point (a nurse joins the hospital), a middle (a series of working days which may involve significant repetition) and an end (the nurse leaves the hospital). And if a nurse has a lifetime-scenario, then so does a scalpel, or a broom, or indeed any other device. The scenario allows us to consider the circumstances agents will find themselves in over various timescales.

This pattern prompts the security analyst to systematically consider situations otherwise neglected. The lifetime of an agent can be viewed as a series of internal actions and interactions with other agents. Interactions take place via protocols. By exploring the life of an agent through a scenario we are prompted to consider the agents it comes into contact with, and in which contexts, and to consider the security issues that arise.

Solution

Build scenarios by tracing through the activities of an agent over time. The scenarios represent the operational contexts for an agent. Use the scenarios as prompts to discover the situations an agent will encounter, and so increase the chances of completeness in requirements.

Step 1. Simple scenario. Identify a feasible series of actions over an appropriate timescale. This may be the steps in a procedure, or the whole lifetime of an agent. Identify the agents one encounters, together with any possible interactions between those agents and the agent in question. Identify any locations entered and boundaries that are crossed and consider any security issues that arise.

Step 2. Fully moded scenario. Using "**Consider Modes**", produce a whole lifecycle of an agent (person or device), and generate various modes that happen in each stage.

Step 3. Analogous scenarios. Using the various "**Analogy**" patterns ("Analogy", "Substitution", "Generalisation", "Specialisation", "Devices are People too"), form other related scenarios to generate further requirements, by considering analogous lifecycle events of any device or any person.

Origin and Application of the "Scenario" Pattern

After brainstorming, we chose to apply the Scenario pattern to "A day in the life of a SmartBroom" – a mundane device that might be used in new ways once made "smart", and hence lead to new security requirements.

Step 1. Simple scenario

A broom starts its day in the cupboard, is taken out, is used to sweep a variety of floors, is cleaned at some point, and is eventually returned to the cupboard to await use again. Its locations are restricted (a ward broom should not be used to sweep the garden, or an operating theatre, to avoid spread of material and infectious agents). Tracing the location of a device as it proceeds through its day is useful.

For safety reasons, access control is an important issue (this SmartBroom is an expensive and powerful device): we must guarantee that a broom can be accessed only by legitimate users. Identification, Authorisation and Authentication need to be considered: who is participating in this scenario (e.g. cleaners, brooms, other staff, devices nearby, cupboards/storerooms, rooms/corridors)? Who may open a cupboard and take a broom out? When can it be taken out? And what protocols are needed?

The broom must be used only in places authorised by the security policy. The policy should cover aspects such as: where this broom is allowed to clean; how the broom is allowed to move from one place to another; what it is allowed to remember of the places it is near or passes through; and whether it is allowed to tell other agents (people or devices) where it is or has been, etc. We may wish to restrict the places a broom may be used: a general ward broom should not be used to sweep the garden, or a sterile operating theatre. Thus, there needs to be an enforceable policy of restricted use. We must be careful in how we partition space in the hospital and what restrictions we impose. There is little point in authorising a broom for use in a particular room, but denying it access to the corridors by which that room is reached from the storage cupboard!

We might specify whatever policy we might wish to hold, but this will need to be enforced. In some cases there may be a direct and clear implementation of policy: if a cleaner is not authorised to access a storeroom, then the room may simply remain electronically locked. In others, some rectifying action must be taken. A broom will generally be unable to prevent itself being moved by a cleaner through an unauthorised corridor. In such circumstances we might require some notification be generated, with an expectation of action. For example, a talking broom might inform its user that a security breach has occurred, or the environment might detect the breach and sound an alarm. Such considerations lead to the development of the more general "**Alarms and Emergencies**"; other considerations during this phase of the scenario lead to input to "**Conflict Resolution**" (Liu 2004).

What happens to the internal state of a broom when it is returned to storage. Should its current working memory be erased? It may be appropriate to break any association established earlier with a particular cleaner, but what happens to the history of its day? The latter question was sparked by use of "**Devices are People too**": memory is one of the concepts subject to that analogy. It reminded us that there are potential "device privacy" implications from day-to-day historical information. Issues of memory must be considered when brooms and other items of equipment are disposed of, leading into the "fully moded scenario".

Step 2. Fully moded scenario

A device may have many modes: maintenance mode, emergency mode, work/stand by mode, normal/abnormal mode, etc. A patient may have chronic disease/acute disease mode, in hospital/at home mode, etc. A doctor may have work/holiday mode, day/night mode, weekday/weekend/bank holiday mode, full-time/part-time/retired mode etc.

Scenarios occurring in the hospital exhibit various timescales. Some situations happen over a lifetime (e.g. the lifetime of devices or patients). Some situations happen in a day (e.g. family visitors coming to visit patients, emergency

circumstance). Even so, we can discover common features. For example, we may find many situations that are *repeated* (e.g. multiple entries of a patient to hospital, a broom repeatedly taken out to clean the floor and taken back after use). These patterns give us clues to help build up a suite of temporal modes for all agents.

Step 3. Analogous scenarios
By analogy to devices, nurses, doctors, patients might be tracked over time. What does a nurse do during the day? Where do they visit? Thus, a nurse will come into the hospital, change into uniform, look through patients' records, prepare tablets and injections for patients, go to see patients in the wards, and routinely record what things happen. If a general device is attached to a nurse, should it be allowed to announce its location? In the extreme circumstance, if a nurse is attending a sensitive patient (e.g. a famous patient or an HIV patient), devices attached to them might be required not to announce location, except to specific requestors. The lifecycle of a nurse can be "Substituted" by the lifecycle of a consultant to see what happens, and what new requirements emerge.

Such considerations helped us to develop "**Analogy**" and its raft of specialisations.

6 "Consider Modes" Pattern

Intent
To systematically generate and consider relevant modes of system use.

Motivation
It would be convenient if our security requirements were simple and our systems obeyed a one-size-fits-all philosophy. However, circumstances change, and our requirements may change with them, subtly, or in major ways. For example, we may prohibit the use of fire exits in normal usage, but certainly wish to allow their use in times of fire. Similarly, night-time operations may have different requirements from day-time operations. Once such varying circumstances and qualifiers have been identified, we can consider whether policy refinements are needed in their context. However, identifying such circumstances is not easy. Moding has been found to be a useful concept in the safety domain. This pattern provides a process to prompt us to systematically generate and consider relevant circumstances.

Solution
Four classes of abstract qualifier are particularly useful: **time-scale** mode, **spatial** mode, **event/circumstance** mode, and **attributes** mode.

- When a requirement is generated for one component of a mode (usually "normal operation", such as "daytime" or "on ward"), systematically ask questions of all the other mode components.
- When a new mode is discovered, generate all the mode components for it, and add them to this Consider Modes catalogue

Time–scale mode. A variety of temporal modes can be determined by calendar perspective, people perspective, and devices perspective.

- Calendar perspective: day (day/night, morning/afternoon/evening/night), week (weekday/weekend, specific days of week), month, season, calendar year, financial year (special last month of financial year), other kinds of years (tax, academic etc.)
- People perspective: (a) staff: on-shift/off-shift/break, holiday/at work, healthy/off sick, full-time/part-time/retired; (b) patients: short-term/long-term stay, etc.
- Devices perspective: (**Analogise** the people perspective) time modes for devices might be include shutdown/hibernation/at work/stand by, normal operation/abnormal operation, maintenance/upgrade/emergency, and full-time/part-time/disposed.

Spatial mode. A hospital can be partitioned by location in a variety of ways: partition by location type (such as cupboard, operating theatre, wards or corridors); partition by floor (basement, ground, first floor etc.); inside the hospital building, outside the hospital buildings (in the grounds) etc.

Event/circumstance mode. This mode relates to operational events/circumstances. Thus, we may identify a normal mode of operation. This reflects perhaps the most common set of circumstances. (In Use Cases normal operation would be reflected in the normal flow of events.) However, sometimes things deviate from such normality. For example, we can identify two sorts of emergency. There may be an externally generated emergency, such a major rail accident; such circumstances are fortunately not normal, but hospitals must nevertheless be prepared for their occurrence. An internally generated emergency would be a fire in the hospital. These two different classes of emergency must obviously be dealt with in different ways, and give rise to different security requirements.

Furthermore, circumstances can be considered on a loading axis: very busy (e.g. train accident), busy (e.g. regular peak time loading), not so busy (e.g. normal operation) etc.

Attributes mode. Different system components and stakeholders need to be treated in different ways according to particular attributes they possess. Requirements may be partitioned according to the attributes of each agent.

The most obvious attribute is role. Medical staff will be able to access data that patients should not. More generally, the actual role played by stakeholder will determine the rights and constraints that should be applied to them.

Other attributes might be used for making distinctions. A device may be mobile or immobile. It may be easy to casually steal a mobile device (such as a PDA), but stealing a scanning machine weighing several tonnes and fixed to the floor is a different prospect. One might wish to permit free communication (e.g. with respect to giving away location information) with immobile devices on account of this.

Different values of scalar attributes of a system may act as a cause for discrimination. Thus, we might consider agents with different physical attributes (such as size, weight or shape) differently. Non-physical attributes such as cost of replacement may also be relevant.

Origin and Application of the "Consider Modes" Pattern

The recognition of the modal nature of the "in cupboard/in use" part of the SmartBroom scenario sparked a **Generalisation**, which led us to develop the "**Consider Modes**" pattern, and many of its examples, in parallel with developing the Scenario pattern instance. This led us to generalise the scenario pattern to include a "fully moded scenario" step.

7 "Analogise" Pattern

Intent
To elicit and derive requirements for some agents and circumstances by analogy with requirements for other, analogous, agents and circumstances.

Motivation
Analogy has been found to be a great tool in the history of thinking [18]. The process is not generally subject to automation and so requires imagination. Thus, revisiting some security requirements elements but choosing to 'wear a different hat' forces the analyst to consider whether analogous concepts exist. This is a generally applicable pattern. We can apply it to specific technical requirements patterns for the system or to patterns for generating patterns etc.

Ubiquitous healthcare environments will comprise a mixture of agents. These may be people, infrastructure or mobile devices. We explicitly construct a variety of analogies between these classes of agents.

Solution
The solution can be at a very high level of abstraction. It reduces to taking a concept from one area of investigation and reinterpreting it in another situation.

In practice, more specific instances of this pattern are used, where we draw analogies between various classes of people and devices, taking characteristics of one and interpreting them for the other. We do not expect *every* application of the pattern to result in new requirements – some analogies simply do not hold – but we do expect the systematic application to throw up new and unexpected considerations in many cases.

"**Substitute X with Y**" systematically reinterprets requirements when one class of agent is substituted for another (for example, "consultant" for "nurse"). "**Specialise**" is an example of "Substitute", where Y is more specific than X, to check that a general requirement makes sense in specific cases. ("All devices shall announce their location when queried; specialise "device" to "dangerous drug bottle" and reanalyse.) "**Generalise**" is an example of Substitute, where Y is more general than X , to investigate if a more generic requirement can be formulated. And if X corresponds to people, and Y to devices, we get "**Devices are People too**", expanded below.

Origin and Application of the "Analogise" Pattern and Its Specialisations

We formulated "**Analogise**" very early on, since analogy is such a useful tool, then developed the particular specialisations subsequently. These patterns formed an important part of the process of deriving the other process patterns, by deliberately applying them to the patterns themselves. For example, "**Consider Modes**" was generated by application of "**Generalise**" to the modes discovered in the SmartBroom "**Scenario**" instance.

8 "Devices Are People Too" Pattern

Intent
Apply "**Substitution**", with X =people, Y = devices.

Motivation
In ubiquitous healthcare systems the distinction between people and devices is blurred (for example, we may often take the presence of one as indication of the presence of the other). We may be able to take advantage of this. We have well-established notions of what people want to maintain in terms of their personal well-being, but we have far less idea of what should be the well-being of a device or infrastructure component. If the distinction between people and computational facilities/devices is becoming blurred, perhaps we should consider how requirements established for one apply to the other.

For example, a requirement for a person to undergo a health check can translate to a requirement for a device to undergo a maintenance check (e.g. to calibrate an instrument). Although this may seem a little odd at first sight, it can have useful consequences. For example, we might not otherwise consider aspects such as the 'privacy' of a device, which has interesting interpretations.

By analogy with people, security issues can be identified for devices (e.g. trays, brushes, scalpels, tablets) in various situations. The application of such analogy patterns allows the early generation of ideas and requirements. The earlier such ideas/requirements are generated the better. (Some such ideas can also be generated by the application of other patterns.)

Solution
Starting with an assertion such as "Devices are people too" or "People are devices too", we define requirements for one, and then reinterpret these for the other. Some analogies may be more natural than others. The analyst is free to reject any that seem inappropriate. We simply wish to generate ideas for consideration.

In daily life, people have basic needs (e.g. food, sleep, privacy, health etc.). Using the analogy "Devices are people too" we can consider the corresponding needs of devices (e.g. power, maintenance, upgrade, security policy, privacy, input or output, communication with people etc.).

The table below captures some analogous concepts between people and devices. Some concepts have direct interpretations, e.g., memory (though it is perhaps easier to upgrade a devices memory!). Others are clear with re-interpretation, e.g. sleeping and

standby operations (some devices are even described as being in sleep or power saving mode). There is also a clear analogy between training/retraining and maintenance upgrades. One can also generate analogies between the various forms of audit activities, e.g. health checks.

Person	Device
Food	Energy
Sleep	Stand-by
Training	Upgrade/maintenance
Holiday	Shutdown/hibernation
Audit	Audit
Communication	Communication
Privacy	Privacy
Well-being	Health

When applying analogy, we must think about what the devices really need, what happens if devices go wrong, what kinds of granularity are appropriate, what would happen if we upgrade the device to a new version, or we reconfigure the devices, or whatever. For example, at intervals, medical staff need to be retrained; by analogy, we might consider that it is essential to update or upgrade the devices.

Origin and Application of the "Devices Are People Too" Pattern

We are used to considering security issues such as confidentiality for patients, but is there a corresponding notion for devices? By analogy with patient privacy, we look for some privacy policy for devices (e.g. trays, rooms, brooms, tablets). Suppose a SmartBroom is to be used only to sweep an HIV patient's room. If this broom is tracked by other things (people or devices) without legitimate privilege, its 'privacy' would be leaked, and it might be possible to link it with other things (e.g. location, room number and patient in that room). Thus, by considering the need for privacy of a broom, we have helped maintain the privacy of a person. The usefulness of an analogy is determined by the eventual results of generating it. People might say that the whole concept of device privacy is nonsense; this is in fact a minor concern. However, by simply considering the analogy we have generated new concerns that must be dealt with. The analogy can serve a purpose simply by prompting new questions.

Take another security requirement such as "The location of nurses shall be determinable only by agents with appropriate authority." What about devices? Should access to device location be similarly constrained? We could ask 'What is the reason for the requirement?" There may appear to be good personal safety reasons why nurses would not wish to disclose their location in various circumstances. Are there circumstances where devices should be similarly 'concerned' for their own 'safety'? Drug addicts might have designs on dangerous drugs bottles. More generally, if a device has high value then thieves might wish to obtain its precise location in order to steal it. So here we see how a reinterpretation of the concept of

'personal safety' for devices sparks considerations from which new requirements inevitably occur.

Here we might also investigate the privacy angle? If we are constrained in our ability to track a nurse but have free access to where their Personal Digital Assistant is, then (for many hours of the day at least) we do in fact know where they are. Thus we are prompted to consider the privacy aspects of devices and people together.

"**Devices are People too**" arose early in "**Brainstorming**", and was later formalised as a pattern. Concepts of device privacy led to several of the more specific patterns such as "**Conflict Resolution**" (Liu 2004).

9 "Model Out of Sync" Anti-pattern

Intent

To investigate the link between the computer model and reality by applying the notion of "**Devices are People too**" and "**Consider Modes**".

"Devices are people too" leads to consideration of incorrect links between the physical device and its model. "Consider modes" leads to consideration of the times of establishing and breaking the links.

Problem

A real world agent (person or device) is generally represented in the computer by some model of that agent, and the model is updated by sensing attributes of the real-world agent (possibly indirectly by sensing some attached *tagging* device). It is possible for the real world and the model to get out of synchronisation, for a variety of reasons. (See also [Jackson 2001, chapter 7]). This can potentially lead to severe problems.

We might evaluate the model status of a tag in order to deduce the real world status of an agent. We want the location of the person; we measure the location of their tag; we assume these are strongly linked, that we can take the tag to be a *proxy* for the person. If the link between tag and agent is broken (e.g. sensor removal), or the mapping between real world and model is wrong, then we get a mistaken view of the status of the real world.

Causes of Real World out of Sync with Model

Erroneous Linking Assumption

The link between what is measured in the real world and what is deduced about the real world from that measurement might be incorrect.

The location of an agent is not necessarily equal to the location of its tag. A person might accidentally mislay their badge, or deliberately remove it because they do not want anybody know where they are. The assumption "real world location of X's tag equals real world location of X" does not necessarily hold.

This is a particular example of a more general problem. Consider the example of monitoring the tablets a patient takes by tagging each strip of tablets so that the removal of a tablet can be sensed. If a tablet is removed from the strip, we might assume that the tablet is consumed by the patient. However, a patient might remove

the tablet, but then throw it away, or give it to another patient to consume. The assumption "tablets removed from X's strip" does not necessarily imply "tablets consumed by X".

Erroneous Mapping
What is measured in the real world might be incorrect. Even if the modelling assumptions are correct, it is possible for software or physical problems to cause the mapping between the model and the real world to drift out of sync. For example, a sensor might not sense its location correctly, due to low power, a software glitch, or communications interference. So we get the erroneous mapping between the sensor and the real world. The assumption "reported location of X equals real world location of X" does not necessarily hold.

Solution
There are several ways to bring the model and the real world back in synch.

Step 1: Identify linking assumptions
Identify the linking assumptions and the reasons for these linkings. Often it is not possible to guarantee these assumptions (how would you guarantee a member of staff did not remove their tag?), and so techniques to resync the model in the case of the identified assumptions not holding need to be designed.

Step 2: Analyse how links might break
In order to design a suitable resyncing strategy, we need to know how likely the link breakage is to occur, and what the consequences of such a break are. Infrequent low-consequence occurrences will have different recovery mechanisms from high-frequency high-consequence ones. For example, we might have information concerning how often tablets are removed but not consumed, whether this happens accidentally or deliberately, and what the consequences are if it happens.

Step 3a: Choose Calibration and Testing
While the model is running, it can be tested and calibrated it in order to find problems (e.g. wrong data, erroneous mapping). The "**Consider Modes**" pattern suggests use of the maintenance mode to ensure the normal status and keep the system workable. If we test or calibrate the model frequently, it is much easier to find the mismatching or the wrong mapping between the model and the real world. Step 2 gives input to how frequently this needs to be done in each case.

Step 3b: Chose Redundancy
Redundant sensing (multiple sensors) can be used to make the links stronger, and reduce the possibility of errors. In the tablets example, for high-consequence cases, it would be reasonable for a nurse to watch the patient taking tablets rather than leave the patient alone, to ensure that the tablets are consumed, and hence ensure the link is not broken. Again, Step 2 gives input to how much redundancy is appropriate in each case

---

Origin and Application of the "Model Out of Sync" Anti-pattern

In our discussions with the Health officials, it became clear that their current major concern is the difficulty of keeping the computer-based patient records up-to-date,

since the current state of the art data entry is nowhere near as immediate as pen and paper note taking. We formalised this into a pattern, then applied "**People are Devices too**" and "**Consider Modes**" to generate the "**Model out of sync**" pattern.

10 Meta-patterns

One of the most interesting parts of this work was how it was possible to apply these process patterns to the process of generating the patterns themselves. This has convinced us that the patterns are more widely applicable that just to pervasive healthcare, or even just to the pervasive computing domain.

11 Discussion and Conclusion

We have provided an account of several *process patterns* for eliciting security requirements in novel domains, and of how the actual patterns were developed. Some of the patterns are very abstract and so find wide applicability. Thus, "**Generalisation**" and "**Specialisation**" are two crucial, but very general, thinking tools. One might argue that we should not be surprised that such patterns can be applied. However, we find that the deliberate application of such patterns brings useful results.

An informal appreciation of the blurring in healthcare systems (and pervasive system generally) of the distinction between devices and people led directly to "**People are Devices too**". Applying "**Generalisation**" to this allowed it to be seen as a particular instance of "**Analogy**". Applying "**Analogy**" and its specialisations to requirements can be a useful way of generating new particular requirements, and also a good way of 'sanity checking' requirements of supposed general applicability.

The "Day in the life of a SmartBroom" work provided valuable insights in itself, and when subject to "**Generalisation**" led to the creation of "**Build Scenario**", which is highly flexible and can be instantiated as required. (Of course, we have simultaneously generalised away from broom, to device, to agent).

We believe "**Consider Modes**" and "**Devices are People too**" to be the patterns most valuable for ubiquitous systems. This work demonstrates that applying process patterns to generate new requirements patterns has considerable promise.

Acknowledgements

We thank Ian Sutcliffe (Director of Communications & Information) and Andy Moore (Head of IT) from Harrogate Healthcare NHS Trust, UK, for very helpful discussions in the early stages of this work.

References

[1] S. Adolph, P. Bramble, A. Cockburn, A. Pols. *Patterns for Effective Use Cases*. Addison-Wesley, 2003.

[2] C. Alexander, et al. A Pattern Language: towns, buildings, construction. OUP, 1977

[3] R. J. Anderson. Security in Clinical Information Systems. *British Medical Association*, Jan 1996.
[4] R. J. Anderson. *An Update on the BMA Security Policy*. (1996). Available from http://www.cl.cam.ac.uk/users/rja14/bmaupdate/bmaupdate.html
[5] J. E. Bardram. Hospitals of the Future – ubiquitous computing support for medical work in hospitals. In *Proc. Second Ubiquitous Healthcare Computing*, 2003
[6] K. Beck. *Smalltalk Best Practice Patterns*. Prentice Hall, 1997
[7] J. Bohn, F. Gartner, H. Vogt. Dependability Issues of Pervasive Computing in a Healthcare Environment. In *Proc. First International Security in Pervasive Computing*, 2003
[8] W. J. Brown *et al*. *AntiPatterns*. Wiley, 1998
[9] A. Cockburn. *Writing Effective Use Cases*. Addison Wesley, 2001
[10] J. O. Coplien. A generative development-process pattern language. In J. O. Coplien, D. C. Schmidt, eds, *Pattern Languages of Program Design*. Addison-Wesley, 1995.
[11] M. Fowler. *Analysis Patterns*. Addison-Wesley, 1997
[12] E. Gamma *et al*. *Design Patterns*. Addison Wesley, 1995
[13] Michael Jackson. *Problem Frames*. Addison Wesley, 2001
[14] Yang Liu. *Security in Ubiquitous Healthcare Systems*. MSc thesis, Dept Computer Science, University of York, UK. 2004
[15] The Open Group. *Guide to Security Patterns*, Draft 1. April 2002
[16] S. Stepney, F. Polack, I. Toyn. An Outline Pattern Language for Z. In D. Bert *et al*, eds. *ZB2003: Third International Conference of B and Z Users, Turku, Finland, June 2003*. LNCS vol 2651, pp 2–19. Springer, 2003
[17] J. M. Vaclav. Protecting Doctors' Identity in Drug Prescription Analysis. *Health Informatics Journal*, December 1998.
[18] Medieval Theories of Analogy. Entry in Stanford Encyclopedia of Philosophy. http://plato.stanford.edu/entries/analogy-medieval/#3
[19] Effective Security Requirements Analysis: HAZOPs and Use Cases. Jill Srivatanakul, John A Clark and Fiona Polack. Proceedings of Information Security, 7th International Conference, ISC 2004. Lecture Notes in Computer Science, Vol. 3225, pp 416-427.
[20] Challenging Formal Specifications with Mutation: A CSP Security Example. Jill Srivratanakul, John Clark, Fiona Polack and Susan Stepney. Proceedings of the 12th IEEE Asia Pacific Software Engineering Conference (APSEC) 2003.