RESEARCH

On middleware for emerging health services

Jatinder Singh* and Jean M Bacon

Abstract

Healthcare concerns have become diverse, ranging from acute and chronic conditions to lifestyle, wellbeing and the prevention of illness. Increasingly, individuals are taking responsibility for monitoring their own conditions. Healthcare technologies are increasingly used not only for administration, but also in specialist treatments and many forms of monitoring, including when a person is mobile. As well as formal interactions with professional carers and results from specialist procedures, care may involve ad hoc interactions with an individual's community. Together, these yield a wealth of data relevant in different contexts. Unfortunately, many existing healthcare systems are inflexible, single-purpose, and self-contained, so that we cannot fully realise their potential. We believe that a framework for flexible interoperability of healthcare-relevant components is crucial, in a time of increasing need from an ageing population.

We present a vision of *pervasive, preventative and personalised healthcare*. To achieve this we believe that the application logic embodied in components should be separated from the policy that specifies where and how they should be used. This may be in ways not contemplated by their original designers. Middleware should therefore provide a framework that supports not only traditional communication among components but also dynamic reconfiguration of components in response to circumstances that arise, with the management and enforcement of high-level policy integrated with the middleware. By this means, functionality for patients, carers and health administrators can be customised and provided as, when and where required.

This paper explores middleware requirements and challenges arising from technology- and population-driven developments in healthcare provision. We describe the specific requirements that middleware must address, and present some practical steps towards addressing these from the initial stages of a middleware (SBUS).

Keywords: healthcare; assisted living; middleware; policy enforcement; dynamic reconfiguration; event-based systems; pervasive computing; security; privacy; SBUS

1 Introduction

With the world's ageing population, there is an increased burden on healthcare resources [1]. As such, there is a global push to improve the efficiencies and effectiveness of health care services. The goal is to improve patient outcomes, alleviate chronic conditions, and more generally improve an individual's quality of life [2]. This in turn reduces healthcare expenditure, as there is less reliance on more costly health services [3].

Traditionally, healthcare provision focuses on acute issues. These are significant, highly symptomatic, health concerns that often require urgent treatment, such as a heart attack or stroke. Such incidents are often serious, greatly impacting a patient's life, and consume significant health resources. Health services are evolving to take a more on-going, preventative approach to care [4]. This involves risk management, taking active steps to mitigate potential health concerns. These steps can be clinical interventions, such as medication or periodic diagnostic tests, as well as lifestyle factors, such as exercise, diet, stress-levels, and so forth.

Information is the key to improving health and wellbeing [5]. Clearly, better informed practitioners give a better quality of care. Technological advances, such as sensors and monitoring technologies, means that there is increasingly more data available than ever before. The more data that medical professionals have access to, for example on a patient's physiology, lifestyle, other medical interventions, the more effective advice and treatment regimes can be. This enables care to be better customised to the individual and their circumstances, and also contributes to improving general care standards by providing input into case-studies, best practices and medical research.

*Correspondence: jatinder.singh@cl.cam.ac.uk
Computer Laboratory
University of Cambridge, UK
Full list of author information is available at the end of the article

Information is relevant not only to the medical professional. An explicit goal of emerging care models is to empower patients to take control of their health, well-being and any conditions that they may have [6]. Therefore, as more data becomes available, it is important for patients, and their care community (e.g. family members, social carers) to be better informed, since more understanding of their conditions and risks can help to reduce risk and lead to positive change.

Information technology already plays an important role in healthcare, though the focus so far has predominantly been to support the acute-based approach to care. Thus, IT systems tend to concern clinical aspects, providing health-enterprise services, such as management tools, patient administration and health records (to various degrees), or a particular diagnostic capability.[1] Many of these systems are bespoke; only for specific purposes, and operate in vertical silos. If interoperable, there tends only to be integration with other local systems (cf. across care units) or the major systems– often an expensive undertaking and therefore driven by management concerns, rather than focusing on more flexible care pathways.

The way forward is to realise the vision of *personalised, preventative care* [7]. This involves a more holistic approach, where healthcare not only occurs within the traditional, clinical space, but also informally through patients and their care community in an on-going, day-to-day manner. Technology is central to this vision. With respect to data, sensor technologies are increasingly providing rich streams of information giving valuable insight into a person's daily activities, progress and well-being. Advances in communication technologies enable interactions at a distance (remote care), and improvements in connectivity footprints provide the means for patients to be 'continually online' for monitoring. The uptake of mobile devices by both patients and medical professionals promises their increasing familiarity with technology [8], providing a highly personal mechanism that can be leveraged to communicate, inform, alert and monitor. From the clinical perspective, this will lead to better informed care, more efficient use of carer time, and improvements in the speed and quality of response. Continuous monitoring of environmental and physiological state provides rich datasets to aid diagnosis through analysis of risks, traits and trends. Monitoring can be extended to provide alerts of detected conditions, for reminders, etc. This is important for patients and their care communities, to give them insight, feedback support and reinforcement of their care goals. As well as providing rich data for clinical use, to customise care and diagnosis, this data is also useful for medical research.

There are systems for remote care (telecare) and self-monitoring.[2] However, these tend to be bespoke, closed systems, designed for specific conditions or exclusive concerns, such as raising particular alerts, or diabetes management. While a step in the right direction, to properly realise the goal of preventative care more integrated, flexible approaches are required. The systems should be able to deal with the idiosyncrasies of individual patients, their care needs, their support networks and their care team, and capable of leveraging a range of different health systems, services and data as appropriate.

The vision for a system environment should therefore be *pervasive healthcare* [11], rather than a number of specific systems (integrated to various degrees). That is, a number of systems—including applications, services, sensors, etc.—should come together, seamlessly, when and where relevant, as appropriate for each individual. Data drives health and well-being processes, enabling tailored interactions across clinical and more informal and/or personal systems. Care efficiencies are improved, as integrated systems and improved communications help the movement away from the environment of manual, human-initiated actions.

Pervasive healthcare imposes significant challenges for supporting infrastructure. Firstly, in such an environment, it will be less clear which systems are health related; heart-rate sensors are obviously relevant but, for some people, lack of phone communication could indicate depression. The availability and relevance of different systems will vary between individuals. Each user, clinical or otherwise, is different, they will vary in how, when and why they use particular systems. Further, security and privacy concerns pervade, given the sensitivity of personal/health information. However, such sensitivity may depend on context.

Perhaps most importantly, system developers will not be able to envisage all the circumstances in which their systems may be used. Therefore, these concerns can no longer all be encoded in application logic. Indeed, this has led to today's vertical silos and interoperability problems. Useful functionality comes not from single systems, but by coordinating systems in various ways.

Middleware provides a layer of abstraction that mediates between applications and network infrastructures [12]. Often described as *systems-glue*, middleware is crucial in supporting this emerging healthcare because middleware operates *across systems*, to assist

[1] See `systems.hscic.gov.uk` (21 Mar 2014) for a flavour of the many systems in use by the National Health Service (UK).

[2] For background information and examples, see [9] and [10].

communication and management. The challenges of pervasive healthcare means that middleware, in addition to its traditional role of enabling interoperability, must be more active in controlling systems and driving interactions between them, when and where appropriate, in order to meet user-defined goals. Policy, representing an individual's (patient and/or carer) requirements, is crucial; defined outside application-logic, it operates to orchestrate and reconfigure system components to meet particular goals. The purpose is to allow people (and their carers) to customise how and when various systems are used, and how their data is handled, in order to best manage patients' health and well-being.

This paper explores the impact of this emerging healthcare vision and the requirements this model imposes on supporting infrastructure. We begin by describing the evolution of healthcare provision (§2), and the resulting systems requirements (§3), before emphasising the importance of policy for enabling dynamic management across system components (§4). We then explore areas of systems research, and consider their fit within the broader healthcare vision (§5). The SBUS middleware is introduced as proof of concept to practically illustrate the types of capabilities required by emerging health infrastructure (§6). Policy enforcement specifics are then explored (§7), and we conclude with a summary of open challenges (§8).

2 The evolution of care

The burden on health resources is ever increasing, given the world's ageing population [1]. The majority of care services relate to *chronic* conditions. These are on-going, long-term health conditions or diseases, such as heart disease, cancer, diabetes, etc. The World Health Organization shows that 75% of the total population will develop one chronic condition and 50% two or more conditions [2]. As people live longer, the number of people living with chronic health conditions is increasing. Chronic conditions, particularly when improperly managed, can lead to acute episodes requiring urgent medical attention.

The worldwide goal is to improve people's health, well-being and quality of life, and at the same time, reduce the burden on health resources. *Preventative care* fundamentally involves taking active steps to reduce the risk of developing acute and chronic conditions, and also improving the management of chronic conditions (and/or the factors that can lead to chronic conditions). In addition to clinical aspects, such as diagnostic services and medication, lifestyle factors, such as smoking, alcohol intake, exercise and diet directly impact many aspects of health. Lifestyle management can often limit the onset and exacerbation of both acute and chronic conditions.

It is clear that such an evolution in the care paradigm would lead to more informal care practices; that is, managing aspects of health and well-being outside the formal, traditional clinical space. In general terms, this entails: a) establishing a patient support network, including professional and informal carers (friends, community and family), b) assisting with everyday activities, c) ongoing observation to indicate any improvement or deterioration in condition(s), d) educating and informing patients on how to 'help themselves', and e) enabling (formal) interventions when and where required. This approach would lead to a much wider coverage by healthcare services.

Healthcare systems

Currently, a vast number of systems underpin clinical health services. In formal care environments, such as a hospital ward or a surgical theatre, the system components relating to care tend to be standardised, used for a particular purpose. Some are more general, enterprise-level systems such as electronic health records, designed to improve communication between medical professionals.

Others are more specialist, e.g. monitoring a particular set of vital signs or the treatment flow of a particular condition. Many of these systems operate standalone, or have few, specific interactions with major services, such as National Care Records or the patient administration systems (PAS) in the UK.[3] From the clinical perspective, the lack of integrated technologies means much inefficiency: actions and interventions are manually initiated, e.g. doctors must log-on and switch between many different systems, and communication is typically synchronous (pagers that require follow-up phone calls), requiring staff to be available and reachable. Preventative care adds another dimension to the healthcare technology landscape; it requires systems to be *patient-centric*, rather than enterprise-centric, that can be customised to the user—be they the patient or carer.

Technology can enable a more holistic approach to health, in which a wide variety of systems, sensors and applications come together, when and where appropriate, to achieve particular care-related outcomes. Some of these will be person-centric, providing monitoring of aspects of health, lifestyle and well-being, to give information and feedback to enable positive change. Others, aimed at medical practitioners, will facilitate their communication with others, direct access to monitoring devices, and support their interactions with more formal care systems. However, it seems that information flows should not be partitioned between clinical

[3] For specifics, see `systems.hscic.gov.uk` (21 Mar 2014).

and informal. Rather, the line between the formal and informal care processes blurs, as person-centric systems may interact with more formal systems, when and where necessary.

Such a vision is realistic. Monitoring and communications technology have developed to the extent that they are potentially exploitable in large-scale, widespread healthcare. At the same time, people (patients and medical professionals) are becoming increasingly familiar with and accepting of a wide range of technologies [8, 13]. Thus, there is scope for more mobile communications and sensor technologies to help support this vision of pervasive healthcare. There are UK Government initiatives to support assisted living developments via the Technology Strategy Board[4] and the UK Research Councils' Healthcare Technology theme.[5] The EU proposes to continue its Ambient Assisted Living Joint Programme (AALJP).[6]

However, there are significant barriers that must be overcome.[7] Most current systems, enterprise and diagnostic, including research systems, still operate in vertical silos, with little interoperability except between major systems. These limited interactions tend to be specified as part of the procurement contract. This will become more of an issue moving forward, where a plethora of new systems will become relevant to providing care. Even initial movements towards supporting informal care processes suffer from the same issues: concerning a single standalone application or closed service, focusing on a particular aspect of care, e.g. a monitoring infrastructure for a specific condition or a "panic button". It is said that a panic button service tends to restrict the movement of the elderly who are afraid of falling to its 50 metre range. This greatly impacts quality of life. Enabling such a person seamlessly to raise an alarm when inside the home or when out and about is the vision of an integrated service.

To realise pervasive healthcare, system components need to operate across application-level boundaries, to enable wide-ranging and customisable health and well-being services — which can seamlessly include assisted-living, formal and informal care processes. Interactions may occur across administrative domains, e.g. to include doctors, insurance companies, and so forth. Further, the same system components or services can be used for different purposes. For instance, heart rate data can be useful for the patient when exercising, for a General Practitioner (GP) for diagnosis, and for a paramedic in an emergency. Also, systems designed for other purposes could be integrated for certain individuals to provide health-relevant data, e.g. an application monitoring television viewing habits might have been designed for targeted advertising purposes, but can also contribute to a view on psychological well-being.

We now explore the specific requirements that the emerging care model imposes on supporting infrastructure.

3 Healthcare infrastructure requirements

As argued in §1, information underpins the emerging care model since information can be used and combined for a range of purposes. This leads to appropriate, timely interventions to achieve better health and well-being. Thus, from a systems-perspective, requirements concern data: production, consumption, communication and management. As technology becomes increasingly pervasive, there will be a rapid growth in the number of data sources/sinks (forming the *components* of systems and services). The use of these components will vary depending on user preferences and environmental context. Some will be relevant to clinical concerns, some will relate to patients and/or informal aspects, and some will be useful for both. Thus, the infrastructure that supports care must facilitate and provide data availability across a highly variable range of applications and services, as well as providing mechanisms for dynamic and flexible management.

Much of the responsibility for managing these concerns falls on middleware. By mediating the communications between system components, middleware "glues" them together. At present, most middleware focuses on enabling communication at the request of applications. Indeed, middleware abstracts various communication specifics but typically, applications still control how and when they connect to and interact with peers, brokers or services. In this section, we show how the emerging care environment extends this traditional role of middleware to include the dynamic composition of systems to support requirements in real-time.

3.1 Supporting data sources and monitoring

Healthcare is data driven; the more informed medical professionals, patients and their care community, the better the outcomes. *Big data* will become increasingly important to healthcare, where data and analytics can lead to improved diagnoses, decision making, treatments, and responses [15].

Monitoring people and their environments is central to the vision of personalised, preventative care. Sensing technologies will operate to generate rich representations of an individual's physiological state, and also

[4] www.innovateuk.org/healthcare (21 Mar 2014)
[5] www.epsrc.ac.uk/research/ourportfolio/themes (21 Mar 2014)
[6] www.aal-europe.eu (21 Mar 2014)
[7] See [14] for a roadmap of general issues concerning assisted living.

to generate information concerning physical environments (which might relate to a static location, such as a particular room in a house, or the current location of a mobile user). This data can be filtered, aggregated and interpreted, thus driving positive steps for managing and improving health and well-being.

There are a wide range of data sources that can be used for patient monitoring. Data representing different aspects of state will be produced by various user devices, such as mobile phones, body and environmental sensors/actuators, as well as online services, databases, etc. The information used may include location, movement, ECG, environmental context (e.g., noise levels, temperature, weather, pollen levels, etc.), inference of social context, messages, calendar events, and many more. In addition, there are also knowledge-base sources aimed at the general public, such as NHS Direct,[8] WebMD,[9] etc., that can help determine the relevance of particular information, how to interpret results, and the appropriate interventions to take.

Note that personal and mobile technologies have specific data management and communication concerns, different from those of standard clinical systems (which tend to reduce to standard enterprise systems, often client-server). Further, each data source may be relevant to meeting a number of different care goals; ECG sensors and pulse-rate monitor outputs can be used by individuals, their GPs and paramedics under different circumstances. As argued in §2, a wealth of data not designed for specific health purposes is available and relevant for some individuals, such as (changes in) phone or TV usage.

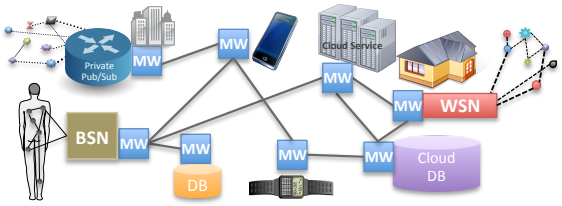


Figure 1 Systems-wide middleware integrates a range of systems and networks.

Thus, it is particularly important that systems do not function within a single vertical silo or product ecosystem, whether through technical constraints (e.g. only operating with other products by the same producers) or functionality (e.g. only operating with other systems with the aim of targeting a particular goal). Each individual will differ on the systems they use, as determined by their conditions, care budget, physical environment, care community, etc. System components should be able to address a range of concerns, operating across products, services and management domains. As Fig. 1 illustrates, middleware can integrate a wide range of systems and services.

Clearly some of these concerns align to those of the *internet of things* (IoT) [16], in which a myriad of devices, many representing everyday objects, are available online. This is because a vast number of data sources, sinks and services have the potential to be relevant to health and well-being. Indeed, the supporting infrastructure explored in this paper can certainly be used in a wider IoT-context; however, given the inherently stringent requirements of healthcare, the management and control capabilities of the infrastructure will be above and beyond the connectivity and interaction concerns of most IoT application scenarios.

Requirement 1: *Future healthcare requires a generic middleware capable of supporting current and future technology and the types of data produced.*

3.2 Supporting multiple communication paradigms

It is clear that healthcare is data driven. In a preventative care environment, pervasive systems offer functionality by means of system components *interacting*, exchanging data to realise system functionality. The supporting infrastructure must therefore be sufficiently flexible to support interoperability across a range of components in different scenarios. This imposes a number of considerations for the supporting middleware.

Firstly, the middleware must support a number of *interaction paradigms* (data exchange patterns). The vast majority of health-focused systems, like most systems in general, operate in a request/reply or Remote Procedure Call (RPC) manner. RPC interactions are important, for instance, to query a patient record, access a historical log of heart rate data to aid diagnosis, or to perform particular processing operations (e.g. calculating BMI). However, the emerging care environment will necessarily involve live data streams, the natural form of monitored data. Such data will be analysed to detect significant incidents and raise alerts, e.g. recognising a fall or a heart-attack and raising an alarm. Supporting infrastructure must therefore be able to support the range of interaction paradigms.

Given healthcare is data driven, it naturally maps to an event-based approach. An *event* can be defined as a data-rich encapsulation of some occurrence. Events can operate in a RPC manner, by one event encapsulating the request and another the response, and within a stream representing an individual data point. Events can also encapsulate a higher level of meaning

[8] www.nhsdirect.nhs.uk (21 Mar 2014)
[9] www.webmd.com (21 Mar 2014)

and/or context. An event-based model makes sense for an emerging care environment, as it can represent everything from an alert (perceived emergency), data reading (e.g. clinical result, ECG sensor reading), health-record query/response, as well as more benign, system-relevant occurrences, such as a person entering a hospital, changing their privacy preferences, etc. The communication of events can greatly improve care efficiency, by reducing the need and reliance on manual human-imitated actions.

Events can also pave the way for flexible interaction paradigms. For instance, some components, particularly user-facing applications, will actively drive their interactions with others, e.g. a patient accessing their record entails a query to a particular server. Events can represent the connection, query, processing and/or response. This (RPC) is the predominant form of interaction today. Other components are passive, e.g. a body-sensor system focusing on producing vital-sign data, having no concern as to potential consumers.

It is clear that service discovery is a critical issue for infrastructure that aims to support the emerging care environment. In order to break the application silos prevalent in healthcare, it must be possible to discover the relevant data sources and sinks with which to interact dynamically, at runtime. This is particularly important in a pervasive healthcare scenario, as many interactions are not predetermined, but rather the relevant system components and forms of interaction are dictated by user preference and the particular circumstances. Thus, unlike much of today's communication which involves interactions with particular addresses, there must also be mechanisms for finding components based on the data they serve (and other attributes).

Event-based modelling also assists service discovery. If events are typed, it means service discovery (and hence interactions) are based on *the data they provide*, rather than a specific address and/or particular known systems. Further, type systems pave the way for further description of events and data that a particular component handles, as well as details about the component itself. This facilitates more formal descriptions, the semantics of which can assist in managing dynamism and heterogeneity [17].

Requirement 2: *Future healthcare requires a middleware capable of supporting a wide variety of communication patterns. Event-based middleware is most appropriate.*

3.3 Security-aware middleware

Health data information is inherently sensitive. Such concerns are exacerbated in pervasive computing environments, as these involve extremely detailed physiological and environmental data.

The current approach of encoding security policy within systems themselves is inadequate, as each system will vary in its definitions, capabilities, protection measures, etc. First, this hinders usability; leading to different login credentials for different systems, which many NHS doctors rightly complain about and often subvert by sharing sessions between users or taping passwords to machines. It also hinders accountability. Secondly, the governance capabilities of one system might vary greatly from that of another. Further, having security policy encoded in several systems for local enforcement, often repeated and represented in different ways, increases complexity and leads to mistakes. Therefore, governance regimes must apply *across* a range of technologies and locations.

Middleware deals with interactions, across systems. By making the middleware security aware, it means that governance policy can be generalised across a range of applications. For instance, the same policy can apply to a doctor's access to a Patient Record System, Hospital Administration System and a patient's live data-feed.

Therefore, it is important that middleware provides the mechanisms for enabling security. Specifically, this concerns: a) when and how components may communicate, b) the prevention of eavesdropping, c) information flow management, d) service discovery, e) audit.

The first entails access control policies, which can set up the privileges to enable connections, as well as actively intervene with respect to an interaction, e.g. forcing or closing connections when appropriate. To support this, components must be identifiable so correct privileges can be defined. It is important that the infrastructure supports encryption, to protect information in transit. Information flow management means having the ability to trace where information has flowed from and to where, and also to set certain constraints on its flow at runtime (e.g. it must not pass certain boundaries, or must touch certain systems before flowing to others)–see [18]. Sometimes uncovering the mere existence of a service associated with a patient may be damaging, e.g. services relating to sexual health; and as pervasive health will involve many personal services (surrounding an individual user), it is likely that many of these should be hidden from others. Thus, service discovery processes must also have the possibility of being regulated. Finally, audit gives visibility, which not only enables the ability to unpick the past, but also provides a normative form of behaviour control (the knowledge of being watched can prevent misuse). Audit is crucial to supporting governance in the health services. Such concerns are particularly important in this pervasive environment, as interactions will occur ad hoc, with potentially any source.

Requirement 3: *Future healthcare requires security-capable middleware*

3.4 Supporting dynamic reconfiguration and adaptation

We have described how the emerging care environment entails many different system components interacting with each other in various ways, to achieve various outcomes. A rigid workflow is no longer an option and systems must be configured to meet individuals' needs. Further, it will not be possible to predetermine all the required components and interactions when tailoring to individuals. Thus, the role and functionality of middleware must be extended to include mechanisms to drive and control the interactions between components dynamically. To achieve this, it must be possible to specify and enforce, at the middleware level, how and when interactions should occur.

In practical terms, this means that there must be a mechanism for the middleware to reconfigure systems at runtime, adapting to circumstances as they occur. Given that middleware provides communication, this means the ability to manage runtime interactions. The key is the ability to control connections between components. At present, most middleware leaves this to the applications, as discussed above.

Enabling runtime connection management brings much flexibility; for example, to control the properties of an interaction (i.e. who is talking to whom), perhaps they should interact with someone else, perhaps some change in context implies a different set of interactions should take place, in addition to lower-level concerns, e.g. component failure. Initiating (or ceasing) an interaction is one aspect of this. Reconfiguration is also highly relevant to security, as it also involves changing the properties of the components themselves; security policy will certainly be dynamic, and thus the system must be able to manage components and their interactions to properly effect this properly — a simple example may be "nobody may access my location information except in an emergency, when my doctor can". This means the location component must be able to enforce such a constraint (either directly, or by being 'told' what to do) as appropriate.

Such scenarios involve components relying on others to initiate and manage interactions on their behalf. Middleware must therefore support both direct and third-party initiated interactions. The latter is particularly important, bringing flexibility as it allows interactions to be managed outside component application-logic.

In summary, middleware for the emerging care environment must not only support the interoperability between systems, but also the means to drive those interactions, and reconfigure the system as appropriate to the circumstances. However, middleware provides merely the mechanism. It is *policy* that will manage and define these orchestrations, coordinating between the various components, often across application and administrative boundaries, when and where appropriate in order to meet particular functionality goals.

Requirement 4: *Future healthcare requires a middleware capable of effecting dynamic reconfiguration of system components in response to event-driven policy*

The next section focuses on how policy can be expressed and enforced to meet this requirement.

4 The role of policy

Policy-based systems operate to effect some change (by taking actions) in response to particular happenings or circumstances. It is the role of policy to express these circumstances and actions.

Traditionally, the policy used by middleware targets network management, resource allocation and/or quality of service e.g. dealing with node failure, or allocating sufficient resources as requested by an application. However, as technology becomes increasingly pervasive and connected, it is important that policy encapsulates higher-level concerns, to bring about user-level goals. The emerging healthcare environment takes advantage of functionality coming from a number of different systems, used in different ways, by different people.

The focus is no longer solely on managing resources to serve a particular application requirement, but also on managing ranges of system components across application boundaries. More specifically, policy can trigger, or regulate user/application actions or behaviour, or react/respond to data generation and inference. Such actions are aspects of *coordination*, operating to mediate and orchestrate components to meet high-level functional goals. Coordination may include *actively* generating data or starting/stopping an interaction e.g. emergency action is to send an alert message to paramedics, and to set up a voice connection to a relative. Coordination may also enable (or prevent) some *potential* interaction, e.g. authorising a carer to access sensor devices when visiting a home.

Effecting such coordination requires the means to control components from *outside their application logic*. That is, enabling reconfigurations initiated by third-parties. This means that it is no longer the individual components that must manage and enforce policy, but rather, the components can be instructed (by those who are 'policy aware') to take specific actions to meet user goals. This is key to enabling systems to be user-driven, to coordinate component use to meet specific, individual concerns. It also allows new functional possibilities, where system components can be

used/reused in various ways, not envisaged by the original developers. Clearly, this becomes increasingly important in a pervasive health environment which needs to account for individual preferences, user mobility entailing interactions in completely new environments, and the fact that care interventions are event-based - requiring response on certain happenings. All of these could result in different component compositions and interactions in different circumstances.

In summary, policy in a middleware context represents user specified goals and functional concerns, meeting these by operating to **a) bring functionality**, and **b) regulate/control** within and across applications, by reconfiguring the system/environment.

Fig. 2 shows a policy-based reconfiguration for an emergency situation concerning an elderly male patient. In the general case, some of the patient's data is streamed to his phone to provide feedback on his current state. Data is also stored persistently to assist in diagnosis and prognosis. In an emergency situation, policy exists to bring help. This entails alerting his wife and the emergency services (A&E) of the incident; sending vital signs streams directly to A&E so they have a better indication of the situation and progression; and relaxing privilege constraints on various data sources to enable those providing help to access data that may assist.

4.1 Context and events

Context is crucial to policy enforcement; policy combined with context determines when various systems are brought together. Since the emerging care environment is user-centric, the infrastructure and environment must be tailored to individuals. Patients (with help from their support network), informal carers and medical professionals will all have preferences as to how the services supporting them are managed and controlled. These preferences form the *policy* that manages and governs systems environments.

In a pervasive care environment, context can be derived from many aspects: treatment/intervention regimes, practitioner interactions, feedback from physiological (e.g. heart rate, movement) and environmental (e.g. location, pollution levels) sensors, interactions with clinical services (e.g. querying electronic health records), etc.

Policy is context-sensitive, e.g. a patient may say that nobody can access their location, except in an emergency, or "only my treating doctor can access my vital-signs data"; or a practitioner may only access clinical systems when they are physically present in a hospital ward.

Policy actions are tied to context, where changes in context lead to defined responses. For example, the results from a laboratory test will define a particular care pathway (treatment plan); detection of some anomaly leads to referral to a specialist. A timer might remind a patient to take medication, a doctor leaving a ward or going "off duty" might lose certain access rights, an alert might be sent on detection that someone has fallen, a doctor querying a health record must be authorised and audited.

As established above, healthcare is data driven, and thus is amenable to an *event-driven* systems approach. Information can generally be broken down into particular events (occurrences) that individually, or in aggregate, through event composition, represent some significant change in state. For example, an event might represent a single occurrence, such as someone entering a ward, taking a reading, taking medication, leaving home; or can be composed with other events to represent some higher-level event, e.g. a significant rise in heart rate with a drop in blood pressure could mean a blood-loss event, or leaving home without taking medication should result in an extra reminder. Events can also represent concerns at various levels, e.g. systems level concerns such as a loss of connectivity, or higher-level concerns, such as a user action (e.g. pressing a panic button, initiating a query, entering a hospital). Policy rules can be defined to respond to these events, by dictating the components that could or should interact and the associated security and governance concerns.

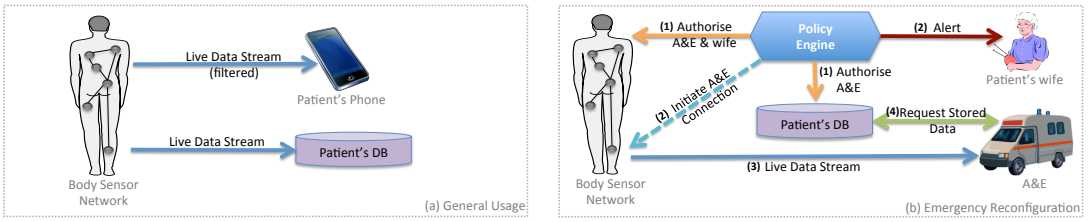
In Fig. 2, the reconfiguration is triggered by an emergency. This could be represented by a single event that may have been derived through analysis of the underlying sources, e.g. a rapid accelerometer change followed by some period without subsequent movement (GPS, indoor positioning) suggests a serious fall.

We have argued that an event-based representation of context is particularly suitable for emerging healthcare. Firstly, events facilitate a response. Aside from the more complex examples listed, in an environment where information is sensitive, even the most basic happenings (data access, sensor samples) require some sort of response, such as being stored persistently or logged. Events provide well-defined hooks enabling response actions to occur. Secondly, an event-based approach facilitates context awareness in the middleware. Middleware concerns data communication, while events encapsulate data (e.g. a message can neatly encapsulate the data for an occurrence); thus rather than merely managing the conduit, the middleware can be active with respect to the data it handles.

4.2 Policy actions and engines

We have identified three general types of policy action relevant to healthcare. These are:

Reconfiguration: Taking a configuration action,



(a) General usage (b) Emergency response

Figure 2 Policy driven reconfiguration reacting to an emergency situation.

such as initiating/removing a connection or changing the privilege of particular components.

Event production: Generating an event to transfer some information, e.g. to raise an alert, inform of some happening, or simply respond to a query.

Policy Management: Policies are contextual, thus a change in state might change the set of active (applicable) policies, e.g. a set of restrictive privacy policies may be relaxed in a medical emergency, making more components visible to an entity.

To effect these actions, a) the communications support in the middleware must allow, through the appropriate hooks and interfaces, reconfigurations to be initiated by third-parties, external to the application logic of the components being affected; b) the components enforcing policy should be integrated into the same middleware, in order to receive events (context) to evaluate policy rules, produce new events, and so forth.

To manage policy, there is a requirement for system components that are dedicated to managing the policy process. *Policy engines* are services that encapsulate, and enforce, a set of policies. Policies often take the form of *Event-Condition-Action (ECA)* rules [19]. A policy engine will watch for particular events corresponding to its ruleset. On the occurrence of these events (which may be composite), the action selected by the policy rule will be executed. Some events might trigger a change in the active rules, while others might produce events, or issue the middleware with particular reconfiguration operations for particular components.

Fig. 2 shows these actions. Event production policy enables the alerting of the patient's wife (step 2). The reconfiguration alters privileges on the database and sensor network (step 1), so that the wife can access sensor information (such as her husband's location) and A&E has the ability to access both the live sensor information, and the patient database in case historical information is relevant. A connection is then automatically established between the sensor network and A&E (step 3) so that A&E is immediately made aware of the patient's current state and ongoing progression. More generally, policy exists to relax privileges in an emergency situation to aid response; thus the policy engine will deactivate some of the more restrictive policies (not illustrated in the figure). A&E may then, manually, query the database for any relevant historical data (step 4), made possible by the earlier privilege reconfiguration (step 1).

4.3 Taking a policy-driven approach

So far, we have seen the following advantages from taking a policy driven approach. Firstly, it can break down vertical application silos. Operating outside application logic, and across system boundaries, it allows for more functionality and flexibility, enabling components to be used and reused in ways not previously possible. It allows for fine-grained personalisation and customisation, responsive to a range of different circumstances. Such functionality is crucial for systems supporting emerging care services.

In addition, this approach greatly reduces the burden on application/service developers, and the likelihood of errors. This is because developers need not account for individual users' requirements by having some mechanism for them to specify and encapsulate their preferences, and then, only from the concerns foreseen by the developers. Also, the burden of maintaining the details of all potential contexts in which components can operate is removed from developers. This facilitates component use/reuse, paving the way for new components to be integrated with minimal configuration. Otherwise, developers would need to enumerate all possible circumstances in which particular systems might be used, including the specifics of the operating environment, such as nearby components and interaction protocols.

The use of policy engines means fewer policy definition/enforcement points, which can simplify management. The alternative, of having each application maintain and enforce its own policy leads to policy gaps and inflexibility since systems will be limited in the policy functionality they provide. In practice, components would only provide for specified policy

that they deem relevant. Further, trying to encapsulate cross-system functionality in each application itself is not only impractical, but is prone to errors, particularly as different applications may have different mechanisms for accounting for preferences. All of this is a major concern in healthcare where governance is central.

There will be a number of policy engines operating simultaneously within various environments. Issues of conflict arising from competing engines poses a significant challenge, and requires careful consideration. However, the scope and reach of the policy engines can lead to a natural resolution. In practice, a policy engine should only effect actions on components that authorise it to do so, which models the real-world in the sense that only those who 'own' an entity can control it. This inherently limits the scope for conflict. For example, there may be a policy engine managing a hospital ward, and one on a doctor's phone. The ward's engine will set the baseline for interaction, e.g. who can access the various monitoring equipment, whereas the doctor's engine will reflect the doctor's individual preferences regarding when it connects to particular components within the ward. Having an engine that encapsulates policy aids policy management. The alternative is to have policy fragmented among different components, using and limited by the expression mechanism of each.

Given that policy entails some components controlling others, trust becomes an issue. To function effectively, there needs to be the ability to regulate who can access, and reconfigure, particular components in specified circumstances. And such regulation must, itself, be dynamically reconfigurable. Further, there is often the requirement for audit, not only of the actions taken but also the policy applied. Such concerns will vary depending on the circumstances and environment. Indeed, issues of trust can be encoded in policy, which if the middleware supplies appropriate security/governance mechanisms, can be implemented through similar mechanisms to effect control over management processes.

Overhead is another concern of policy-based systems. Given that policy enforcement involves rule evaluation, based on particular events (communicated by data flows), which also involve reconfigurations, there will naturally be some impact on performance. However, this must be considered with respect to the flexibility of the approach. We explore this more in §7.3.

4.4 Policy summary

Middleware is required that enables policy to coordinate the components in the system. Policy-enforcing middleware allows control *across components*, irrespective of their individual logic. This is crucial as it enables user goals to drive the system, bringing components together in particular circumstances to meet users' requirements. It enables components to be used/reused for a number of purposes, facilitates personalisation/customisation, and naturally helps to group related policy through dedicated policy components.

From the middleware perspective, this approach moves middleware beyond its traditional role of enabling communication. Instead, it provides the ability for components to be managed by third parties, and means that middleware can actively drive *how* and *when* communication occurs.

This is important for emerging healthcare, which requires a pervasive systems environment. A policy-driven approach allows flexible functionality, operating across systems, tailored to individuals, in order to effect a range of health-related goals.

5 Moving towards the vision

So far, we have outlined the requirements that the emerging model of healthcare imposes on supporting infrastructure, arguing for a policy-based middleware with dynamic, third-party initiated reconfigurability. There is no single solution or approach that addresses all these challenges. However, there has been work in a number of areas that provides a solid foundation on which to build. Rather than providing an exhaustive literature review, we instead highlight general areas of relevant systems research and their fit within the broader vision of pervasive health.

We have reached these requirements for future middleware through focussing on the healthcare domain. It is interesting to note the overlap with work on generic requirements for emerging middleware, as explored in FOME (Future of Middleware) [20]. For the healthcare domain we concur with the findings of FOME for future complex and dependable systems and our work begins to address the highlighted challenges, in a targeted manner—see §6.

5.1 Service composition and adaptive middleware

Reconfigurable middleware is an area of much research. Middleware that allows configuration and customisation is generally termed *adaptive* (see [21]) or *reflective* (see [22]). Such middleware exposes the current system configuration and state to enable reconfiguration based on inspection. Reconfigurations may come from applications themselves, or involve managing lower-level concerns (e.g. the network) to fulfil application resource or quality of service requirements. Work in this area clearly relevant to pervasive healthcare, reconfiguring lower-level systems concerns

to meet application requirements. However, there is also the need for higher-level controls through policy, not only to serve application requirements, but to work within and across a range of applications to meet users' functional goals.

Related is *service composition* middleware (SC) (see [23]), which combines services to provide particular functionality. This involves taking an application-level task (request) and mapping it to a combination of services [24]. SC considers resource allocation and task distribution and ordering in response to application-specific requests or goals. Again, SC is highly relevant, however healthcare requires the means to *directly control* system components. In a general policy-based middleware, meeting the goals of service composition is only one of the targets that policy could address.

Given the trend towards the internet of things, some recent work considers service composition at scale [25]. Certainly, over time the line will blur between infrastructure for supporting care, and that for more general concerns; however, for the moment, the nature of healthcare imposes specific requirements that must be directly accounted for (§3.1).

Work in this area is ongoing. The authors of [26] are concerned with dependable systems and highlight "better coordination facilities" as a topic for future research. The six research challenges for middleware for future complex systems addressed in [27] include "deriving valid, high-performance configurations of highly configurable infrastructure platforms" and "static configuration and dynamic reconfiguration".

5.2 Sensor networks

Sensor networks (SNs) are well researched, with much literature focusing on networking, addressing the constraints imposed by devices and the operating environment. A lot of work takes place in the context of wireless sensor networks [28], and to a lesser extent body sensor networks [29]. The focus is low-level, considering sensor nodes/devices and their specifics, including data acquisition, resource management (power, memory, and other hardware constraints), node placement, failures, routing protocols, code migration/ deployment, etc. [30, 28]. *Data aggregation* [31] concerns how and when to aggregate and fuse data across nodes to improve efficiency and data richness. Some recent work considers network abstractions and virtualisations [32, 33]. Sensors provide much important data for pervasive healthcare. It is, however, explicitly recognised that sensor networks must be integrated into a broader system infrastructure [30, 33]. Thus, middleware aiming to support emerging healthcare must be able to incorporate (or interface with) a range of sensors and sensor networks (see requirement 1, §3.1). This not only allows sensor data to be processed and consumed throughout the system, but also allows policy to influence sensor network behaviour, such as prioritising the data flows that pertain to an emergency situation.

Generally, our insight is that for future health services, it is not just the infrastructure and resources underlying applications and services that must be configured and managed, but also the applications and services themselves, in terms of how and when they (inter)operate. Configuration at this higher-level is encapsulated by user-defined policy — which can operate across applications, as well as manage lower-level concerns on behalf of applications. The work in adaptive, reflective and SC systems is of direct relevance to the latter, and provides useful insight into realising the former.

5.3 Policy-based systems

There is work that considers high-level policy enforcement [34]. However, these policy models typically involve imposing a particular structure on the environment; for instance, defining entities as managed objects e.g. the *self-managed cells* of *Ponder2* [35], the grouping of agents with trusted controllers [36], and/or other constraints, e.g. a particular form of interaction [37, 38, 39] (see §5.5). Clearly, policy specification and enforcement specifics are relevant; however, the nature of pervasive health systems requires systems to be open and flexible. Thus, any policy-based middleware must deliberately avoid imposing modelling, design or communication constraints such that any structuring is dictated by user and usage requirements.

Reasoning about policy is difficult, especially about consistency when policy is decentralised. A formalism is proposed in [33] which allows for reasoning about access control, especially relating to emergency management.

5.4 Representing context

Policy is contextual, in that it is enforced in particular circumstances. It follows that richer representations of state entail more flexible policy. There are a number of contextual/reasoning models that enable complex state representations [40, 41], which in pervasive environments often involve combining and processing data across many underlying sources. Richer and complex state models provide the means for more powerful, granular and expressive policy rules, which allows for policy that better represents particular healthcare concerns. If the context model is well structured, it enables analysis and reasoning over policy behaviours, useful for determining operational semantics, potential

policy (or contextual) errors or omissions—which one can imagine would be most useful for health services.

Managing context in emerging systems is difficult, due to its inherent dynamic and heterogenous nature. An ontological approach presents a promising way forward [17], as it enables modelling of context, message types, namespaces, etc. based on semantic meaning. This can assist managing dynamism and reconfigurablity, for example the ability to dynamically negotiate a method for system interoperability at runtime [42]. Such work directly supports the policy driven approach we advocate, as policy will impose the constraints and circumstances for facilitating an interaction, in addition to enabling direct interventions to trigger system changes. However, managing semantics is not without its challenges [43], for example agreeing a common vocabulary/coding scheme. This is a long-standing problem even in the more specialist clinical care domain [44]; such concerns will likely be exacerbated by integrating informal care scenarios.

5.5 Communication specifics

Regarding communication (see requirement 2, §3.2), the design and optimisation of interaction paradigms, such as RPC and pub/sub (see [45]) have been well researched. However, we see that much healthcare infrastructure accounts only for a single form of interaction, often RPC. It is important that the supporting middleware enables *both* request-reply and stream-based communication, which is important to serving a range of application requirements.

With respect to communication, there is some work that considers integrating policy into pub/sub infrastructure [38, 39, 46], for enforcement throughout a broker network. Pub/sub, however, is inappropriate as the sole means of communication for emerging healthcare, as it provides only for stream-based interactions (request-reply is cumbersome), and the layer of indirection (event-bus) favours anonymous interactions through a shared communication channel, which is inappropriate for healthcare as it raises identity, security and policy enforcement concerns.

6 SBUS: Middleware towards this aim

Having argued the need for a policy-based approach, we now introduce *SBUS* [47], a middleware designed and developed to support the requirements of emerging healthcare. SBUS provides an open systems framework for securely reconfigurable components. Its main contribution lies in its support for policy-driven reconfiguration of components, and management of their interactions, with fully decentralised management. This enables policy to represent high-level concerns, to operate across applications to achieve users' functional goals.

Here, we present SBUS as a proof-of-concept to give an overview of the concepts, considerations, and design processes for a policy-based approach.[10] The focus is on policy for directly controlling components and their interactions; in moving to wider solutions, such functionality will complement other work, e.g. that concerning contextual representation.

In order to indicate the practicalities of policy-driven healthcare infrastructure, we now provide an overview of SBUS functionality, and follow by describing the means for policy enforcement.

6.1 SBUS architecture (requirement 1, §3.1)

SBUS aims to provide the mechanisms for building and managing complex systems environments. As such, there is deliberately no particular structure imposed on system design. Its design aim is openness and flexibility, to provide the building blocks to enable any structuring required by the user, of applications or the operating environment.

In line with this, and to account for the variability and requirements of the emerging care environment, the architecture allows incorporation of other systems, such as closed or proprietary networks, wireless and body sensor networks and other systems that manage device and resource constraints in specific operating environments. This, of course, requires *gateway* components to export data outside the subnet's environment (see Fig. 1). Closed or proprietary sensor networks usually provide this gateway functionality [30].

6.2 Communication (requirements 1, §3.1 and 2, §3.2)

SBUS is a data-centric communications middleware. Data is encapsulated within a *message* of a specific type. A messaging approach suits healthcare, as a message can neatly encapsulate details of an event.

The basic unit in SBUS is a *component*: an SBUS-enabled process (i.e. an application, service, or part thereof) that uses the middleware to manage its communication. Each component has a number of *endpoints*, which can be thought of as typed communication ports. The endpoints of different components are connected *(mapped)* together to enable communication (Fig. 3).

Each endpoint is associated with a schema (in *LITMUS* [48]) describing the message type(s) it handles. Communication is type safe, the middleware ensuring that mappings only occur between compatible endpoints, i.e. that schema and interaction modes agree, and messages correspond to the schema for that mapping. Type identifiers (hashes) are encoded to make messages self identifying, removing the need for a central type authority. Content-based filters can be imposed on mappings to control message flows.

[10] See [47] for further technical details on SBUS.

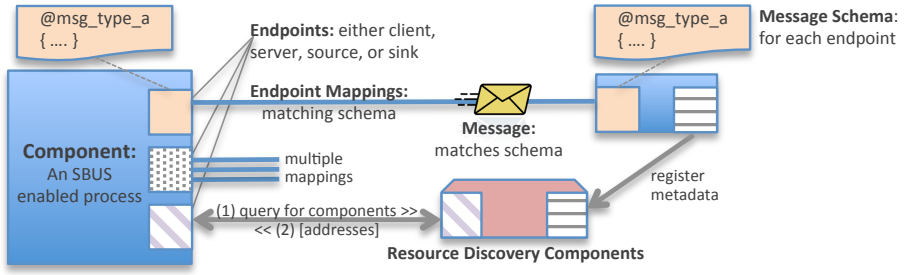


Figure 3 SBUS communications overview

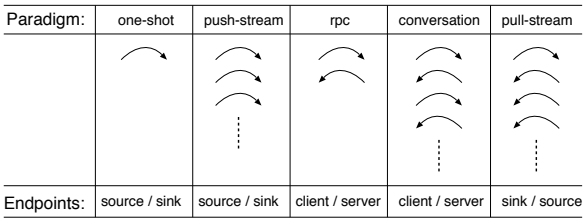


Figure 4 SBUS interaction paradigms, encapsulating both stream and client-server based interactions

It is important for healthcare applications that the infrastructure supports a range of interaction paradigms. Thus, SBUS was designed to support client/server*and* stream-based interactions. An endpoint takes an *interaction mode*: either a *client* (the query issuer) or *server* (returning a result); or for stream communications, either a *source* (producer) or *sink* (consumer). Fig. 4 illustrates the directly supported interaction paradigms. Mappings only occur between endpoints with corresponding types and interaction modes, i.e. sources with sinks (possibly 1-to-many), or clients with servers.

Communication is naturally peer-to-peer and thus the infrastructure is inherently decentralised. Again, this is to provide the building blocks; more complex interaction models can be built where required. For instance, it is simple to implement pub/sub (event-bus) functionality and message-queue brokers to enable indirect and asynchronous communication. Such flexibility is important in supporting pervasive healthcare.

Resource discovery

We have described how resource discovery is a significant concern for emerging healthcare, due to the variability of the computing environment. Components (including policy engines) must be able to find dynamically the appropriate components to interact with, when and where necessary.

For a connection, one requires the network address of the component with which to interact. *Resource Discovery Components* (RDCs) assist by maintaining a directory of active (registered) components in the environment. A component registers with an RDC so it is discoverable by others. The RDC provides a lookup service returning the addresses for components who match a query, somewhat similar to DNS. Components register their *metadata* with RDCs—each describing itself, its functionality, and data handled—thus enabling powerful lookup queries.

The lookup query constraints tend towards two categories:

Identity: Concerns component specifics, such as its class (named-type), instance-name, author, owner, or public key (i.e. when seeking one specific component).
Data: Concerns the data that the component offers, based on endpoint types, interaction modes, etc.

The purpose of the lookup queries is to facilitate the *flexible* discovery of relevant components at runtime, based on the data they serve, aspects of their identity, or some combination thereof. Thus such constraints can apply in combination. The constraints are encapsulated within a mapping operation, so that rather than connection requests necessarily being address-based, they can also be metadata-based. Such queries could look for a component with a particular name (of a particular public key), or any component serving a particular video data, or a component owned by 'Ward 1' that serves patient administration data, etc.

For further flexibility, we have also explored *schema negotiation*, which aims to enable communication between components whose endpoint type schemas only partially match (see [47] for details). This involves discovery query operators that compare of the endpoint type structures of a potential connection, to determine whether a connection can be agreed. If so, SBUS will automatically convert incoming messages into the format the component expects.

This represents only our initial steps regarding negotiation,[11] requiring guidance of the attributes to negotiate. The ability to take account of event semantics enables more powerful, automated interoperability functionality [17]. However, as our initial focus is on SBUS specifics, our approach is an attempt to balance the advantages of strong typing with the flexibility required for dealing with different environments. Such functionality is useful for policy designers who have some knowledge about the components needing to interact; allowing communication that would not otherwise be possible.[12] This is not unrealistic, as there may be some knowledge of the components that could interact; for instance, a user buying new sensor kit, or a hospital by way of procurement.

There may be any number of RDCs in an operating environment. RDCs may be federated and replicate information, e.g. across a national health service. Others may operate within a specific scope, dealing only with a particular set of components, such as those in a patient's house. Again, it is important not to impose any structure on the operating environment; but rather to allow any structuring to come from user requirements. For instance, there may be a number of RDCs in the same environment, several cooperating to manage the lower-level concerns for components of a large-scale distributed application, and one to handle the user-facing services.

The location of an RDC must be known/discoverable: perhaps by running at a well-known address; infrastructure providing the address on connection (e.g. through DHCP options at a low-level, or a policy-engine at a higher level); or by prior knowledge if deliberately obscured. A component maintains a list of RDCs with which it interacts, which is changeable at runtime.

SBUS is decentralised, RDCs exist only to aid interactions. Discovery without RDCs is through *inspection*, where a component is probed to retrieve information via its endpoints and connected peers, enabling service discovery by trawling a connectivity graph. This is useful when an RDC is unavailable, or inappropriate.

To reiterate, SBUS aims to provide a number of different discovery capabilities that can be leveraged as appropriate, in an attempt to address the inherent variability of the emerging systems environment.

SBUS also provides mechanisms to manage disconnections and failures, the technical specifics are given in [47]. We mention this as dynamic reconfiguration around failures are crucial for pervasive health, for example, in providing a seamless/continuous service to the sick and vulnerable.

[11] See [49] for a survey of various techniques.

[12] By policy designers we mean those who manage the environment or user preferences and/or system deployment—not general application developers.

6.3 Security (requirement 3, of §3.3)

The SBUS security model enables the protection and control of middleware operations. These complement application-specific security mechanisms, e.g. clinician log-on services, or biometric protection for mobile devices. The security mechanisms fall into three categories:

Transmission. Given SBUS is peer-to-peer, control is intuitive because communication is directed. This differs from an event-bus approach where a shared communication channel potentially allows many components to see the same message. To protect the data (messages) and metadata (e.g. protocol state) from eavesdropping at lower network layers, *Transport Layer Security (TLS)* [50] is used. Before any SBUS communication, components exchange certificates, which after validation are used to create a secure communication channel.

Access Control. Each component maintains an *access control list (ACL)* for each endpoint describing the components that may connect. A mapping is established only if each peer authorises the other.

All privileges can be dynamically changed; when this occurs, all mappings are examined to determine whether they remain authorised—if not, the connection is closed.

Access control policy is defined for a component by its class (self-described type), instance name and/or public key. This enables a range of specificity, allowing security policy and component discovery queries to apply to a particular component, or a group. Coupling component identity to certificates enables strong authentication, which is important to ensure that the access control policy is applied to the correct components.

The result is a regulated namespace, which is appropriate for healthcare; e.g. if the name of the component should encapsulate a patient-ID, instance names must be governed. This is important in environments such as healthcare where data is inherently sensitive. If components do not specify a certificate, and thus cannot be authenticated, by default they may only interact with remote endpoints without access control constraints (world-readable).

Of course, the ACL can be extended to incorporate other component metadata or even other authentication systems.

Filtering: There are cases where a particular component should only receive *some* of the messages emitted from an endpoint. To effect this, filters can be imposed

`map(map_params)`	Establishes a mapping between endpoints.
`unmap(map_params)`	Terminates a mapping.
`divert(divert_params)`	Moves an endpoint's mapping(s) to another component.
`subscribe(filter)`	Changes a mapping's content-based filter(s).
`privilege(ac_policy)`	Alters an endpoint(s)' access policy.

Table 1 SBUS reconfiguration functions

(by parties external to the communication) on a mapping to select the messages transmitted. In this way, the filter acts as an authorisation rule evaluated in the context of each message, based on its content.

Protecting discovery: There will be instances where even the existence of a component may be sensitive, e.g. services relating to sexual health. A component can avoid being discovered, by electing (or being told) not to register with an RDC. However, this may preclude important interactions. An RDC maintains access control policy to dictate the components that may register and query, which is useful where an RDC is responsible for a particular grouping. However, more flexible, granular controls are also required. As such, the local RDC mirrors (where appropriate) the ACLs of its registered components. These are used to filter the results of a discovery query, so that only details of accessible components are returned by a query.

Discovery by inspection can reveal sensitive information. For this, SBUS provides two forms of control. First, a component maintains access control policy restricting the components that may inspect it. Secondly, a component can dictate whether its existence is revealed to others in an inspection operation.

Though *security by obscurity*, these measures help prevent inadvertent discovery of services, providing an extra hurdle for the malicious. Of course, the access control regime still protects the data and component metadata even if an address is known.

6.4 Reconfiguration (requirement 4, §3.4)

Runtime reconfiguration is crucial to supporting emerging health services, and was the fundamental concern in designing SBUS. Table 1 presents the SBUS reconfiguration API, which a component uses to change its state. SBUS ensures that all related operations are performed, e.g. that removing a privilege closes connections that are no longer authorised, and that the RDC is informed of the privilege change.

Third-party initiated reconfiguration

A key contribution of SBUS is that it enables third-party initiated, or remote reconfiguration, where a component effectively 'invokes' the SBUS operations of another. This makes it possible to instruct components on how and when to behave; e.g. to map or unmap, update privileges, apply filters, etc.

Such functionality is implemented through *control* messages. Each component has a set of default control endpoints that directly correspond to the reconfiguration API (Table 1). If a component receives a control message, it will perform the relevant operation according to the control message's parameters; this is equivalent to self-invocation of the operation. The security mechanisms ensure that control messages are only acted on when issued by appropriately trusted peers.

Fig. 5 illustrates a component instructing another to undertake a mapping (step 1). Here the control message forces an RDC query (step 2), though passing the network address avoids this step. The component, as instructed, then establishes the mapping (step 3).

Figure 5 Third-party initiated mapping

It is this capability that enables powerful and flexible policy enforcement. Further, as any component can influence another, it allows decentralised control. Such an approach is crucial to realising the vision of pervasive healthcare (for example, it underpins the functionality of Fig. 2). This is because it enables application-independent support, where data and components may be used for a number of purposes; at a high-level to meet user and application-level goals, as well as at the system level, for service composition, connection management, etc.

7 Policy Enforcement

Policy encodes particular goals. Its role is to regulate, coordinate, control and manage a system, by performing actions in the particular circumstances. It is SBUS' third-party reconfiguration capability that paves the way for application-independent and cross-application policy enforcement. That is, policy actions achieve goals through executing SBUS reconfiguration operations.

In §4 we described the general categories of policy actions relevant to care, which in SBUS include: *reconfiguration*, involving the management of components, e.g. altering mappings, privileges, and filters; *messaging*—generating and transmitting new events and alerts; and

policy management selecting (or de/activating) the applicable ruleset.

SBUS provides the mechanism for any component to (potentially) affect another. This means, for example, a distributed application can control its components as it deems appropriate. However, as discussed in §4, to assist in managing complex environments and interactions, and to deal with cross-application concerns, it is useful to have components dedicated to policy specifics.

To explore policy enforcement, we implemented two policy engines that aim at different environments. One approach was to integrate policy engine functionality into a relational database, the other more lightweight, designed to run on a mobile device. The reasoning was to demonstrate a powerful, expressive policy engine, suitable for managing the components belonging to a particular systems environment such as a hospital, surgery or patient's home; while the mobile policy engine encapsulates an individual's preferences, to govern the interactions between their mobile device and components in their surroundings.

Both approaches defined policy in ECA rules, where each rule embodies a specific action, to be applied in response to a particular happening. Thus, high-level policy is encoded, and the response automatically effected, through a set of rules.

7.1 Database policy engine

Databases are highly amenable to policy enforcement capabilities. Firstly, databases operate across applications, enabling indirect and asynchronous communication between components. More importantly, systems require persistence. Coupling policy enforcing capabilities with databases allows policy to leverage the rich representation of state and context encapsulated within the datastore.

We built policy engine functionality into a database by first making the database an SBUS component (see [47]). Relations were defined between endpoints and table schemata, so that a database tuple converts (in a type-safe manner) to an SBUS message and vice-versa to enable the sending and receiving of messages. Policy rules were then implemented as database *triggers* (ECA rules), where the enforcement conditions referenced tables—perhaps corresponding to an endpoint (incoming message), or a more complex state representation. The database rules engine would then enforce these on the relevant happening, taking the defined action which could include creating and transmitting a message, altering the triggers to change the active policy set, updating state representations by updating/persisting data, or effecting a reconfiguration through sending SBUS control messages.

Initially, the database policy engine was developed for managing particular environments, for instance, to control interactions between systems (and people) perhaps within a hospital, particular ward, or patient home. This is because a database requires a server, and thus is part of a relatively fixed infrastructure. However, with the increase in the availability and prevalence of cloud services, such an engine could also facilitate policy enforcement at a number of locations.

7.2 Mobility: Lightweight engines

Some components are mobile, moving between different operating environments and infrastructures; e.g. from leaving a car to entering a hospital. Policy engines can enforce user preferences to manage and govern the interactions with the particular environment, e.g. automatically connecting a nurse to a patient's vital-signs sensor when entering their home.

We implemented a lightweight policy engine on Android to manage the interactions between components on the device in its physical environment. Developed as a service component, it maintains a simple rules engine, where rules can be defined to execute on particular events. Though the engine could respond to any event, our focus was changes in location. On moving to a new environment, as determined by OS-events (e.g. connecting to a new network) or higher-level event definitions (e.g. sensor-data indicating a location change), the service automatically applies rules to manage and dis/connect components (typically those on the device with those in physical proximity) as appropriate.

7.3 Policy considerations

Since policy actions are triggered by changes in state, a policy engine needs the ability to detect, or be told of, state changes relevant to the policies it is responsible for enforcing. Our approach leverages the existing messaging capability to also provide both the representation of context (as events) and policy actions (as messages and reconfiguration instructions through control messages). Such an approach precludes the need to implement any specific policy layers, or particular constraints over the infrastructure and its design. Policy engines are components like any other and render the middleware policy and context aware.

The event processing capabilities of the middleware directly impact the flexibility and expressiveness of the policy rules. That is, a system only capable of dealing with primitive events is far more limited than one employing *composite events* [51] or other context modelling techniques [40] that can correlate and compose events within and across message streams. Such capabilities may be built into policy engines, and/or implemented in other components (such as sensor gateways)

that report the relevant state transitions to policy engines.

Without event semantics, higher-level components would need to understand and interpret the data. Reasoning over events and event flows facilitates data transformation and informed choices, e.g. in service discovery. As §5.4 mentions, while the role of semantics is clear, it remains an ongoing challenge in healthcare. Interoperability is difficult, even between the two prominent clinical coding schemes [52]: HL7,[13] and SNOMED.[14] There are groups dedicated to such issues, including OpenEHR[15] (see also [53]), and Integrating the Healthcare Enterprise,[16] though these focus on the clinical perspective. Issues of semantics compound in the pervasive health environment, given the enormous range of potential components, interactions, and usage scenarios.

Policy conflict is a concern, particularly in an environment with dynamically changing users and services, and with numerous policy engines. We have previously considered issues of conflict detection and resolution for trigger-based policy [54]. Thus far we have encountered few conflicts primarily because in practice, there will be a number of policy engines (operating within different scopes)—likely linked to component/infrastructure ownership—each having specific concerns. For instance, a policy engine might manage the components in an hospital ward; policies relating to that ward will be centrally defined. Similarly, a user will have their own policies that dictate how the local components (on their phone) interact with an environment; e.g. a doctor may wish to interact with components in a ward, but this only with those components that the environment (here, the ward's policy engine) authorises.

Note that our focus thus far has been generic, providing the mechanism for policy enforcement that is capable of effecting a wide range of goals. Further work is required to explore effective mechanisms (interfaces) for users to author policy. In practice, policy authoring and rule derivation will necessarily be determined by the applications being targeted, depending on likely users, components, etc. For more on policy specification issues, see [55].

There is generally a tradeoff between flexibility and efficiency. Thus, one intuitively expects policy enforcement to entail performance overheads. This is because such an approach involves policy engines, and the associated monitoring of state changes, evaluating policies and taking relevant actions. Firstly, it is important to note that healthcare presents an environment that operates at human speed. Therefore, in many situations, any degree of automatic response will be more than adequate, especially given the status-quo in which interventions tend to be manual, with little automated assistance, e.g. using a pager or telephone.

We explored some overheads with respect to policy enforcement—see [47] for details of the experiments—which indicate the practicality of policy-based coordination, and third-party initiated reconfiguration. We found for a particular scenario involving the database policy engine, that the overheads of policy enforcement were lost in the network variability when clients components connected through a particular wireless network; with a small statistically significant overhead (∼5ms out of a ∼35ms operation) for the same scenario where components connected through a particular Ethernet network. This is important, given wireless networks will be a predominant medium for communication in pervasive care environments. Our Android implementation, for a particular scenario, was significantly slower at rule evaluation for a particular scenario (∼75ms to enforce a policy, cf. ∼12.5ms for the database policy engine).

We mention these results as they indicate that performance will depend on the underlying infrastructure, and that policy may not necessarily introduce a tangible overhead. Ultimately, any timing information will vary according to the implementation and the environment: depending on factors such as the physical infrastructure, OS (e.g. Android), network load, cross-traffic, database size, message sizes/frequency, number of rules, users and components, etc.

Such factors must be considered by those designing for emerging care environments, if their system has specific performance requirements. For example, systems that require a particular service level, e.g. a pacemaker system, may be better developed as a bespoke, closed system to ensure the performance necessary for safety. Interfaces (or gateways) can still provide for integration of the specialist system with the general systems environment, for non time-critical operations (e.g. parameter tweaking). However, many health interventions occur minutes or even hours after incident detection; certainly, an automated response is a movement in the right direction. Indeed, our particular experiments resulted in sub second functionality (see [47]), which opens up real healthcare possibilities.

The focus of our policy work is to bring together systems in a more general manner, focusing on flexibility and adaptability to bring functionality and efficiencies previously not available.

[13] `www.hl7.org` (21 Mar 2014)

[14] `www.ihtsdo.org/snomed-ct` (21 Mar 2014)

[15] `www.openehr.org` (21 Mar 2014)

[16] `www.ihe.net/` (21 Mar 2014)

8 Summary and concluding remarks

Healthcare is undergoing an evolution, driven and supported by the health profession, government initiatives and technological innovation. The healthcare systems environment will thus involve an increasing number of applications, systems and services. Users, be they clinical, informal carers or patients, will leverage these in various ways, to provide a number of different health outcomes relevant to their preferences and circumstances.

Information, communication and management capabilities underpin the vision of pervasive, preventative and personalised healthcare. In this paper we have presented and discussed the requirements for middleware to support this. We see the open challenges as follows:

1 The need to incorporate existing and future sensor technologies into a generic system architecture.
2 The need to move towards an open systems approach by developing applications without embedded policy and predefined interactions.
3 The need to support policy authoring on behalf of many categories of individuals.
4 The need to develop an event-based middleware architecture to achieve:
   - Response to changes in people's and environmental contexts.
   - Support for a wide range of communication patterns.
   - Support for policy engines as middleware service components.
   - Support for policy-driven, dynamic reconfiguration of components by third parties, when authorised.
   - Support for security and privacy.

Our work on the SBUS middleware infrastructure and policy engines demonstrates the feasibility of this open systems approach. This moves beyond providing 'systems-level-glue' to actively driving new functional possibilities through high-level, user-specified preferences, thus making the vision of pervasive, preventative and personalised healthcare a real possibility.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

This article represents our work and experience over many years in infrastructure for supporting health services. Both authors contributed with respect to the concepts, technical work and text provided in this paper, and both have read and approved the final manuscript.

Acknowledgements

We acknowledge the support of the UK Technology Strategy Board and the Engineering and Physical Sciences Research Council for the PAL project, grant TP/AN072C, 2009-12.

References

1. World Health Organisation, US National Institute of Aging: Global health and ageing (2011)
2. Department of Health (UK): Improving Chronic Disease Management (2004)
3. World Health Organisation: Innovative Care for Chronic Conditions (2002)
4. Department of Health (UK): NHS 2010–2015: from good to great. Preventative, people-centred, productive. (2009)
5. W.R., H.: Medical informatics: Improving health care through information. Journal of the American Medical Association **288**(16), 1955–1958 (2002)
6. Department of Health: Supporting people with long term conditions to self care (2006)
7. European Science Foundation: Personalised Medicine for the European citizen (2012)
8. Franko, O.I., Tirrell, T.F.: Smartphone app use among medical providers in ACGME training programs. Journal of Medical Systems **36**(5), 3135–3139 (2012)
9. Baum, P., Abadie, F.: Strategic intelligence monitor on personal health systems phase 2, market developments - remote patient monitoring and treatment, telecare, fitness/wellness and mhealth. JRC-IPTS Working Papers JRC71141, Institute for Prospective and Technological Studies, Joint Research Centre, European Commission (2012)
10. Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. IEEE Journal of Biomedical and Health Informatics **17**(3), 579–590 (2013)
11. Mihailidis, A., Bardram, J.E.: Pervasive Computing in Healthcare. CRC Press, FL, USA (2010)
12. Bernstein, P.A.: Middleware: a model for distributed system services. Communications of the ACM **39**(2), 86–98 (1996)
13. Payne, K.F.B., Wharrad, H., Watts, K.: Smartphone and medical related app use among medical students and junior doctors in the United Kingdom (UK): a regional survey. BMC Medical Informatics and Decision Making **12**, 121 (2012)
14. Next Generation European Ambient Assisted Living Innovation Alliance: AALIANCE2 Roadmap (2013)
15. Groves, P., Kayyali, B., Knott, D., Van Kuiken, S.: The big data revolution in healthcare: accelerating value and innovation. New York (NY): McKinsey Global Institute (2013)
16. Mattern, F., Floerkemeier, C.: From the internet of computers to the internet of things. In: From Active Data Management to Event-Based Systems and More, pp. 242–259. Springer, Berlin, Heidelberg (2010)
17. Blair, G.S., Bennaceur, A., Georgantas, N., Grace, P., Issarny, V., Nundloll, V., Paolucci, M.: The Role of Ontologies in Emergent Middleware: Supporting Interoperability in Complex Distributed Systems. In: ACM/IFIP/USENIX Middleware 2011, Springer LNCS 7049, pp. 410–430 (2011)
18. Bacon, J., Eyers, D., Pasquier, T., Singh, J., Papagiannis, I., Pietzuch, P.: Information flow control for secure cloud computing. Transactions on Network and Service Management, Special Issue on Cloud Service Management **PP**(99), 1–14 (2014)
19. Chakravarthy, S.: Early active database efforts: A capsule summary. IEEE Transactions on Knowledge and Data Engineering **7**(6), 1008–1010 (1995)
20. Issarny, V., Blair, G.: Guest editorial: Special issue on the Future of Middleware (FOME'11). Journal of Internet Services and Applications **3**(1), 1–4 (2012)
21. Sadjadi, S.M., McKinley, P.K.: A survey of adaptive middleware. Michigan State University Report MSU-CSE-03-35 (2003)
22. Kon, F., Costa, F., Blair, G., Campbell, R.H.: The case for reflective middleware. Communications of the ACM **45**(6), 33–38 (2002)
23. Ibrahim, N., Le Mouël, F.: A survey on service composition middleware in pervasive environments. International Journal of Computer Science Issues **1**, 1–12 (2009)
24. Kalasapur, S., Kumar, M., Shirazi, B.: Dynamic service composition in pervasive computing. IEEE Transactions on Parallel and Distributed Systems **18**(7), 907–918 (2007)
25. Ben Hamida, A., Kon, F., Ansaldi Oliva, G., Dos Santos, C.E.M., Lorré, J.-P., Autili, M., De Angelis, G., Zarras, A., Georgantas, N., Issarny, V., Bertolino, A.: The future internet, pp. 81–92. Springer,

Berlin, Heidelberg (2012). Chap. An Integrated Development and Runtime Environment for the Future Internet
26. Little, M., Shrivastava, S.: Another look at the middleware for dependable distributed computing. Journal of Internet Services and Applications **3**(1), 95–105 (2012)
27. White, J., Dougherty, B., Schantz, R., Schmidt, D.C., Porter, A., Corsaro, A.: R&D challenges and solutions for highly complex distributed systems: a middleware perspective. Journal of Internet Services and Applications **3**(1), 5–13 (2012)
28. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **38**(4), 393–422 (2002)
29. Chen, M., Gonzalez, S., Vasilakos, A., Cao, H., Leung, V.C.: Body Area Networks: A Survey. Mobile Networks and Applications **16**(2), 171–193 (2011)
30. Wang, M., Cao, J., Li, J., Das, S.K.: Middleware for wireless sensor networks: A survey. Journal of Computing Science and Technology **23**(3), 305–326 (2008)
31. Rajagopalan, R., Varshney, P.K.: Data aggregation techniques in sensor networks: A survey. IEEE Communications Surveys & Tutorials **8**, 48–63 (2006)
32. Leontiadis, I., Efstratiou, C., Mascolo, C., Crowcroft, J.: Senshare: Transforming sensor networks into multi-application sensing infrastructures. In: European Conference on Wireless Sensor Networks, pp. 65–81 (2012)
33. Marinovic, S., Craven, R., Ma, J., Dulay, N.: Rumpole: A Flexible Break-glass Access Control Model. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp. 73–82 (2011)
34. Sloman, M.: Policy driven management for distributed systems. Kluwer, Journal of Network and Systems Management **2**, 333–360 (1994)
35. Twidle, K., Lupu, E., Dulay, N., Sloman, M.: Ponder2 - A policy environment for autonomous pervasive systems. In: IEEE Symposium on Policy for Distributed Systems and Networks (Policy'08), pp. 245–246 (2008)
36. Minsky, N.H., Ungureanu, V.: Law-governed interaction. ACM Transactions on Software Engineering Methodologies **9**(3), 273–305 (2000)
37. Matthys, N., Huygens, C., Hughes, D., Ueyama, J., Michiels, S., Joosen, W.: Policy-driven tailoring of sensor networks. In: Springer, Sensor Systems and Software, S-CUBE'10, pp. 20–35 (2010)
38. Singh, J., Eyers, D.M., Bacon, J.: Disclosure control in multi-domain publish/subscribe systems. In: ACM 5th International Conference on Distributed Event-Based Systems, DEBS'11, pp. 159–170 (2011)
39. Wun, A., Jacobsen, H.-A.: A Policy Management Framework for Content-Based Publish/Subscribe. In: ACM/IFIP/USENIX Middleware 2007, Springer LNCS 4834, pp. 368–388 (2007)
40. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing **6**(2), 161–180 (2010)
41. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing **2**(4), 263–277 (2007)
42. Bennaceur, A., Blair, G., Chauvel, F., Gang, H., Georgantas, N., Grace, P., Howar, F., Inverardi, P., Issarny, V., Paolucci, M., Pathak, A., Spalazzese, R., Steffen, B., Souville, B.: Towards an architecture for runtime interoperability. In: Leveraging Applications of Formal Methods, Verification, and Validation. Lecture Notes in Computer Science, LNCS 6416, pp. 206–220. Springer, Berlin, Heidelberg (2010)
43. Paolucci, M., Souville, B.: Data interoperability in the future of middleware. Journal of Internet Services and Applications **3**(1), 127–131 (2012)
44. Campbell, J.R., Carpenter, P., Sneiderman, C., Cohn, S., Chute, C.G., Warren, J.: Phase II evaluation of clinical coding schemes: Completeness, taxonomy, mapping, definitions, and clarity. Journal of the American Medical Informatics Association **4**(3), 238–251 (1997)
45. Mühl, G., Fiege, L., Pietzuch, P.: Distributed Event-Based Systems. Springer, New York (2006)
46. Singh, J., Vargas, L., Bacon, J., Moody, K.: Policy-Based Information Sharing in Publish/Subscribe Middleware. In: IEEE 9th Symposium on Policy for Distributed Systems and Networks, Policy'08, pp. 137–144. IEEE Computer Society, Palisades, NY, USA (2008)
47. Singh, J., Bacon, J.: SBUS: A generic, policy-enforcing middleware for open pervasive systems. University of Cambridge Computer Laboratory Technical Report (TR 850) (2014)
48. Ingram, D.: Reconfigurable Middleware for High Availability Sensor Systems. In: ACM 3rd International Conference on Distributed Event-Based Systems (DEBS'09) (2009)
49. Haslhofer, B., Klas, W.: A survey of techniques for achieving metadata interoperability. ACM Computing Surveys **42**(2), 1–37 (2010)
50. Dierks, T., Allen, C.: The TLS Protocol (RFC 2246). Internet Engineering Task Force, (1999). Internet Engineering Task Force
51. Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S.-K.: Composite events for active databases: Semantics, contexts and detection. In: Very Large Data Bases, VLDB'94, pp. 606–617 (1994)
52. Benson, T.: Why interoperability is hard. In: Principles of Health Interoperability HL7 and SNOMED. Health Information Technology Standards, pp. 21–32. Springer, London (2012)
53. Garde, S., Knaup, P., Hovenga, E.J., Heard, S.: Towards semantic interoperability for electronic health records–domain knowledge governance for open ehr archetypes. Methods of information in medicine **46**(3), 332–343 (2007)
54. Singh, J.: Controlling the dissemination and disclosure of healthcare events. PhD thesis, University of Cambridge, and Computer Laboratory Technical Report TR 770 (2009)
55. Reeder, R.W., Karat, C.-M., Karat, J., Brodie, C.: Usability challenges in security and privacy policy-authoring interfaces. In: Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part II. INTERACT'07, pp. 141–155 (2007)