Towards a domain-driven, model-based approach to developing clinical information systems: innovating for the future in NHS Wales

Dr. Mark Wardle MD FRCP

Consultant Neurologist
University Hospital Wales
Heath Park, Cardiff. CF14 4XW

3rd February 2017

Contents

1 Introduction

I have written this document to explain my thoughts on the most appropriate way to design and clinical information systems that valuably support the process of healthcare. As such, it includes a methodology for creating software for use by clinicians, managers and administrators to support both the day-to-day care of patients and the management of services for groups of patients.

I divide the "process of healthcare" into three core themes which are related and inter-dependent:

1. The 'day-to-day' care of a single patient; data relates to a single individual.

2. The management, audit and governance of clinical services; data relates to grouped, aggregated patient data and is service-specific.

3. Supporting clinical research; data relating to a patient's suitability for research projects as well as enrolment, consent and research-level data.

All three themes are important and traditionally, healthcare information technology has focused only on the administrative side of the management of clinical services. As a result, we commonly know how many patients are seen in clinic, but have very little information as to why they are being seen or how they fare after treatment. Important questions such as "how many patients with Parkinson's disease do we care for?", "what are the outcomes for patients after having a mastectomy?" or "why are we seeing these patients in outpatient clinic?" cannot be answered.

I have developed a working electronic patient record system that has been in daily and continuous use since 2009 within NHS Wales that uses many of these architectural design ideas. In particular, it is designed around a robust information model that abstracts the reality of the generic and specialised processes of healthcare together with prospective longitudinal outcome data.

My general conclusions are:

- The development of healthcare information technology must be underpinned by a robust open platform built on data standards, clinical modelling and interoperability.

- Such developments need teams made up of clinicians and information technology staff, with all involved receiving protected time as well as training, education and career development.

- There is and will continue to be a myriad of disparate heterogeneous systems and all developments must consider interoperability of both data and process as a pre-requisite to continued funding.

- Aggregated service-level data must be made readily available to appropriate clinical staff and those responsible for the management of clinical services. All systems should consider the benefit of making aggregated data available in real-time and therefore how the underlying model and implementation can support such uses.

My specific conclusions for NHS Wales are:

- The soundbite "Once for Wales" is currently being interpreted in a "system" or "product" based context forcing an approach in which success is defined by the rollout of specific applications. "Once for Wales" needs to be re-defined and re-imagined to focus on functionality, core national infrastructure and a service-based approach underpinned by a focus on data standards, interoperability and process mapping.

- That a focus on a single application "Welsh Clinical Portal" will not meet the needs of clinical users or patients, slows and stifles innovation and is not fit-for-purpose; indeed there are already and will continue to be a multiplicity of user-facing applications which must be considered when considering health technology in NHS Wales.

- That current strategy pits the NHS Wales Informatics Service against individual health boards in relation to both strategy and the use of limited resource, instead of fostering collaboration and innovation.

- That current developments focus on user-interface design rather than defining the problem, modelling the data and the processes relating to those data and understanding information flow.

- That current technology and research grant funding frequently creates single products or solutions with limited scope and little chance of more widespread adoption due to lack of cohesive management from Government and a lack of a formal interoperability framework in which new technologies can be piloted and subsequently rolled-out. Instead, many projects result in the creation of more data silos in which data and processes relating to those data are not available across the health and social care services outside of the original scope of the project. As such, limited value is obtained from current grant funding.

- That a new framework for success and progress be developed that focuses upon high-level clinical and information requirements rather than the deployment of a specific application. Such a framework must focus on those targets that have the most benefit for patients. As such, we must deem it a success when a clinical user can access the patient record in an emergency care setting irrespective of the software used instead of only thinking we are successful when a particular organisation rolls out a particular product.

- That to realise the benefits of information technology in healthcare for patients in Wales, a new multiple-tier open platform strategy be developed. Such a platform would consist of a core national foundation of robust, highly regulated, highly reliable services supporting tiers of additional high-level frameworks acting as vendor-neutral middleware layers providing functionality across organisations and clinical pathways. A new top-level innovation tier should be developed which allows small, low-risk, highly innovative applications to be developed that support specific clinical processes or workflow.

- That the current organisational structure of the NHS Wales Informatics Service (NWIS) in how it relates to Welsh Government, individual health boards and its own staff is not fit-for-purpose. To whom does NWIS and its staff work and are the current governance arrangements sufficient to ensure success in the future? Are staff integrated in teams to support the clinical and technical architecture outlined within this document?

- NHS Wales should consider a clinical informatics fellowship scheme and informatics academy in which healthcare professionals and information technology professionals with interests in informatics, performance, continuous improvement and software can engage and develop.

To provide evidence for these statements, I will begin in Section 2 with a discussion on the medical record and its structure and the workflow that is built upon this foundation. In creating an electronic health record, I believe a focus on developing clinical and process information models to be essential and I discuss this in Section 3. I will then cover how data standards and terminology are used within such an information model in Section 4.

The design of clinical information systems relies not only on informatics expertise, but on understanding of the constraints and limitations of any technical solution. It is therefore critical that those who commission, scope or design clinical systems have an understanding of sound software engineering practices. As such, in Section 5 I discuss the advantages and disadvantages of different technical architectures. I strongly believe that the technical architecture is determined logically from an understanding of the medical record and clinical modelling as outlined in Section 2 and Section 3. Similarly, the architecture determines the required organisational structures that build, support and maintain those core pieces of software. As such, I extend the discussion from technical architecture to organisational structures in Section 6.

Finally, I will work through several clinical examples to demonstrate the methodology in action in Section 8. Firstly, I consider systems designed for the management of emergency units in Section 8.1. After this, I explain how I would approach the structured recording of diagnostic and problem lists in Section 8.2. I cover the possible future development of a national growth chart for children in Section 8.3 and my final example covers prescriptions in relation to electronic discharge documentation in Section 8.4.

2 The medical record

> "You're a victim of it or a triumph because of it. The human mind simply cannot carry all the information without error so the record becomes part of your practice."
>
> Dr Larry Weed, Internal Medicine Grand Rounds, 1971

The medical record encompasses and complex, disparate and evolving collection of information relating to the day-to-day care of an individual patient. Aggregated, such data provide valuable information about cohorts of patients managed by a clinical service supporting clinical governance and the management of the that service.

2.1 Paper medical records

Paper medical records are expensive, poorly accessible and do not support team working effectively. They are frequently poorly structured and while they contain large amounts of data, it is difficult to use those data to support clinical care. Paper records must be tracked and within a large organisation and such records need to be made available to clinical and administrative staff in order to appropriately manage the patient. This is particularly difficult when a single patient may be under the care of multiple services which is particularly common for patients with chronic long-term health conditions.

2.2 Clinical governance

> "If you can't audit a thing for quality, it means you do not have the means to produce quality".
>
> Dr Larry Weed, Internal Medicine Grand Rounds, 1971

Clinical governance is a framework in which we are accountable for continuously improving the quality of our services safeguarding high standards of care by creating an environment in which excellence in clinical care can flourish.

An important part of clinical governance is in the use of data to understand both process and outcomes. A typical clinical department will know how many patients are waiting for an outpatient appointment but will not be able to understand why they are waiting or understand a patient's eventual outcome because usually only administrative data are collected systematically.

Many clinical teams will undertake service improvement projects or clinical audits, in which a clinician identifies a problem, undertakes to measure that problem by clinical audit, analyses the results and makes an intervention in order to solve that problem. For example, a typical clinical audit might be in the assessment of risk of deep vein thrombosis in inpatients in which a doctor will systematically analyse the paper notes of a group of inpatients, record and present the results and enact a change to improve those results with a plan to re-audit after that intervention. Such a methodology has been formalised by the Six Sigma quality initiative[1] in which five steps form a virtuous cycle of continuous improvement (Figure 1).

Such continuous improvement is possible with paper records, but it cannot be performed systematically and routinely, requires a large amount of manual intervention and the cycle of improvement is slow; the analysis of results and a plan of action to effect improvement may take months with no guarantees that any systematic assessment of those interventions will be performed.

It therefore follows that the routine and systematic collection of process and clinical outcome data offers the potential to radically improve the efficiency of the continuous improvement lifecycle as long as sufficient consideration is given to the use of such data, the context of those data and an understanding of the processes in which those data are involved.

2.3 Electronic medical records

Electronic medical records and workflow should model and abstract real-life processes but digitisation of paper processes should not be a slavish exact copy of existing processes, particularly when technology offers opportunities to improve the process of healthcare.

[1] See `https://www.isixsigma.com/dictionary/dmaic/`

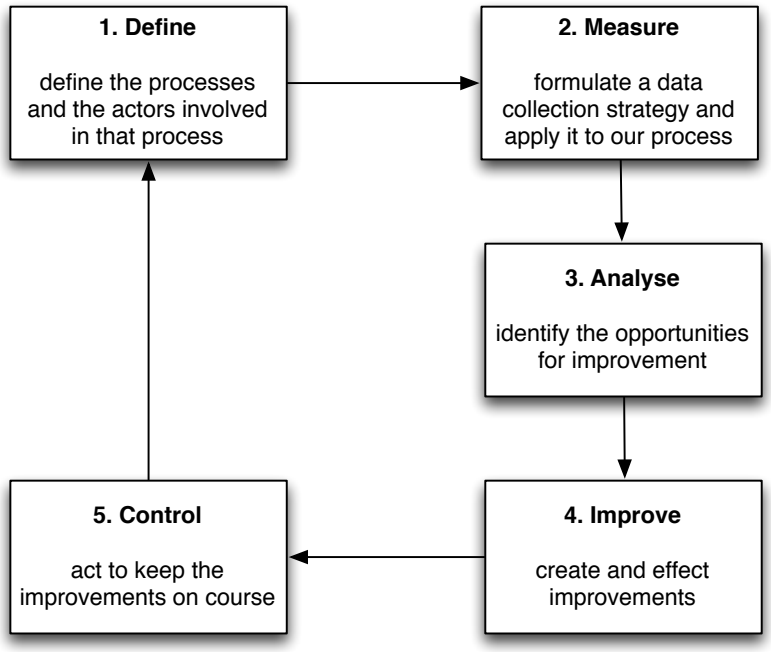

Figure 1: Six Sigma methodology of continuous quality improvement.

Indeed, we are facing an explosion of clinical data created by health professionals, administrators and patients themselves and clinical information systems must consider how they will integrate and make such data useful in clinical practice. For example, Figure 2 shows how all of these sources of data are mutually dependent on one another. For example, what is the point in recording patient reported outcome measures if that data cannot be safely linked to a particular intervention or procedure and the co-morbidities and baseline assessments for that patient?

The logical conclusion is that we will always have a multiplicity of systems and that in order to create robust clinical systems that outlive our current level of technological advancements, we have a responsibility to insist on open well-documented data standards, to integrate a well-developed clinical terminology, and foster collaboration, iterative development and embrace interoperability.

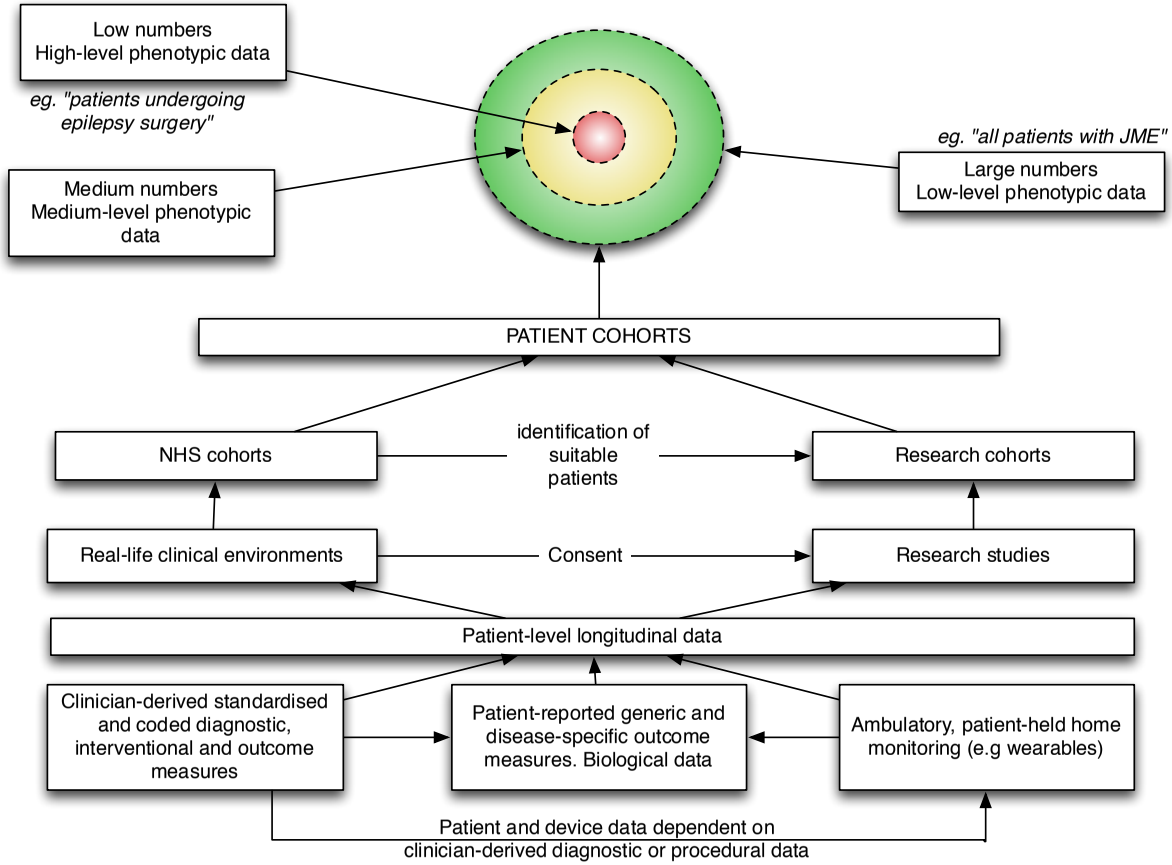

Figure 2: Data flows within the 'medical record'

3 Clinical Modelling

A model is an abstraction of the real-world distilling knowledge into core concepts and processes. A model is a careful balance between complexity and over-simplification with models that are too complex providing brittle abstractions which are poorly generalisable and models that are too simple not supporting all of the demands placed upon it in real-life clinical scenarios.

Modelling is an art but the approach can be made more systematic if modellers work iteratively within a supporting organisational environment clinical and technical staff work together in logical teams. Importantly, the scope of their model should be limited; large enough to solve the domain problems but small enough to result in a practical implementation. As such, it is important to have an appropriate separation of concerns in which teams can work independently of other teams as much as possible and that interfaces between that team and others are formalised on a contractual basis.

3.1 High-level functional requirements

Clinical modelling is dependent on knowledge of the domain and the functional requirements to which the model will serve. It is useful to abstract these requirements to make them as generally-applicable as possible and not to generate needlessly specific requirements with limited applicability. However, the process of generated functional requirements and a subsequent data and process modelling is not a one-way process but must be an iterative two-way development in which there is feedback and discussion among all stakeholders and between all steps of requirements-gathering, modelling and implementation.

3.2 The process of developing a clinical model

I suggest dividing clinical modelling into three related steps:

1. Modelling data within a medical record
2. Modelling processes and workflow relating to a medical record

3. Implementation of a clinical model into a workable and practical solution.

These three steps should be considered separately but are dependent on one another and inherently depend on the functional requirements. In designing clinical information systems, a data model may change infrequently or a design may even be correct in perpetuity; a systolic blood pressure recorded using a cuff on the right arm of an individual will always be a systolic blood pressure recorded using a cuff on the right arm of an individual. However, a model of the processes of care will likely change more frequently as a result of changes in working practices and service design, and an implementation may change even more frequently as a result of new technologies and devices.

As each step is dependent on each other, modelling must be an iterative work with feedback loops between all steps rather than performed in sequence by different members of the team. An implementation cannot be designed without consideration of the data structures and processes that leverage the data within those structures and a model cannot be designed unless it is practical and efficient to perform an implementation. It also follows that clinical modelling must be performed by a team of individuals with a range of skills including those with important domain knowledge such as health professionals as well as those responsible for the design and implementation of the resulting software such as technical architects and software developers.

Traditional waterfall-based design methods emphasise a sequential process involving business analysis and scoping, design, implementation, verification and then subsequent maintenance. Indeed, such methods are commonly used within NHS Wales with a focus on producing a product to solve a particular problem which then becomes ‘complete’ and enters ‘maintenance’. I strongly believe that this approach is wrong and instead development should by underpinned by a robust model developed iteratively considering both workflow and implementation together with an understanding that any solutions will never be complete.

3.3 Modelling the medical record

The breadth of knowledge within the medical record is potentially daunting with multiple types of data recorded in many different contexts and for many different

purposes. Examples of data recorded for the day-to-day clinical care of patients is shown in Table 1. For these examples, such data will need to be shown to users in order for them to make sense of a patient's condition or clinical course, but also used to monitor a clinical service by providing real-time data on patient flow and progression along one of many clinical pathways.

I advocate a clinical record in which much information is stored in a highly structured format. Some free-text is necessary but where possible, clinical information should be recorded using a standardised terminology such as SNOMED CT held within larger data structures representing the clinical model.

Advances in technology such as new interventions, evolving diagnoses, new diagnostic tests and changing processes result in an ever-expanding knowledge base to which software systems must evolve and adapt. As such, no software system designed to model the medical record will ever be complete and as such, designing for incremental change in the medium and long-term is a pre-requisite. If real-world knowledge is ever-changing, then models too must constantly be refined and evolved. As such, the design and subsequent constant refinement and evolution of any model underpinning a complex clinical information system must be supported by an appropriate organisational structure which facilitates an iterative development approach. Such iterative development requires domain experts such as clinicians work closely in teams with developers to constantly refine their own understanding, focus the analysis of requirements and distil what they know to the core essentials that must underpin the design of any subsequent information system.

3.3.1 Electronic medical records and the importance of context and provenance.

An advantage of electronic records is that such records can be made universally available wherever they are needed such as when a patient presents to an emergency health facility or a new outpatient clinic or when the patient contacts a team via email or telephone. A disadvantage of electronic records is the need for real-time curation of patient records so that users are not overwhelmed by excessive information that is not relevant in their specific clinical context. It is therefore vi-

Type of data	Description
Prospective clinical narrative	Free-text and structured longitudinal data recording the status of the patient, possibly historically, sometimes contemporaneously, often speculations regarding future progress together with management plans as a means of recording and communicating between healthcare professionals and administrative staff and potentially the patient.
Diagnostic	Structured coded information relating to the clinical problems of the patient, together with contextual information including provenance, dates and evidence.
Procedural and interventional	Structured coded information relating to the procedures and interventions experienced by the patient longitudinally potentially including indications for the procedure as well as a plan of management for a patient after the intervention.
Pathways and clinical services	Information relating to which pathways and clinical services the patient has been, will be or is currently registered together with their progress.
Administrative data	Appointments and scheduling for a variety of clinical encounters including diagnostics and interventions linked to pathways or clinical services.
Patient-reported outcome measures (PROMs)	Structured and unstructured reports from the patient regarding their progress using a range of generic and disease-specific health measures.
Patient-reported experience measures (PREMs)	Structured and unstructured reports from the patient regarding their experience of using health services.
Radiology	Reports of radiology investigations such as X-rays or scans.
Laboratory	Reports of laboratory investigations such as biochemistry or haematological laboratory tests.
Special	Reports of specialised investigations such as cardiac or neurophysiological tests.

Table 1: Examples of the types of data within the medical record

tal that any model supports not only the recording of information but the context and provenance in which those data were recorded and that views of aggregated information for a single patient take into account such contextual cues to ensure users see the information that is required and optionally filters information.

Such issues also arise in paper records, particularly when a single patient has multiple volumes of notes. However, as paper records serve a single organisation end-users may find important information related to their own service based on filing standards, paper colour, logos, page layout and the names of colleagues and can flick through the physical record more easily than a poorly curated and unstructured digital counterpart. However, paper records fail for patients looked after by multiple organisations, in which clinical correspondence from an outpatient clinic in one facility will usually not be available in another unless manual action by end-users is taken.

3.4 Modelling the processes involving medical records.

Medical records are more than a simply narrative of what has happened to a patient listed in sequential order. A number of important processes and workflows exist that relate to the sending and receiving of paper-based forms that must be considered in any abstraction of medical enterprise. Paper records such as printed correspondence, request forms, results of investigations are sent to a responsible individual in order for them to be read and acted upon. On some occasions, such documentation is annotated, forming a narrative of the actions undertaken by the end-user or forwarded to another, forming a chain of correspondence. Electronic systems are potentially safer offering guarantees of sending and receiving such requests, providing full audit trail functionality and non-repudiation of comments and action. However, any process relating to a medical record must be modelled with as much care as the modelling of data structures.

Most data standards relating to healthcare, such as HL7, openEHR, and FHIR, model the data relating to healthcare and not the processes and workflow that reflect the care of patients itself. While modelling data is an important pre-requisite in the conception and design of clinical information systems, modelling the processes and workflow that relate to that model is firstly an important test that data

modelling is fit-for-purpose and secondly, critical itself in ensuring a resulting clinical information system will be a workable and valuable asset in providing healthcare services.

Process and workflow will change frequently as clinical services evolve and develop and any model must take such changes into account so that any subsequent implementation can flexibly handle different workflows. Abstracting workflow is the most appropriate method to future-proof any clinical system and modellers must then consider how to model the configuration of more abstract workflows into a concrete implementation at runtime.

In most clinical services, data and processes currently revolve around the completion and processing of clinical documents which are subsequently filed as part of a longitudinal prospective and immutable medical record. As such, a transactional document model is usually an appropriate reasonable and accurate abstraction of current processes of care.

3.5 Modelling and implementation

Tying a clinical model to its implementation is critical. A model cannot be designed without consideration of its implementation as it is possible to develop models of data and processes that would be impractical, cumbersome or inefficient to implement by developers. As such, close working between domain experts and software engineers is necessary to iteratively explore and improve a model to ensure it meets the demands of the domain and of practical software engineering considerations.

3.5.1 Avoiding mismatches between a model and real-life.

A mismatch between the model and real-life processes may cause confusion for users while models that are based on the fundamentals of a particular domain result in clinical systems that are intuitive to use with readily predictable functionality. A model should therefore abstract real-life processes and structures when possible so that end-users can easily understand what is happening within the software they are using and should not be surprised by unintended consequences. However, modelling real-life processes should not be an exact replica of an exist-

ing paper-based process but a careful abstraction designed to capture the essence of the process and to be more widely generalisable.

While organisations, directorates, specialties, teams and even individual clinicians will have their own ways of working with disease and specialty specific data requirements, there is a danger in thinking that data modelling is simply getting a single group of domain experts in a room with a white-board. Instead, such work is simply the start of a process in which these requirements are analysed and distilled and subsequently made as generic and widely applicable as possible and yet not made so generic that a subsequently developed model no longer meets the needs of the original functional requirements.

4 Data standards and terminology

A terminology is a body of terms used within a particular subject of study or profession. A classification is a systematic organisation of things into classes. A classification system such as ICD-10 cannot be used as a terminology and many terminologies cannot be easily used as a classification system. This is an important distinction and relates to the how those data are primarily used.

4.1 SNOMED CT

SNOMED CT is a very large and comprehensive terminology. Importantly terms can be linked by multiple relationships to other terms. This makes it possible for software to determine that *multiple sclerosis* is a disease defined by *demyelination* of the *central nervous system*. When implemented properly, SNOMED CT enables software to make *intelligent* decisions about what to show, what data to request and what forms to present, based on the diagnoses entered. For example, the database would *know* that a patient had epilepsy if they were given a diagnosis of *juvenile myoclonic epilepsy* or *frontal lobe epilepsy* or any of the hundreds of other terms that are equivalent to a diagnosis of epilepsy. Thus a command to 'send an alert when a patient, belonging to a particular consultant, with motor neurone disease loses 5% of their body mass compared to their baseline at diagnosis', can be implemented easily. SNOMED CT allows the underlying logic to simply ask whether the patient has a type of *motor neurone disease* and this would automatically include all patients with related diagnoses such as *primary lateral sclerosis* and *pseudobulbar palsy*.

SNOMED CT is not confined to diagnostic and procedural information. There are hierarchies covering a wide range of medical terminology including anatomical structures, pathology, occupations and ethnic origins. With local extensions such as the NHS' DM&D (dictionary of medicines and devices) these codes can be used in any field that needs structured coded information.

Another advantage is support for synonyms. A distinct clinical concept can and usually has multiple synonyms - for example *Granulomatosis with polyangiitis* was previously known as *Wegener granulomatosis*. With synonym sup-

port, a user entering an outdated or synonymous term would find the synonym and see it mapped into the new modern preferred description of the term.

Within SNOMED CT, clinical terms are *concepts*, *synonyms* are *descriptions* and the relationships between concepts are recorded as *relationships*. While seemingly simple, as relationships themselves are defined by concepts (such as 'IS-A' as in "'*Motor neurone disease*—IS-A—*Disorder of nervous system*") it means that the relationship tree is infinitely extendable over time.

SNOMED CT is owned by the International Health Terminology Standards Development Organisation (IHTSDO) and is an international terminology, with the UK version managed by the UK Terminology Centre (UKTC) of the Health and Social Care Information Centre (HSCIC). There are online training resources provided by IHTSDO and the UKTC [2] as well as a simple online SNOMED CT browser[3].

I have developed an open-source terminology server[4] that provides fast free-text search and navigation around the SNOMED CT hierarchy as well as providing semantic understanding for any concept, allowing client software to answer questions like "does this patient have a type of granulomatous disease?", "was this patient born in Europe?" or even "is this drug a type of β-blocker?" . It would answer yes to the first question simply by understanding the diseases that the patient has been listed as having, answering 'yes' if a patient had sarcoidosis and 'no' if they had multiple sclerosis. Similarly, it would respond with 'yes' if a patient was recorded as being born in France but 'no' if they were born in Afghanistan. SNOMED CT provides the logical relationships in order to drive such computerised decision-making.

4.2 Information models

Each SNOMED CT concept, description and relationship has a unique and persistent identifier that can be stored in a data store. Most clinical applications persist information in a relational database and so a simple implementation may simply store the identifier as a foreign key to the row that represents that entity. However,

[2] `http://systems.hscic.gov.uk/data/uktc/training`

[3] `http://browser.ihtsdotools.org`

[4] `https://www.github.com/wardle/rsterminology`

whilst these terms have meaning when used in isolation (e.g. storing the identifier representing“myocardial infarction”), it is only when these terms are combined together in a logical way as part of a larger data model, that true meaning can be understood. It is analogous to definitions for individual words in a dictionary but true expression results only when these words are combined into sentences and paragraphs. As such, most concepts are only useful when considered within the information model in which it is recorded.

The information model used in which SNOMED CT concepts are stored is therefore critical to derive understanding from what can be inferred from the recording of a concept. This is particularly important for a terminology such as SNOMED CT in which terms may be recorded together to form a compositional (post-coordinated) term such as “Family history of...” (281666001) and “Obesity” (414916001) or out of convenience represented as a single SNOMED CT term “Family history: Obesity” (160311006).

If SNOMED CT had not only defined a terminology but also a wider information model then such compound pre-coordinated terms would be unnecessary and the recording of “obesity” within a model defining a family history would be sufficient. However, while compound pre-coordinated terms risk an explosion of terms to cater for multiple combinations, they do make it easier for end-users to find concepts that represent what they are trying to record and support workflows in which clinical terms are recorded in a relatively unstructured information model, such as that used in primary care historically using Read codes. Of course, SNOMED CT was developed as an amalgamation of SNOMED from the College of American Pathologists and Read (Clinical Terms Version 3) codes with the latter recorded prospectively and longitudinally in UK primary care systems in a relatively unstructured format. There are advantages in SNOMED CT being independent from the surrounding data model within which it is transmitted or stored particularly as SNOMED CT terms generally reflect core concepts relating to health, disease and the processes of care. As a result a range of models can be used with SNOMED CT and similarly, different terminologies can be bound to a model such as LOINC[5]

[5]LOINC is an alternative terminology focused on tests, measurements and observations — see `https://loinc.org`.

I generally recommend a highly structured approach to the storage of SNOMED CT codes in which the information model in which they are stored ensures no uncertainty in interpretation and that to simplify subsequent analysis and retrieval, compound pre-coordinated terms are decomposed into their components are stored appropriately. As such, users may enter information via a highly structured workflow in which the context is evident as part of the user interface such as recording family history or allergies or a less structured workflow in which terms can be entered and decomposed and entered into a more structured information model. Alternatively, an implementation may instead store the codes as entered and deal with ensuring that a diagnostic term recorded in a family history model is equivalent to the compound term.

To provide evidence to support a highly structured approach, I suggest considering how one might use SNOMED CT to record examination findings. While there is a code for "supranuclear gaze palsy" (420675003) there is no compound code to record that there is an absence of this examination finding. While one could request an addition to SNOMED CT, it is much more appropriate to consider examination findings in two categories, those found and those not found. The recording of a lack of clinical sign is important in clinical practice as well as for medico-legal reasons. As such, the information model in which this clinical finding is recorded is critical in providing understanding. SNOMED CT does allow post-coordinated terms in which multiple terms are combined together to give meaning. One possible way of representing the 'lack of' a clinical finding is to post-coordinate with a negation concept to form a compositional term but I advocate using a robust information model as a simpler method of expressing clinical knowledge particularly when one considers *how* users are expected to record such findings.

Example information models which can record SNOMED CT terms in context are HL7 and openEHR. OpenEHR uses the term 'archetype' as a synonym for 'information model'. The use of validated and published data structures support subsequent interoperability between disparate systems which can process that model. However, the use of a specific information model does not necessarily force the use of that model as a format in which to store data, but may be used only as a representation of data to be used for interoperability with other systems. Indeed, a

focus on a model of any form improves the potential for interoperability because that model is likely to be an abstraction of real-life concepts and thus it becomes possible to map from one information model to another.

However, even if two information models represent the same real-life concept and they look superficially similar, the process of mapping can introduce ambiguity and potentially even errors, particularly if a data element is present in one model but not in another. In addition, different terminologies may be used with an information model in a process called 'terminology binding' and so simply mandating a particular kind of information model does not guarantee interoperability.

4.3 HL7

Health Level Seven (HL7) is an international standards development organisation that publishes standards for healthcare interoperability.[6] HL7 publish a range of interoperability standards including HL7 V2, HL7 V3 and CDA, and the HL7 FHIR.

4.3.1 HL7 V2

HL7 V2 refers to HL7's currently most used health standard from HL7 first released in 1989 and deployed internationally.[7] It is fundamentally a messaging standard and early versions focused on 'ADT' messages, messaging relating to the admission, discharge and transfer of patients. Such messages are sent as 'triggers' and therefore adopt a 'push' model of health interoperability. HL7 has grown organically and iteratively over many years with an increasing number of message types including those to record clinical observations and laboratory results for example. The latest version is HL7 v2.6 which was approved as an ANSI standard in 2007.

[6] see `http://www.hl7.org.uk`

[7] see `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185` and `http://www.hl7.org/about/FAQs/index.cfm?ref=nav`

4.3.2 HL7 V3 and the RIM

HL7 V3 was developed from 1992 to define a Reference Information Model (RIM) describing healthcare-related messages and trigger events relating to those messages. The RIM defines an object-orientated model in which types are sub-specialisms of a more generic type. For example, in the same way as bicycles and cars are *types* of vehicles, a 'Person' is a type of a 'Living Subject' but veterinary patients such as dogs and cats are represented as a 'Non-person living subject'. As in object-orientated programming languages, specialist models inherit attributes and behaviours from their more generic parent types.

The three core classes in HL7 RIM are 'Act', 'Entity' and 'Role':

Act An act is a record of something that has or will happen. This will usually include what has been done, to whom, by whom, when, where and how and possibly documenting why.

Entity An entity is a living or non-living thing such as a person, animal or organisation.

Role A role represents a skill or competency of an Entity, such as patient, employee, place or organisation.

The HL7 V3 standard is large and complex and many healthcare organisations with an existing infrastructure built using HL7 V2 have been reluctant to adopt the newer standard so HL7 RIM is not used as much as HL7 V2 internationally.

4.3.3 HL7 V3 CDA

The HL7 V3 Clinical Document Architecture (CDA) defines a document-based information model in which components of the HL7 V3 RIM is used as a header together with a document body consisting of a mixture of unstructured and structured data. As I discussed in Section 3.4, a document-based architecture is an appropriate abstraction of real-life processes and explicitly ensures that clinical information is recorded together with its context; such a document can stand in isolation and be understood by the reader. When contextual clues are removed,

interpretation of structured and unstructured information is potentially hazardous. A document can be immutable once created and document lifecycle and process management including creation, editing and repudiation can be modelled in a straightforward manner.

There are three CDA levels:

CDA level one has a header and human-readable body usually in an unstructured format such as free text or file types such as images or documents (e.g. Adobe PDF).

CDA level two extends level one by including more structured data within the body of the document.

CDA level three allows highly-structured data to be encoded at a high level of granularity.

With the document paradigm limiting mismatch between model and real-life (Section 3.5.1), adoption of the CDA standard has been widespread internationally. It is the most adopted HL7 V3 standard.[8] In addition, the different CDA levels permit flexibility in the recording of unstructured and structured data, easing adoption of the standard. Such an approach permits implementers to use level one of the HL7 CDA to store relatively unstructured documents initially but evolve over time to permit newer applications to store more structured information and yet remain interoperable.

4.3.4 FHIR

FHIR,[9] Fast Health Interoperability Resources, is a new framework created by HL7 which uses modern web services over HTTP to create, edit and share modular resources. Importantly, the HL7 FHIR standards are free to use without restriction and focus on data standards and the implementation of those standards within clinical systems.

The use of FHIR does not mandate that existing and new clinical systems use FHIR standards to define their internal architecture or internal *storage* formats,

[8] see `http://www.hl7.org/FHIR/comparison-cda.html`

[9] see `https://www.hl7.org/fhir/summary.html`

but the use of FHIR can provide an open and standard interface to permit interoperability between different systems created by different vendors. As such, an application or service can provide access to data in an open and extensible format by providing a FHIR server and consume different data as a FHIR client.

There are a range of FHIR frameworks including:[10]

Search within a resource, such as searching for a patient.

Operations on a resource [11], such as fetching an encounter record or processing a message.

Documents representing a composition of other resources with a fixed presentation linked to context.

Messages to allow messages to be sent in response to a specific event.

Services to provide a representation of a service-orientated architecture within healthcare.[12]

There is backwards-compatibility so that FHIR documents can contain HL7 V3 CDA documents. FHIR potentially improves the integration and interoperability between disparate systems. The standard uses open web standards over the HTTP protocol and so development can be straightforward. However, FHIR is still under active development at the time of writing and the current release is designated as a "draft standard for trial use". As such, future versions of FHIR may result in non-compatible future changes.

4.4 openEHR

openEHR is an open standard for the recording of health-related data maintained by the OpenEHR foundation.[13]. It defines a multi-level information model in

[10] see `http://www.hl7.org/FHIR/implementation.html`

[11] These are in addition to the standard CREATE, READ, UPDATE and DELETE REST-based interactions

[12] see `http://www.hl7.org/FHIR/services.html`

[13] see `http://openehr.org`

which the core model (the 'reference model') does *not* include clinical information but focuses on core generic abstractions and process. *Archetypes* represent the clinical information model, with models created, edited and curated as part of a 'clinical knowledge manager' (CKM). This multi-level modelling approach means that health professionals can undertake clinical modelling independent of the underlying implementation. In addition, small and re-usable domain models can be aggregated into a 'template'. For example, if a domain model contains 'blood pressure' and 'heart rate' then this model can be incorporated within a larger 'template' representing an emergency unit admission.[14]

In addition, there is a new openEHR REST-based application programming interface (API) which proposes to define standards to which an openEHR implementation should adhere. However, the development of these standards is still in development and has not yet been finalised.

4.5 LOINC

LOINC is a freely available international standard for tests, measurements and observations.[15] Although limited in scope, it has been adopted by HL7 as the standard code system for laboratory results. In addition, there is a project to map LOINC codes to SNOMED-CT and vice versa.

[14] see `http://www.openehr.org/releases/AM/latest/docs/Overview/Overview.html`

[15] see `https://loinc.org`

5 Technical architecture

Software engineering, like all types of engineering, is complex and error-prone. In order to tackle a problem, architects and engineers usually break down a problem into smaller discrete tasks that can be performed sequentially or in parallel in order to accomplish the final goal.

This section is aimed at clinicians and managers to provide a high-level understanding of the principles of architecture in information technology. As such, it includes sufficient technology jargon to make the case for healthcare enterprise software to adopt a multi-tier layered architecture built upon a service-orientated design. I think it is important for clinicians involved in informatics to be aware of the technical considerations of their design and to understand the building-blocks of robust clinical information systems.

5.1 Have an overall vision but plan for change

Understanding of the final goal is critical in order to understand how to formulate the steps required in order to reach that goal. In some domains, an understanding of the whole of the problem domain is possible by individual team members, but in many large engineering problems, whether it is building an electronic health record or constructing a new hospital, different specialists are responsible for different aspects of the design and implementation and must work together to achieve the final aim. In large complex systems, coordinating the work of multiple teams becomes increasingly complex and time-consuming. For example, an electronic patient record system may consist of separate systems, subsystems or applications in order to handle patient registration and demographics, laboratory results or radiology requesting. The functionality of the whole therefore requires a number of different applications supporting these different clinical processes to work together.

An overall vision must also carefully consider consider future functionality in order that a given solution does not limit future developments. As such, even when the initial scope of services is limited, consideration as to how future devel-

opments will occur and how they will be incorporated is vital in creating system longevity.

5.2 The model-view-controller (MVC) design pattern; ensure separation of concerns

A single application running on a single device, such as a word processing application or an email client, will usually be carefully structured internally into components with different tasks. Many modern applications use a design pattern called the "model-view-controller" (MVC) pattern. This results in software code in three distinct but related domains. The model reflects the data structures, processes and business rules within the application. The view relates to the user interface and how users interact with the application. Linking the two is the controller layer which acts to manipulate and use the model and coordinate the view layer to display and allow user interaction.

This separation of concerns is an important design pattern. At first sight, it may seem to make solving the high-level problem of writing an application more difficult. Indeed, for very small applications, it is frequently quicker to write what is colloquially termed 'spaghetti code' in which logic relating to the business rules and the user interface are intermingled. As an application grows however, the software becomes increasingly difficult to understand and maintain. As business rules are spread throughout different parts of the application, small changes in one module result in untoward knock-on side-effects. A short-term gain in productivity is made at the expense of more difficult ongoing maintenance and development.

An MVC pattern and ensuring separation of concerns can be used within a single application or can be used in large-scale enterprises to coordinate and structure multiple applications, middleware and core services together as a unified whole.

5.3 Adopt test-driven development

An important part of software development is proving that a system does what it is supposed to do; that it is 'correct'. This may sound simple, but many projects

face situations in which the problems that are to be solved change over time. Such issues make it important to ensure that specific projects are limited in scope, with well-defined requirements and interfaces with other systems. Under such circumstances, a project can undergo a range of automatic tests to ensure that it meets and continues to meet expectations. Such tests are commonly called 'unit tests' when a specific unit of functionality is being tested, 'integration tests' when a project's integration with other systems is under test, 'functional' tests when assessing the results of a number of systems against what is 'correct' and 'acceptance tests' when preparing software for deployment.

'Test-driven development' is a methodology in which development starts with the tests that will be used to test that project's functionality even before implementing that functionality.

In large code-bases, small changes can have unintended consequences and unit tests can capture such changes automatically. For example, in my SNOMED CT terminology server, I have incorporated the Dictionary of Medicines and Devices (DM&D) which includes data on pharmacological and other products available within the United Kingdom. I wrote unit and functional tests to test mapping a virtual medicinal product to actual medicinal products and to prove that the software was able to deduce the available routes for any specific drug. A small change to optimise the software and add new functionality unexpectedly affected the logic behind those deductions and this was fortunately picked up by the automated test suite during the build process.

When there is poor separation of concerns and little regard to architecture, automated testing is difficult. Important business logic is mixed-in with other functionality such as that relating to the user interactions and such software is difficult to test. A product may perform so many different pieces of functionality that it is difficult to have confidence that it and its integration with other systems is 'correct'. Importantly, when user interactions are required to test important components of software, automated testing is much more difficult as test harnesses need to mimic user interactions instead of simply proving that a piece of logic gives the intended results when give a particular test of inputs. An example is shown in Figure 3 in which a monolithic application with several different important pieces of functionality, spanning user interaction, business logic and in-

tegration with other systems is tested. How can we be confident that the product performs 'correctly'? In fact, this is a common problem in most organisations that focus on procuring and developing 'systems' to solve business problems. They are developed and integrated at a level which makes it difficult to be confident that they are fundamentally 'correct'.

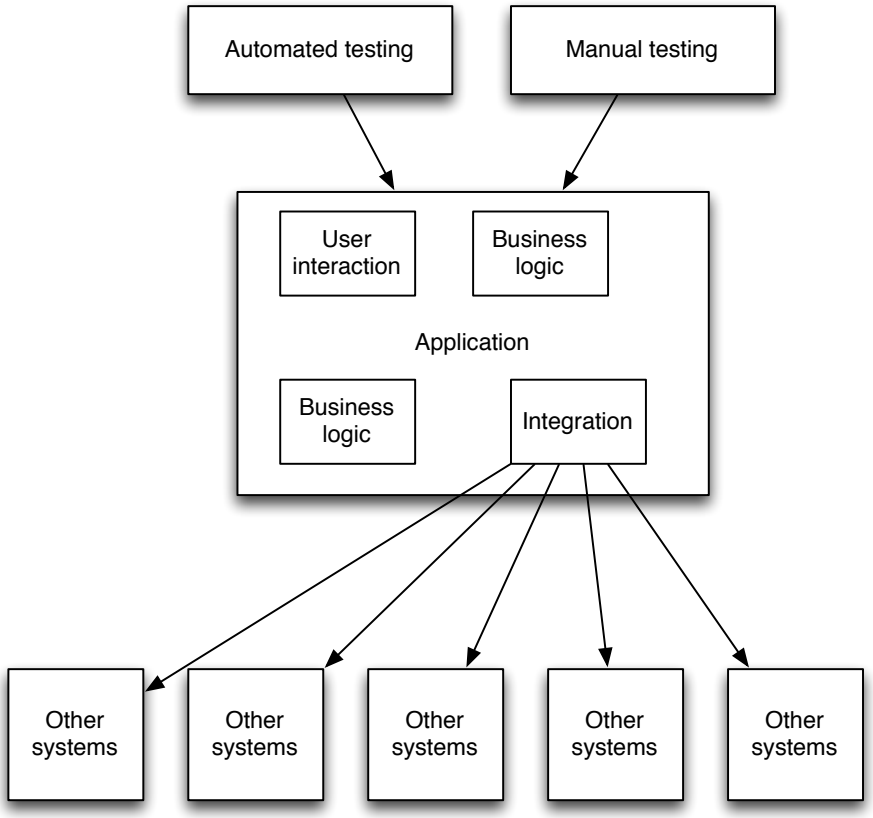

Figure 3: Testing a monolithic, vertically-integrated, poorly structured application.

A different approach is shown in Figures 4 and 5 in which applications and services are organised in a layered approach. Here it is possible to test each discrete module in isolation and when integrated. Important integrations and interfaces can be tested using 'mocked' services, in which a lightweight façade is substituted for an external system allowing testing controlling for external dependencies.

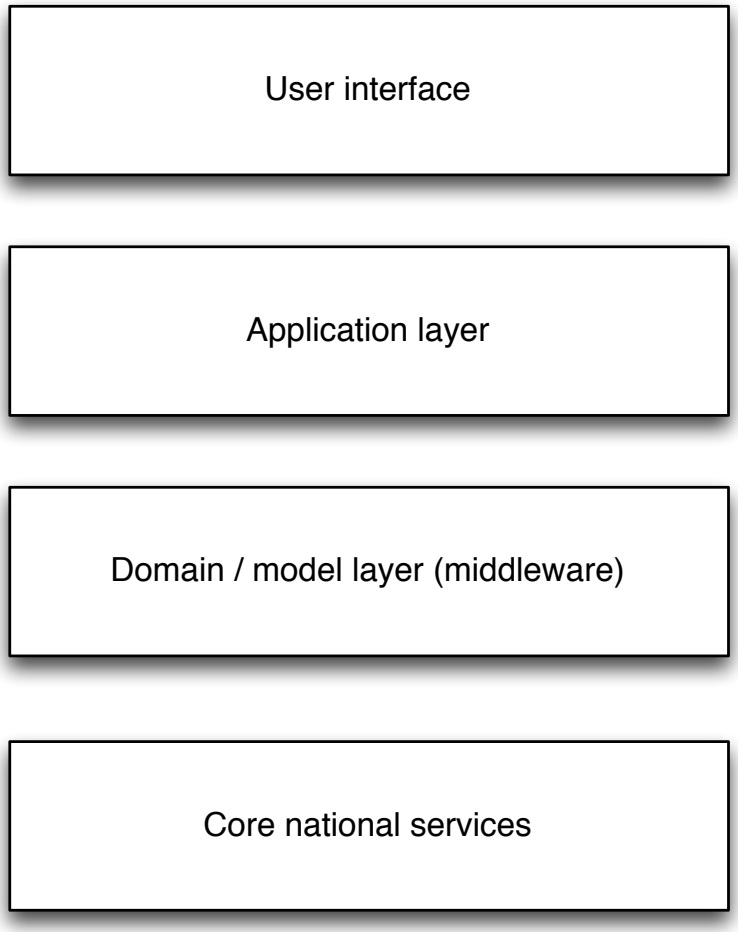

Figure 4: A multiple tier layered architecture model

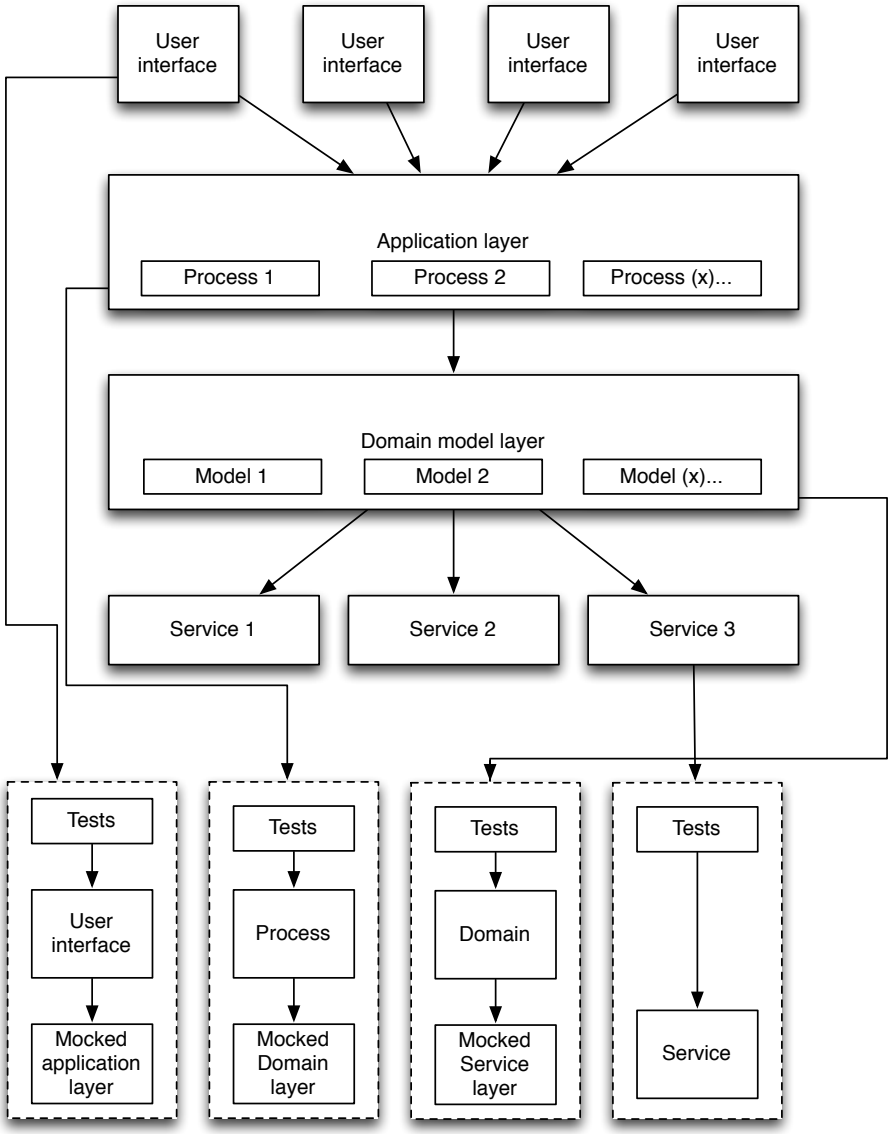

Figure 5: Tested a structured layered architecture

5.4 Favour automated testing and deployment pipelines with continuous delivery

During the development of a software application service, it can be tested to determine whether it passes those tests and fulfils the 'contract' inherent in its implementation. However, as outlined in Section 5.3, test driven development can test individual software components and their wider integration. Robust and automated integration tests requires automated software build tools to take the computer code and turn it into software together with automated deployment and then testing. If these processes require manual user intervention in which developers 'hand-off' their code to the deployment team then it is difficult to have a rapid test-build-test lifecycle. In many organisations, developers finish their code and perform a 'release'. This is subsequently deployed into a test environment in which a range of automated and manual tests can be performed, including those of the software itself and the user interaction. This quality assurance cycle can take weeks.

Instead, automated pipelines are now available in which software components may be developed and tested in isolation, with build processes subsequently automatically deploying those products into a test environment which has been preconfigured. An example of this approach is that of the Docker[16] system which isolates software components into lightweight virtualised environments with a declarative model describing how those software components are put together for testing and deployment.

Continuous delivery means deployment of system applications and services is routine and straightforward resulting in development, testing and integration occurring continuously after even small changes rather than waiting for a 'code freeze' and then entering each stage manually and laboriously as in conventional deployments.

[16] see `https://www.docker.com`

5.5 Use a layered structure within applications

It is possible to adopt a layered organisation within a single code-base. This makes sense for smaller applications in which, although components are themselves loosely-coupled, a component is used only within that single application. It can be developed and integrated into a larger code-base either at runtime via a dynamic library or at compile-time and yet be tested in isolation.

However, for large enterprises in which different components may be used within multiple applications, multiple business processes and contexts, it makes sense to allow components to be deployed in isolation and to be used across a network in a 'distributed' fashion. For example, in my SNOMED CT service, I originally embedded the functionality within a single application but as that application was built using a layered architectural approach, when I needed to allow other applications to leverage that same functionality, it was trivial to extract that SNOMED CT service into its own standalone service.

There are a range of advantages and disadvantages to both approaches. Embedding functionality within a larger application makes sense for smaller pieces of shared code and has no overhead relating to the exchange of information over a network; instead as components share the same memory space within the same application, data exchange is very fast. For large components, it is usually more appropriate to create a standalone service that can be used across the network by a variety of client applications. In the 'distributed' model, it is possible to optimise by caching results and replicate a single service to create highly-available robust systems less vulnerable to failure.

5.6 Use loose coupling between components; design by contract

An important facet of the layered approach is the way different modules are coupled together. Tightly coupled components are difficult to develop and test in isolation; developers need understanding of the consequences of changes in one component on another whereas in loosely-coupled components, an individual component has limited but well-defined exposure to other components so that it is

possible to put the component within a test harness and exercise the system independently from the whole. Consider formalising a coupling between components with a two-step process in which: 1. a formalised contract is written provide the specifications for a component and its interactions 2. a test-harness is written in order to demonstrate that a component meets its specifications include its interactions with other systems

Such loose-coupling ensures that clients of that component do not necessarily need to know *how* a component achieves its outcome, only that, given the specifications in the contract agreed, a component *will* provide the outcome agreed. This results in components that act as 'black-boxes' as in Figure 6 in which a particular business activity is performed within a self-contained component.

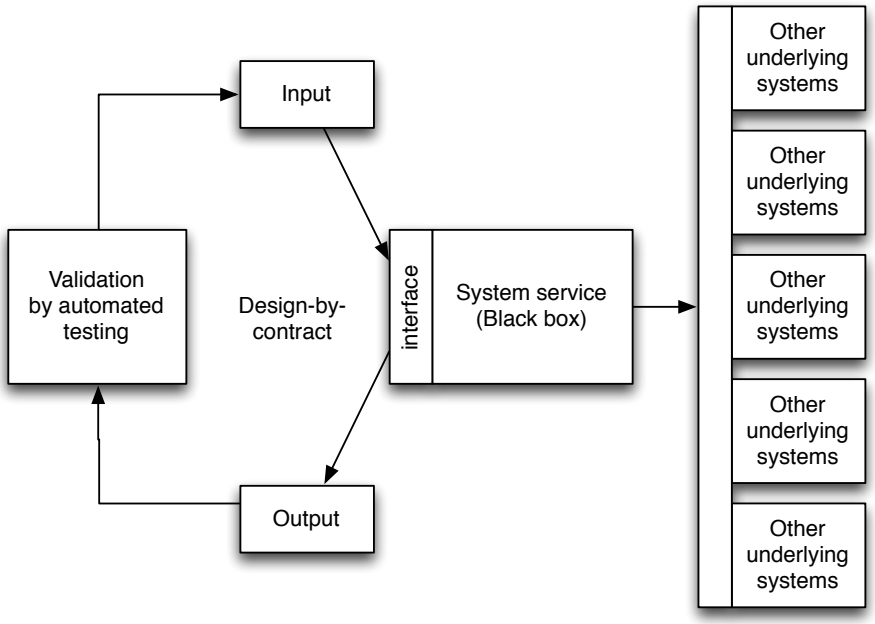

Figure 6: Design-by-contract, loosely-coupled services.

Decoupling functionality allows services to be develop and evolve at different speeds. In addition, decoupling means that services can be deployed flexibly to suit business requirements. For example, a service which depends on a high-performance subsystem may embed that system within its own application to en-

sure adequate performance and reduce the overhead of communication over the network.

5.7 Adopt a service-orientated architecture

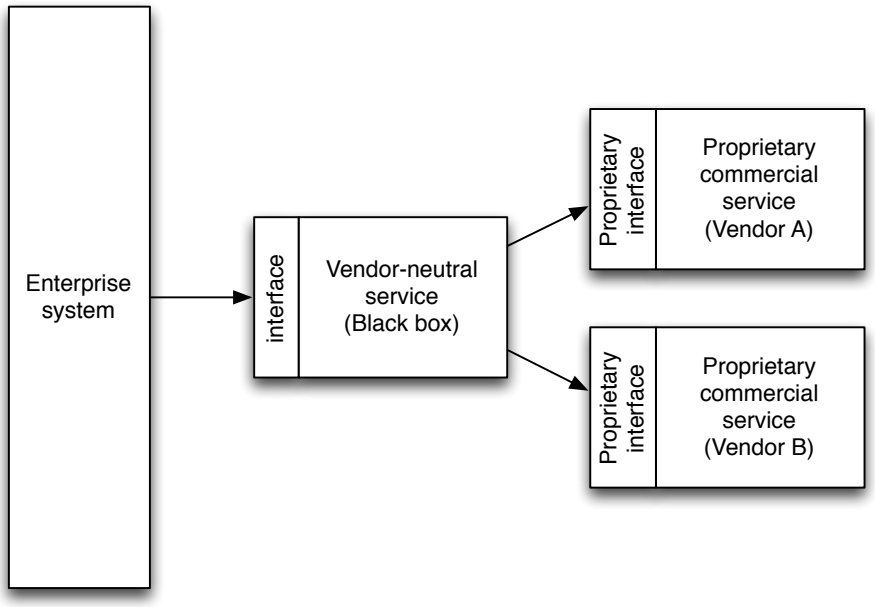

Figure 7: Encapsulation of services within a service-orientated architecture.

A service-orientated architecture (SOA) is a formal specification for the provision of loosely-coupled of services as defined in Section 5.6. Service-orientation is a way of thinking in terms of services and service-based development and the outcomes of services in which a service[17]

1. Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports)

2. Is self-contained

[17] `http://www.opengroup.org/soa/source-book/soa/soa.htm`, accessed 10th November 2016

3. May be composed of other services

4. Is a “black box” to consumers of the service

An important element of a SOA is the encapsulation of possibly complex behaviour behind a well-defined interface. It offers the potential to completely replace large parts of an enterprise application without change to other parts of the application. Such encapsulation also results in vendor-neutral solutions not tied to a single commercial solution. For example, if the specifications for a SNOMED CT service are well-defined, then a range of commercial solutions may be encapsulated by a vendor-neutral layer which handles a switch from one vendor to another. Such a model is shown in Figure 7 in which a solution from Vendor A could be switched to a solution from Vendor B because both services are encapsulated by a vendor-neutral interface providing services to the rest of the enterprise. Such an interface may adopt an international standard such as HL7’s FHIR (Section 4.3.4).

5.8 Use a layered architecture with services

As services may themselves encapsulate other services, then services naturally adopt a layered structure in which high-level services *use* lower-level services. Low-level services may frequently be generic pieces of functionality not necessarily used only within healthcare, but provide ‘computing services’ such as data storage, replication or messaging. High-level services will leverage such low-level services in order to perform their functions, which will usually be more tied into the processes of healthcare and much less generic. For example, one might have a demographic service that matches a patient based on a range of identifiers. In NHS Wales, this service is called the “enterprise master patient index (EMPI)” which currently is provided by the IBM Initiate platform, a generic service that is used in a wide-range of industries such as banking, security and consumer-goods[18]. However, while it might be feasible to expose the interface to this service to systems across the enterprise, there are important reasons why it would be better to encapsulate this service with a vendor-neutral service:

[18] `http://www.ibm.com/support/knowledgecenter/SSWSR9_10.1.0/com.ibm.mdshs.hubover.doc/topics/c_hubover_overview.html`, accessed 10th November 2016

1. The commercial solution may need to be replaced if a vendor no longer actively maintains or supports their software. If a vendor-neutral encapsulation is provided, then no changes in other systems may be necessary if the service is replaced with another.

2. Additional business-logic may need to be performed when client applications perform patient demographic searches, such as recording an audit trail or supplying additional contextual clues to the underlying service based on business rules and processes.

Higher-level services frequently deal with a concepts at a higher-level of abstraction than lower levels.

5.9 Who is in control in a layered architecture?

It is important to consider how to partition and encapsulate services within a layered architecture. Low-level systems should not have knowledge about the software and services that use their functionality. Higher-level services understand and drive lower-level services but themselves do not control the services at a higher-level than themselves.

For example, a service which provides functionality to allow users to sign-up for a mailing list may consist of two user applications, one web-based and one running on a mobile device, a sign-up service that both user facing application use, as well as a business model and data storage layer beneath that. It is easy to see how a user-facing application may drive and control the sign-up service. But what happens when a service at a lower-level needs to communicate something to a higher-level? Traditionally, other mechanisms are used such as callbacks or observers, in which high-level systems register that they wish to be informed of progress or results in a lower-level subsystem and that system itself knows only enough to call that 'callback' or send a message to that observer.

5.10 Model process and not simply data

As I wrote in Section 3.4, it is important to model not only the data relating to healthcare but the processes as well. Processes and rules are liable to change

more frequently than data and will tend to vary between departments or organisations. Modelling process allows these differences to be explicitly declared and accounting for in user-facing applications.

It is tempting to 'hard-code' business logic into applications and services. However, it then becomes more difficult to understand or indeed change that logic in the future to meet changing requirements. Therefore, many enterprises use rule-engines to create a declarative model to shape the behaviour of software. As such, the behaviour is defined by logic rather than a set of imperative commands performed sequentially. A rules-engine[19] is a generic way of expressing business rules and can be used in a wide variety of industries particularly in environments in which business rules change over time. In this way, a rules engine encapsulates the business logic for an enterprise system allowing a separation of concerns. Alternatively, business rules can be provided by high-level enterprise services that may leverage a rules-engine internally but also enforce business rules and process dynamically based on other contextual clues.

5.11 Favour immutability of data, but understand that there is no one 'truth'

Immutability refers to an approach to software engineering in which data are unchangeable. Such an approach is particularly suitable when considering medical records and processes. Indeed, from a medico-legal perspective, all data should be regarded as immutable as it is usually necessary to be able to view the state of the record at any point in time.

Many authors believe that supporting an audit trail is sufficient to deal with changes to data. For example, one approach to the recording of diagnoses or allergies is a canonical model in which users may create, edit or remove allergies. Combine that with an audit trail and it is possible to show the current state of that model as well as look though a history of the changes to identify who did what and when.

However, there is a mismatch between this approach and what happens in real-life (see Section 3.5.1) as most clinicians would not seek to share a single

[19] see Drools (`http://www.drools.org`) as an example

page between all services and expect it to be sufficient for their needs. What is considered a 'diagnosis' for an oncologist may be different to that of a different medical specialty as diagnosis is a much more complicated concept than one might initially think and is dependent on context. A more detailed analysis of how to model diagnosis is shown in Section 8.2, but the essential message is that providing an audit trail does not magically make a data model usable by multiple health professionals over time in the care of a single patient.

For example, a diagnosis of 'angina' made by a cardiologist as part of an episode of care in which the patient undergoes an elective coronary angiogram which shows diffuse coronary artery disease is fundamentally different to a diagnosis made by a non-specialist on the basis of a history of chest pain.

Instead, considering all data to be immutable after original creation forces an approach in which clinical modellers and information technology experts adopt a transactional document model. Data can be updated but as a result of additional processes in which old data are deprecated and replaced with a new version of the 'truth'. The benefits of such a model are demonstrated in Section 8.4 while a mutable model of a prescription chart has unintended and potentially devastating consequences to patient safety. In addition, immutable data can be persisted and made available across a distributed enterprise using a unique resource identifier that is guaranteed to not change. It is of course possible to make changes to data but such changes must be modelled as a process of change and not simply changes to a canonical source of the 'truth'. There is no one 'truth' in medicine, but an ever-changing version of reality fundamentally dependent on context and opinion.

5.12 Make services idempotent

A service that is idempotent can receive the same call or request repeatedly while producing the same result. When combined with an information model predicated upon permanent resource locators that reflect immutable data, idempotency can make systems reliable and more correct. For example, a service providing a centralised store of clinical documents might allow other services to 'send' it clinical documents. If, for example due to network latency issues or user error, client software sends the same document twice, it should not create a duplicate

document. In order to achieve services that are idempotent, frequently client software will need to supply a fixed, permanent unique identifier for the request so that services can match that document with the previous version and replace that version rather than create a duplicate.

5.13 Adopt uniform resource identifiers for access to resources

A uniform resource identifier (URI) is a unique identifier for a particular resource. Combined with a service that can resolve such identifiers and obtain a representation of that source, a URI allows services to link to resources in a safe, permanent and future-proof way.

5.14 Are there exceptions to the use of a layered architecture?

Most health enterprises consist of large vertically-integrated applications which provide functions to interact with users, apply business rules and processes as well storing and retrieving health data. Examples of such systems include existing systems used in general practice and legacy hospital-based systems but also include newly procured systems such as the new Welsh Emergency Department System (WEDS) and the new Community Clinical Information System in NHS Wales.

For legacy software, such systems must be brought into the national architecture in order to ensure that we do not create 'data silos' in which health data created in one system is not made available to other systems. However, it is difficult to justify the ongoing development of similar vertically-integrated applications in the future. It is possible to stipulate a robust interoperability and standards framework to try to link these different systems together. However, such an approach does not magically result in scalable and efficient clinical systems. As shown in Figure 8, integrating multiple systems can fail when a large number of different systems must interoperate simply as a result of requiring so many interfaces between systems.

In some enterprises, architects have mandated the use of a health information exchange (HIE) which acts to exchange information between vertically-integrated applications in different organisations (Figure 9). Such an approach can be useful

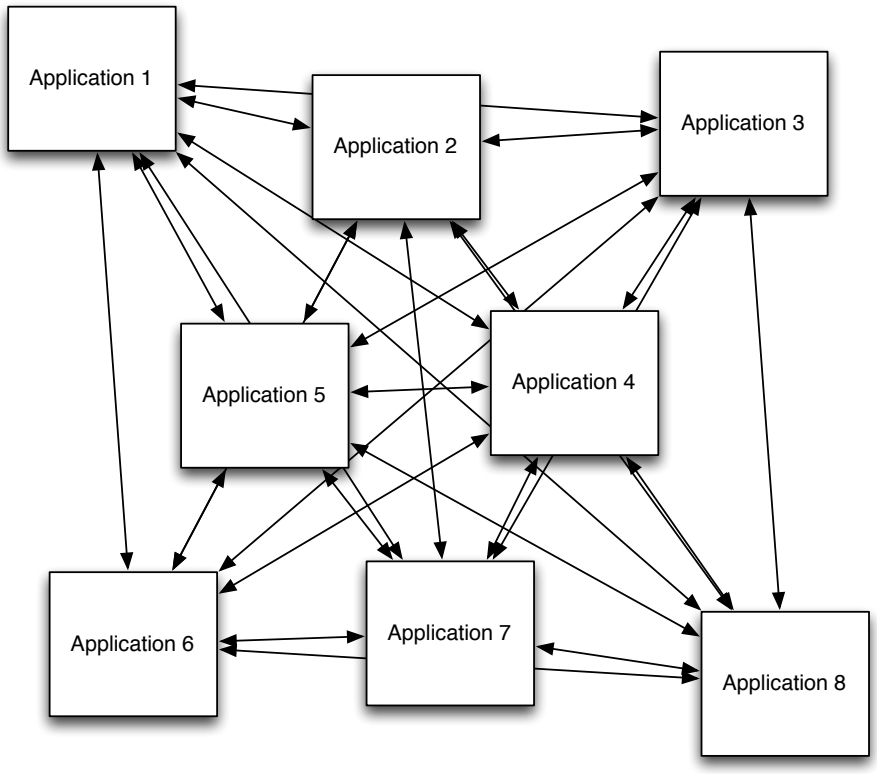

Figure 8: Interoperability between systems when there are multiple systems.

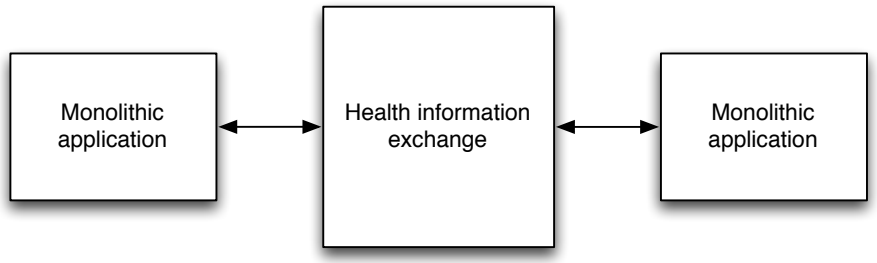

Figure 9: Connecting systems using a health information exchange model.

when one has a number of legacy applications that must be integrated within a wider enterprise but like in Figure 8, does not scale with multiple systems and services.

It is therefore more appropriate to adopt a layered architecture in which legacy vertically-integrated application must inter-operate (Figure 4). Within each layer, services can be horizontally-integrated.

5.15 Push vs. pull

'Push' and 'pull' are two techniques for the transmission of information from one module, system, application or layer. Traditionally, many healthcare systems use 'push' which means that something happens in one system and a message is 'pushed' to another system which needs to deal with that message and perhaps update its own internal state. This is commonly used in HL7 in which messages regarding admissions, discharges and transfers (ADT) are probably the most common. 'Push' therefore has an advantage in that it is widely used within existing healthcare software. However, the historic use of 'push' is a reflection of the need to link together many vertically-integrated applications within a single enterprise.

'Push' based systems usually use some form of messaging platform which allows messages to be sent asynchronously with varying guarantees that a message will be received and with received messages populating a stack which can be duly processed by receiving applications. As such, that receiving application must decide whether to discard the message or deal with it at the point it is received.

A 'pull' approach means that one system requests information from another, rather than being 'pushed' to it. Modern approaches such as the 'Fast Healthcare Interoperability Resources' (FHIR) adopt a mixture of both 'push' and 'pull' paradigms.

At the core infrastructure layer, it makes sense to continue to use a 'push' approach, particularly when dealing with legacy systems. This low-level messaging is a reasonable way for the horizontal integration of systems at the infrastructure layer of a layered architecture. However, most clinical systems do not wish to answer questions like "send me a HL7 message when a patient is admitted", but instead wish to ask "who is in bed 9 on ward 5?". Notifications such as "let me

know when a patient is admitted on ward 5" might be needed, but this can be achieved without necessarily resorting to low-level messaging. At higher levels of the layered architecture, software must deal with higher levels of abstraction and do not necessarily need to know *how* the underlying software has arrived at that status. This is a fundamental characteristic of a layered approach and is an illustration of an important software development technique of developing to an interface and not an implementation. The underlying implementation may change in the future, but software at a higher level of abstraction does not need to know or indeed be changed.

A 'push' approach is acceptable for simple enterprises but when one wishes to ask more complex queries relating to dynamic changing state, then a 'pull' approach is more sensible. For example, within NHS Wales, because laboratory results are sent from the laboratory information management system (LIMS) using 'push' messaging, investigation results are not only stored in a national repository but other applications also need to store those results so that they can be analysed over time. For example, this occurs within some individual health boards in Wales as well as dedicated specialty-specific software such as the VitalData system used in nephrology. However, such systems cannot analyse trends for tests performed before that system started receiving and storing messages about that patient. In addition, when a message is received, how can that information be integrated with other information such as diagnostic and treatment data, particularly if a 'push' model is adopted for the record of that type of information.

The final problem with the 'push' approach is information governance. A system cannot vouch what recipient systems are able to do with the information that is sent to them, and there is no contextual information to help make such a decision. With a 'pull' model, both systems have a say in whether information exchange can take place, with a service acting on behalf of a user able to provide important context such as the setting for that encounter (e.g. a clinical encounter in an outpatient clinic) and who that user is (e.g. the user is a consultant neurologist who is linked to the patient by virtue of their involvement in multiple sclerosis services), and the service from which data is 'pulled' able to check and validate the request before returning the resource or information required.

5.16 An open platform for the health enterprise

After considering the issues raised in this section, the final conclusion must be that we need to create an open "platform" on which clinical applications may run.

Such a platform needs to consist of a range of information and process models that reflect healthcare and its processes as outlined in Section 3. Importantly, there needs to be a contextual vocabulary reflecting the nouns on which clinical processes are undertaken. Such context reflects the higher-order abstract concepts of healthcare including what is conceptually considered to be a 'user', a 'patient', an 'organisation', a 'clinic', a 'specialty', a 'clinical pathway', or a 'clinical service'. Such concepts must exist and be modelled across different systems in order to achieve robust interoperability. An appropriate information model encapsulates these nouns combined with a multi-tier service orientated architecture creates a *platform* on which applications can run and interoperate with other systems.

The "platform" thus becomes both a medical record and manages the workflow and processes relating to that record. A permanent medical record stores information about what has been done, to whom, by whom, in the many processes of healthcare. Such a record needs to make use of appropriate data standards, terminology and other concepts necessary for interoperability. This data layer acts as a platform on which workflow and clinical processes run. Workflow and process management supports the day-to-day business of health organisations in caring for patients. Such a "platform" can evolve over time, initially acting as a health information exchange (see Section 5.14) for legacy existing software to provide interoperability but act as a foundation for new and modern applications in the future.

As such, an immediate requirement for any such platform is an understanding of the levels of interoperability available to existing and new software. I suggest the development of an accredited interoperability framework in which software can be assessed and graded to ensure that future procurements are compatible with the platform. It is perhaps possible to define levels of interoperability graded I-IV, in which highly interoperable application are graded the highest.

An example of an interoperability framework is shown in Table 2

Category and level	Requirements
Information model - how are data stored and exposes to the national platform	
Level 1	Uses own proprietary data storage with limited proprietary API
Level 2	Uses own proprietary data storage with mixture of proprietary and open interoperability API
Level 3	Uses own open standard structured model with open API
Level 4	Uses national platform provided, structured data model
Fixtures - data representing organisations, users and other 'fixed' data	
Level 1	Limited use of national reference data set
Level 2	Uses mixture of own and national reference data set
Level 3	Extensive use of national reference data set
Level 4	Uses national reference data set
Terminology - terminology used within the information model	
Level 1	Minimal use of SNOMED CT or another terminologies
Level 2	Moderate use of SNOMED CT or another open terminology
Level 3	Extensive use
Level 4	Uses SNOMED CT and other terminologies as lingua franca bound to information model

Table 2: An example of interoperability scoring for services and applications. The information model refers to how data are stored and exposes to the national platform. Fixtures refers to the data representing organisations, users and other relatively 'fixed' data. Terminology refers to the terminology used within the information model.

A platform would consist of a variety of reusable components at all levels of the multi-tier architecture. As an example, the user interface layer might consist of components which can be linked dynamically into other application to allow users to list, search and view clinical documents and the application layer may

consist of programming interfaces which allow other software working at that layer to themselves programmatically list, search, filter and fetch clinical documents. Thus, software becomes horizontally integrated with the platform.

5.17 Data analytics and an enterprise-wide warehouse

With a robust platform built using open data standards, it is possible to routinely analyse group-level patient data to monitor service performance. Frequently, data from multiple systems can be aggregated and linked within a separate reporting system such as a data warehouse. It is important that data analytics can occur in a timely way in order to satisfy the requirements as outlined in Section 2.2, but many reports need to be embedded within clinical processes, such as a list of patients waiting or who have been discharged. While a separate data warehouse has advantages for off-line processing, many analytics should be available to clinical staff in real-time.

5.18 User-facing applications

I feel very strongly that a single application will not ever meet the demands of clinical staff. Indeed, most organisations run a variety of different applications to suit different departments and different workflows. For example, most emergency unit applications support the workflow of an emergency department showing staff who is waiting and how long they have been waiting. Should all user-facing functionality and interactions sit within a single application or be broken down into discrete context-specific applications?

Of all layers of a multi-tier layered architecture, it is the user-facing applications that have the greatest need to reflect clinical processes and workflow and will therefore need to integrate and show different information at different times. Indeed, most hospital specialists will have different needs depending on their specialism and my own experience of patient-level disease-specific dashboards is that a dashboard summarising information for a patient with motor neurone disease will be very different to that with chronic kidney disease.

In fact, a platform approach means that decisions on how to plan and develop a user-facing application (or applications) can be made and can be changed over

time. I believe that user-facing applications should be lightweight, workflow and process-bound, easy to develop because most of their complex functionality is already provided by other services within the platform. In this way, application development can be low-risk and low-cost and can work to innovate. Imagine an NHS hack day in which applications are developed that take patient data and perform analysis over time of changes in renal function for example or bring together ambulatory data captured from patient's own devices with treatment and interventional data such as levodopa dose and deep brain stimulation in Parkinson's disease.

The problem with multiple applications is that historically applications have been vertically integrated, including layers supporting user interaction, business logic and data storage and so it has been difficult to integrate these applications into the wider enterprise, particularly as most do not adopt modern data standards. However, these issues are minimised when user-facing applications are a slim, lightweight wrapper around core pieces of infrastructure which actually do the hard work.

However, just because it is possible to have multiple applications, it doesn't necessarily preclude building a main 'portal' which provides access to the 'platform' in a generic but comprehensive manner. Such a portal may provide only read-only access to many important services; such a portal would not aim to replace a theatre management service but might need to present data from such a service in patient or aggregated views.

6 Organisational structures

6.1 Bind organisational structure to the technical architecture

In Section 3, I discussed the interdependencies between requirements gathering, health data and process models and implementation and concluded that the most appropriate technical architecture to support such a methodology is a multi-tier layered service-orientated platform architecture (see Section 5). As such, I concluded that the models and the implementation need to be kept tightly bound. As such, the organisational structures to create, support and continuously improve the software underlying such an architecture must reflect that architectural structure.

As such, I advocate small teams with responsibility for single services with limited scope and formal contracts relating to the interfaces between teams. Such contracts should apply not only to the software interfaces between teams (see Section 5.6) but to the interface between the team members themselves in terms of working relationships and contact.

However, as I explained in Section 5.1, an understanding of the overall design goal is critical to shape the overall infrastructure and high-level functional requirements. Clinicians and software developers, managers and architects must work to shape and guide evolution of the overall infrastructure. Such a team, organised as an architecture board, must represent a range of the organisations that have applications running on the national architecture. In addition, they should have oversight of projects that make up a "platform".

6.2 Strategy, procurement and planning

There must also be organisational structures to support the process of procurement to ensure software products meet guidelines on interoperability and data standards allowing assessment and grading according to the established interoperability framework (see Section 5.16).

In many projects, a group of enthusiasts apply for money for a particular project, are successful in their application and then are faced with the development and roll-out of that product. Many such projects, both research and clinical, adopt this approach. In these situations, the NHS Wales informatics services and indi-

vidual health boards need to integrate that project into the wider architecture. A more formal model of procurement must be instituted in which all procurements are graded on their level of interoperability with the national platform. In order to successful host innovative projects, NHS Wales and the informatics service must be involved at an early stage in health technology bids with a dedicated team in place to ensure that an appropriate focus on data standards and interoperability. All high-value projects must be signed-off by a national architecture board.

6.3 Clinical informatics, career development and skills

In Section 3, I discussed the importance of a team structure bringing together expertise in healthcare, informatics, data standards and technology. In order to meet those demands, any programme of work to develop clinical information systems needs to consider how to recruit, retain and professionalise those involved. Importantly, clinical staff must be given protected time in order to be involved and shape development programmes across the platform.

7 Conclusions for NHS Wales

7.1 Overview

I began by explaining the importance of the medical record (Section 2) and how it is vital to model both data and process within healthcare in creating of clinical information systems (Section 3) as well use the appropriate data standards (Section 4). I extended this discussion to demonstrate how this approach should drive a scalable, multi-tied layered technical architecture (Section 5) and that such an architecture needs a corresponding organisational structure in order to manage it safely (Section 6).

The vision for Wales must take into account an appropriate information technology platform that will support the routine capture of clinical data for use in day-to-day care, service management and clinical governance. It must provide sufficient flexibility to support the data flows shown in Figure 2 in which information captured by clinicians can be integrated with the results of biological data, patient-reported measures and patient-help ambulatory device data to improve clinical care.

A single user-facing application will not be sufficient to support these use-cases, and I recommend multiple applications focused on specific processes or workflow that integrate with a national platform. It is appropriate for an emergency unit application to be developed for an emergency unit, and a theatre system developed for the operating theatre. Indeed, I imagine mobile devices with a specific phlebotomy module leveraging the national platform, a simple lightweight client application supporting a single workflow on a busy ward so that staff know from who and where to take blood samples. We should not be afraid of multiple applications except when they are not interoperable; in that case multiple user-facing applications risk creating silos of data which cannot be used to support and improve patient care.

However, an open 'platform'-based approach does not preclude the ongoing development of a single "Welsh Clinical Portal" offering a generic approach to the retrieval of individual patient-level data. Indeed, a single portal build upon a

robust set of loosely-coupled interoperable components will be a safer and more robust portal than one that has evolved organically.

7.2 Suggested actions

I suggest the following actions for NHS Wales:

- Create an interoperability framework in which there are detailed specifications for a range of levels of interoperability. Compatibility with such a framework needs to be mandated for procurements and development of applications with NHS Wales. I suggest a detailed evaluation of the FHIR HL7 standard and consider implementing as a priority a FHIR interface for the eMPI to provide a vendor-neutral view of an individual patient as part of that evaluation.

- Break-up the business logic and services within the existing Welsh Clinical Portal software and make the user interface layer a simple wrapper around some core services. Allow those services to be used by other applications including those developed by health boards, in order to ensure a safe migration to a new national platform in NHS Wales. The development plan for WCP 4.0 should focus on removing all business logic from the user interface layers and adopt a service-orientated approach to the platform's functionality.

- Evaluate technologies in order to create a national data platform comprising a robust information model at two levels: that of a reference model which are hardcoded and act as the fixtures and scaffolding into which more flexible runtime derived models are stored, such as in openEHR.

- Ensure organisational structures are in place to support a service-based architectural platform and to sign-off health technology projects within Wales to ensure that they meet the standards stipulating within the new interoperability framework. Ensure that there are clear lines of responsibility and ownership for planned developments.

- Mock versions of the core NHS Wales services should be made available via the Internet in order to showcase the Welsh open architecture and encourage innovation. Applications built using the mocked services could subsequently be easily transferred into the live environment, subject to approval. NHS Wales should sponsor NHS 'hack days' in which technical and clinical and information technology professionals can come together to create innovation small applications integrating with a mock-up of the national architecture.

- Create a national architecture board combining skills from clinicians, informaticians, data standards experts and technology leads to oversee a new national platform built as a layered multi-tier loosely-coupled architecture with a focus on clinical modelling and data standards. Define the scope of nationally-provided applications and the national portal to core generic functionality but ensure that artefacts created in other systems are viewable as part of the patient record in an appropriate context.

- Create a national informatics clinical informatics fellowship scheme and informatics academy within NHS Wales to recruit, retain and educate a cadre of clinicians and information technology professionals in planning and delivering information technology for healthcare in Wales.

Mark Wardle
February 2017

8 Worked examples

8.1 Emergency unit systems

8.1.1 Introduction

Emergency unit (EU) systems are specialised information systems that are designed to support the processes of care in the EU. As such, their design is tightly bound to those processes of care including the prioritisation of patients and managing a service that, by its very nature, is unscheduled and works under pressure. In particular, the 4-hour waiting times target emphasises recognition that patients need to be seen in a timely fashion.

Most existing EU systems are principally focused on the managing the service rather than supporting direct patient care. For example, current EU systems within Cardiff and Vale University Health Board show the patients that are waiting and those that have been seen supporting the administration of the service with the generation of discharge summaries. However, documentation of clinical assessments is still paper-based. Large amounts of data are recorded but they are not recorded in an electronic system making it difficult to make use of these data for direct patient care or management of the service.

What is the most appropriate way of implementing effective information systems to support the EU? The current plan within NHS Wales is the procurement and deployment of a single product, EMIS' Symphony product[20] and deployment is ongoing within Abertawe Bro Morgannwg Health Board. Inherent in this procurement is acknowledgement that the EU is a specialist environment with unique requirements relating to process and workflow.

This approach seems reasonable but it is important to recognise that two large scale design processes must be undertaken in order to successfully deploy and integrate a new EU system. The first is the design and configuration of the system itself in order to suit an individual department. The second is an understanding of the interfaces between a new EU system and other components of the health enterprise platform. Indeed, product integration is frequently highly complex and if integration requires interfaces to multiple disparate systems within an organ-

[20] see `https://www.emishealth.com/products/symphony/`

isation to make it work effectively, then implementation can be error-prone and time-consuming.

As an EU application will contain multiple layers of functionality including a user interface layer, a business logic layer and its own data persistence layer, each of these layers will need some degree of interoperability with a national platform. For example, at the user interface level, users might need to be able to switch to nationally provided applications from within the EU application. At the business logic layer, the EU application will need to interoperate with a range of local and national systems in order to handle patient demographics, notify of an attendance, publish clinical document(s), request investigations and make data available for offline processing for secondary uses.

8.1.2 Analyse process, workflow and data requirements; nesting of data

As such, the procurement of an EU system must begin with a detailed analysis of process, workflow and data as well as an understanding of the legislative requirements for secondary use of information including an understanding of the 4-hour wait national target.

At an information model point of view, healthcare data is fundamentally fractal. A fractal describes structures in which patterns recur at progressively smaller scales. For instance, while a large amount of medical and nursing care occurs within an EU, for most clinical staff, a summary of that episode will be sufficient. An information model that adopts a nested design allows detailed clinical data to exist held within an appropriate context. A similar paradigm works for day case attendances in surgery, an emergency admission with pneumonia, a stay on the intensive care unit or membership of a service that supports a patient with a long-term health condition in which a high-level summary view will be sufficient.

8.1.3 Design by contract

As such, what are the inputs and outputs that are required from an EU application? At the minimum, any such application will need to be able to generate a high-level summary document detailing the EU attendance and make that document available to the national platform. Given such a summary needs to include

clinical information such as the list of problems, clinical findings, treatments administered and procedures performed, these data should be highly structured using SNOMED-CT. In order to support legacy applications, a low level of interoperability may simply mandate a very simple data model with only core information made available in a structured format. For modern and newly procured applications, it would be expected that all information relating to that attendance should be made available to the national platform.

Therefore, once process, workflow and data are mapped, a contract must be specified to define how the EU application should interoperate in which detailed specifications as to what will be exchanged between the national platform and the application itself. Such a contract may need to consider existing deployed legacy applications as well as understanding the commercial market for products, taking into account what is available. However, a product which operates at a high-level of interoperability should be scored more highly that one that can exchange only small amounts of structured information (Section 5.16 and Table 2).

Once there is a contract of interoperability and functionality, it is evident that as long as an application satisfies the requirements of that contract, it does not matter whether the application is provided by Vendor A or Vendor B or is a development of an in-house application. It is the contract that is "Once for Wales" and not the application.

8.1.4 National interoperability requirements

A contract for an EU system should include the standardised recording of EU attendances with coded outcomes together with problems, diagnoses and interventions coded using SNOMED-CT.

All attendances should include contextual information such as the:

- health professionals involved in that attendance must be included using a standard reference such as their domain username or name and registration authority reference number (e.g. General Medical Council or Nursing and Midwifery Council number).

- organisation and site to which the patient attended using standardised codes.

- data to support national reporting requirements including date and time of registration, of first and subsequent assessments together with final disposition.

An attendance in the EU should at a minimum generate a clinical document with these clinical and non-clinical data encoded in a structured data together with a human readable view. In conclusion, it is therefore clear that the design and procurement of an EU system must begin with understanding the requirements, a focus on data and standards and a contract of interoperability.

8.1.5 Local integration requirements

No healthcare application or service can be deployed in isolation but instead must be integrated with any existing national and organisational architecture. Interoperability and integration is required at two levels: nationally and within the organisation. Such considerations are important when one considers how to implement an EU application and the amount of work to achieve integration at these two levels may well be considerably different.

If the requirements for local integration are less than for national interoperability, then from an engineering perspective, it will be easier to procure a new application that already supports the national interoperability framework and spend time working on local integration.

Conversely, if the local integration is simpler, or indeed, already in place, then it may be simpler to focus on improving interoperability with the national architecture.

The only other consideration is the functionality of the user-facing application to ensure that it meets the needs of end-users.

8.1.6 A single specification vs. a single product

My final comments on the standardisation of EU applications within Wales is to emphasise that a "Once for Wales" strategy should therefore focus on a single specification and contract for Wales rather than a focus on a product. The specifications and contracts define how any such application should interoperate with the national platform and should be "Once for Wales". Designating that a single

application from a single vendor should be used across Wales is not appropriate, risks innovation, is wasteful of limited resources and takes no account of existing applications and infrastructure.

8.2 Diagnoses and problem lists

An understanding of the diagnoses and problems which a patient faces is a critical informatics problem. Indeed, once such data are available in a routine and systematic fashion, it is possible to implement highly innovative informatics solutions such as automatic monitoring of weight in patients with motor neurone disease or showing a patient with multiple sclerosis how they are progressing compared to a real-time aggregated plot of their peers (Figure 10).

One approach is to use SNOMED CT as a terminology bound to an information model, such as that provided by openEHR. Indeed, information systems such as those used in primary care by general practitioners will typically have a diagnosis and problem list which can be edited by all clinical staff in that practice and these data form a valuable role in identifying cohorts of patients with specific diseases. In general practice, the diagnoses and problems are coded as Read codes and they will typically use a proprietary information model to link the diagnosis with a date.

Some may suggest that a similar framework could be used in hospital practice with healthcare professionals from different health professions, clinical specialties, services and organisations editing a master list of problems and diagnoses. Advocates of this approach suggest that this would be simple to implement as part of the existing Welsh Clinical Portal application and audit trail functionality could be used to track changes in diagnoses over time.

However, I strongly advocate a different approach. Diagnoses and problems are not as conceptually simple as they might first appear. As in Section 5.11, diagnoses and problems are inherently context-specific and there is no single 'truth'. Diagnosis changes over time and there may be significant diagnostic uncertainty over many years for many complex patients.

As such, although the openEHR archetype for diagnosis is detailed and includes metadata about certainty and status, that archetype is insufficient. A wider

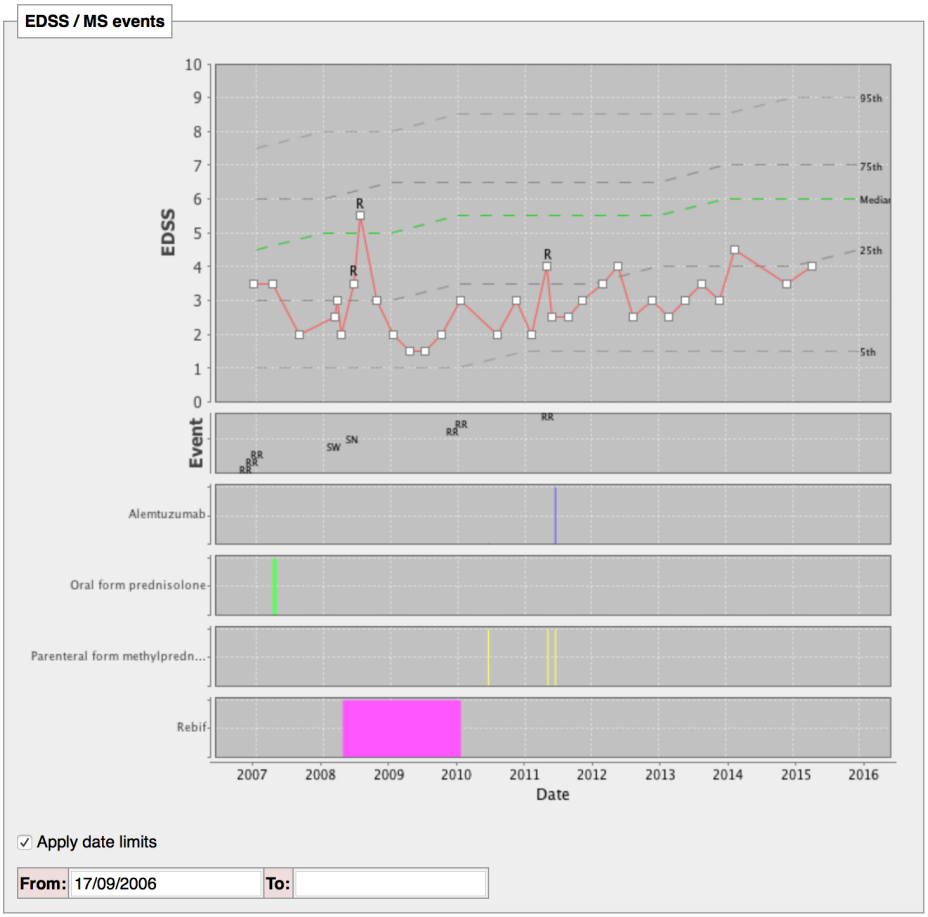

Figure 10: A chart from the PatientCare electronic patient record application showing real-time progress of a patient with multiple sclerosis compared to over 2500 other patients with multiple sclerosis. The dotted lines show the centiles and median progress of the cohort.

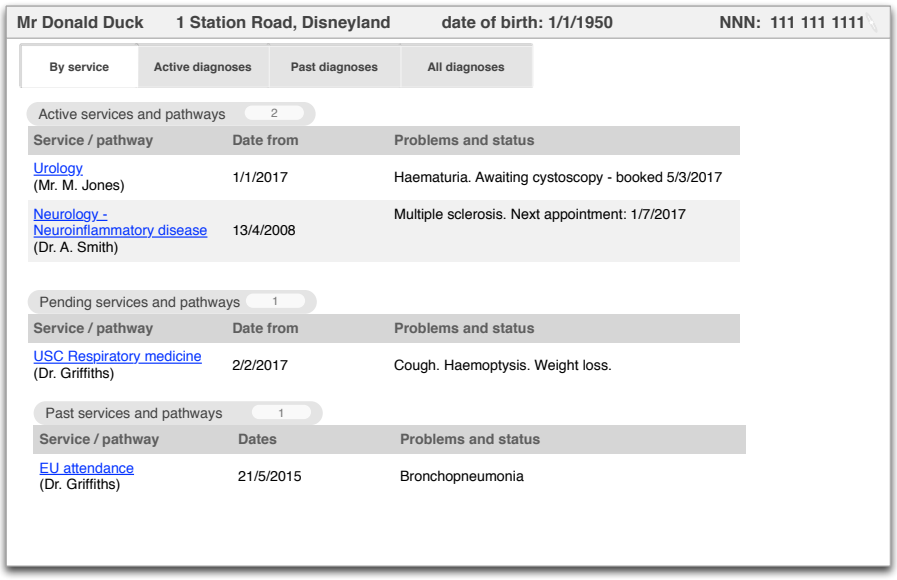

Figure 11: An example mock-up of a diagnostic problem list organised by membership of clinical services and clinical pathways.

information model must be used in order to track patients to clinical services and clinical pathways and it is only in the context of such a pathway, should a diagnosis or problem list be considered.

For example, a patient may be an active member of a range of clinical services and pathways, such as a lung cancer pathway, or a haematuria pathway. A patient may have a range of long-term health conditions such as diabetes mellitus, hypertension or multiple sclerosis. Such diagnoses may be linked by virtue of their general practitioner problem list or membership of a long-term health condition service such as services that care for patients with multiple sclerosis.

The model should therefore bind diagnoses and problems to these contexts so that users can immediately see the contextual information that relate to those diagnoses. An example of how this might look in an information system is shown in Figure 11. As I wrote in Section 5.11, a diagnosis of angina made as part of a chest pain pathway after coronary angiography showed diffuse three vessel disease is different to that made by a general practitioner after the patient reports chest pain. They may have the same SNOMED CT concept but the contextual information around that concept is essential. Similarly, a diagnosis of multiple

sclerosis recorded by the general practitioner in 1980 but with no corresponding membership of a neuro-inflammatory clinical service currently or in the past must be regarded different to a patient under the active follow-up and investigation by an appropriate team.

Such a model does not preclude generating a simple list of diagnoses and problems, but that view of an underlying more complex model is simply that: a convenient view. Users must be able to drill-down to understand the context of any diagnosis and problem list.

Pathways and service registration are critical contextual clues for all clinical data, not simply diagnostic and problem lists. A document created as part of a pathway should itself be linked at the data level so that, for a given service, all correspondence relating to that service can be retrieved easily. Similarly, procedures performed as part of that service or pathway should be recorded and linked in the same way.

Such a model requires a national pathway identifier that can be used across different organisations in order to track care. A pathway may begin in one organisation but continue in another. Such a process does not need the same software running patient administrative systems in Wales, which is how "Once for Wales" is currently framed, but instead, requires administrative systems in each organisation to interoperate with a national platform in order to support cross-organisational working; this should be the true intention of a "Once for Wales" strategy.

In a model in which diagnoses and problems are linked to services and pathways, how does one deal with changing diagnoses? When a patient is admitted with chest pain to the emergency department, a new pathway is created and as part of that pathway, an initial assessment can bind a concept of chest pain to that pathway. Co-morbidities can be copied in to that pathway from the canonical data model representing all contexts and the active and inactive diagnoses linked to them. A health professional is able to choose to incorporate those diagnostic entities into that context, such as an admission or discharge document and update the status of those entities. As such, there is no one source of truth, but merely multiple overlapping opinions as to what is 'truth' at any time-point. During the admission with chest pain, when a prior label of 'angina' is updated to reflect a

new working diagnosis such as that the patient's chest pain is due to gastrooesophageal reflux disease, the diagnosis of angina can be refuted or repudiated with such changes modelled within that pathway context.

This may seem to be a much more complex and messy version of a diagnostic or problem list, but it maps closely to that of reality. As in Section 3.5.1, a model must reflect real-life data and process, and mismatches should be avoided. In real-life, the changing and fleeting diagnostic concepts created, refuted and repudiated as part of the process of clinical care over time are reviewed by general practitioners who update their master list of diagnoses. The final solution therefore, must incorporate a holistic cross-specialty diagnostic list sourced from primary care systems curated, managed and the responsibility of that patient's general practitioner. This list however, complements the real-life model of diagnoses, bound to clinical pathways within hospital-based services.

8.3 An all-Wales national growth chart

8.3.1 Background

Health professionals looking after children frequently need to monitor growth by the systematic recording of weight and height over time. Traditionally, paediatric medical notes will contain a paper growth chart containing normative centiles reflecting the growth of patients with similar baselines characteristics. Unfortunately, paper records are unsuitable unless care is provided only in a single centre by a single team. Other challenges include the loss or misfiling of charts and with any paper record it can be difficult to know who recorded each data point.

In Aneurin Bevan Health Board (ABHB), they created an electronic growth chart solution in 2004 in which health professionals can securely record weight and height and plot those results on an electronic growth chart. They now have data on almost 20,000 patients with a total of almost 60,000 data points. It is a system that has added value to those that use it.

However, while the solution provides important functionality, it is tightly coupled to the ABHB infrastructure - a portal called Clinical Workstation and an underlying product from a commercial company called CCube solutions. As such, it is difficult to take this innovation and deploy it elsewhere without also migrating

the necessary dependencies. It is, in effect, an application containing elements of user interface, logic and process, and an underlying data storage model.

8.3.2 Using the growth chart across Wales

How can the the innovative functionality already developed be deployed across Wales? One option would be to deploy the electronic growth chart as a national product and link to it from other products when health professionals looking after children need to record a weight and height. However, this would be a mistake, essentially focusing on products and not an information model. It would result in linking applications at a user-interface level but not at a semantic data level.

Weight and height recording are generic requirements, not simply useful in paediatric practice and it would be a mistake to spend valuable resource on products that meet the requirements of only a single department or specialty without considering how such functionality might benefit other users across NHS Wales.

Instead, as I have opined in Sections 3, 4 and 5, the national 'platform' should support the routine, systematic storage of structured clinical information using a robust information model. OpenEHR archetypes for height and weight measurements are available as part an OBSERVATION class.[21] A platform that links patient demographics with these archetypes together with an understanding of clinical context (was it measured in clinic or at home, which clinical service, which clinical pathway, which user performed the measurement, what was their job title?) can safely act as a repository for such structured information.

Importantly, simply adopting an OpenEHR archetype does not necessarily result in automatic interoperability. Critically, the contextual information relating to that structured data needs to be standardised (see Section 5.16). If the ABHB solution uses the same user directory than stipulated by the national platform, [22] then different software have the same meaning for a specific user so that "John Smith" in one system is the same "John Smith" in another because they share the same user record by virtue of a unique username and critically, that differentiates

[21] See the OpenEHR Clinical Knowledge Manager - `http://www.openehr.org/ckm/`

[22] This is 'NADEX' in Wales, an Active Directory of all users in Wales

that “John Smith” from the other “John Smiths”. Similarly, data relating to organisations, hospital sites, surgeries and services must be standardised across the enterprise in order to permit full interoperability between systems.

Using this information model, there are then two approaches to integrating the ABHB solution and making its innovative functionality available to other centres. If the ABHB product is already established and, as built, is difficult to extricate from its existing data model, then horizontal integration at the data level may be appropriate. This would be logically called a “legacy integration approach” in which the ABHB product continues to store data in its proprietary data store, but makes its data available to the national platform either by exposing a service from which to “pull” results or adopts a “push” approach when new data are added. Such an approach suits legacy systems which are themselves vertically integrating binding together multiple layers in one application. Such a solution could make the underlying growth measurements available as OpenEHR archetypes or as an FHIR service providing a limited OBSERVATION type of resource made available via a REST-based web service.

However, if the product is itself built with a clear separation between user interface elements and the data model, then it may be possible to adopt a national platform for the storage of data and simply innovate at the workflow, process and user interface levels, leveraging the core standards within the platform to store and retrieve weight and height data for an individual patient.

Indeed, the latter approach could result in the growth chart being more widely used by different specialties. For instance, a relatively small and simple project could be undertaken to take data from the national platform and display the data on a range of growth charts, just as in ABHB. Indeed, as the national platform has provided the necessary infrastructure required, actually plotting the data and comparing to normative data sets would be straightforward. However, once tested and deployed in one organisation, it would be straightforward to apply this innovation to other organisations and other services.

8.3.3 Designing for innovation

In addition, this approach provides future opportunities for innovation. A project to create a "growth chart solution" will do just that, create only a piece of software to create growth charts. Any additional functionality requires change in that proprietary product and therefore potentially limits its more widespread adoption and the development of more innovative solutions using the same data.

Instead, why shouldn't child health surveillance occur in a semi-automated fashion, with logic and algorithms assessing children's growth velocities to send clinical staff alerts when growth is tailing off? It is an important component of clinical information system design to consider the future vision for healthcare information technology and how health data can be integrated logically to allow real-time and grouped alerts. If we expect all children with chronic health conditions to have their growth plotted at a certain interval, then small, relatively easy to write software can be deployed to combine growth chart data with records of chronic diseases to automatically monitor the frequency of observations and send alerts when growth is less than expected.

In conclusion, a focus on a product-based design limits future scope for innovation and risks vendor lock-in. Health data must be made more widely available at an infrastructure level.

8.4 Prescriptions

An example of software to handle prescriptions is the NHS Wales' 'medicines transcription and electronic discharge' (MTeD) system. In it, the Welsh Clinical Portal software receives an 'Admit/Discharge/Transfer' (ADT) message from the underlying Patient Administrative System (PAS) informing it of a patient admission. On receiving this message, a draft discharge document is created which allows clinical staff to start populating the discharge advice letter (DAL) throughout the admission. This appears as a draft document in the document list for that patient. At the same time, in the absence of inpatient ePrescribing within NHS Wales, a list of medications is transcribed, either on admission or more usually by pharmacy staff during the admission. This is kept synchronised with changes in the inpatient medication chart as much as possible. When a discharge is planned,

a two-step process is required to sign off the DAL and to sign off the final list of medications. Sign off of medications originally required signing by a member of the medical team and then final check by a pharmacist, but this restricted use out-of-hours and so the software was changed to support a DAL being sent without a final pharmacy check when used out-of-hours.

There are several important realised and potential benefits to this system:

- Paper discharge letters were incomplete, frequently illegible and needed to be sent by post. Electronic discharge letters can be sent immediately and can form part of an electronic record for patients.
- Work on completing discharge letters could begin on admission and potentially continue through the inpatient stay.

However, as designed, there are several major disadvantages as well:

- The solution is tied into the low-level messaging architecture forcing a document to be created at the time of admission and sent at the time of discharge from the underlying PAS. We now have many patients with alerts showing 'DAL incomplete' even when the patient has been discharged but with no mechanism for completing the document. This process does not match what happens in real-life and therefore is an important model mismatch (see Section 3.5.1).
- The medication list sign-off and its incorporation into a discharge letter happens when a discharge notification is received from PAS. This is another model mismatch between the process in software and what happens in real-life on a ward.
- If the DAL is loaded, users cannot edit the medication list within the DAL, but have to switch back to a different screen to edit the medication list.
- There is no way alternative mechanisms for creating a DAL can be used. For many clinical situations, such as day case surgery, a DAL could be completed automatically based on workflow and process models within the day unit, such as the type of procedure performed and whether there were any

complications. The current system is a closed system and any adaptations would need further development time.

There are also some serious usability concerns with a clunky user-interface and performance problems.

How could this design be improved? One of the most difficult decisions in clinical information system design is how to limit the scope of a particularly system dividing up responsibility for different tasks to logically-organised systems.

Fundamentally, what are we trying to achieve? The key issue is the ability to send a timely, accurate and legible DAL at the time of discharge. When this process is performed using paper-based solutions, a doctor will typically complete a handwritten form, the ward staff send it to pharmacy which checks and issues the prescription, it returns to the ward and either taken by the patient or posted to the general practitioner. A carbon-copy is filed in the medical notes. If a patient is discharged when pharmacy staff are not available, sometimes the prescription is picked up on the next working day or in emergency settings, some wards or units will dispense medications from their stores.

When one considers that process, it is easy to see how the team ended up creating the system as designed. However, the creation of a DAL is fundamentally part of the workflow of the unit and so I would consider dividing up the solution into the following components:

1. A patient-level document store which allows clinical documents to be stored linked to clinical context including admission episodes and national pathway identifiers. Such a store needs to mandate an appropriate information model so that documents of type 'DAL' can include diagnostic, intervention and medication information in a structured format encoded in SNOMED-CT. The medication information should be able to store not only a list of medications at the time of discharge but other critical information such as which medications have been stopped, when a certain medication should be stopped etc. In some cases, it will be appropriate to support a DAL which semantically encodes "no medication changes" for use in certain clinical environments.

2. A workflow system that allows a generic DAL to be created linked to the admission episode and national pathway identifiers.

3. In the absence of a national ePrescribing solution, a workflow solution to allow the recording of admission medications and subsequent changes to those medications. This list is conceptually linked to the inpatient stay and is kept loosely in sync with the paper prescription by pharmacy staff. This becomes the "current medication list" and should a national ePrescribing system be developed, then logically, that service could provide that information.

As such, in certain clinical settings, depending on the workflow and process required, at the time of discharge, a doctor starts or completes a DAL, which automatically brings in a copy of the "current medication list" from the interim or final ePrescribing software, allowing the doctor to sign it off. At in the real-world, the draft DAL does not include the medication list. If a DAL is complete and a list of medications has been pulled from the "current medication list" service. If changes are needed, then those changes can be made as part of the DAL and those changes are pushed back to the "current medication list" service. Depending on need, that DAL can then be checked by pharmacy staff or medical staff can explicitly choose whether to send without a pharmacy check. Once the patient is discharged, the DAL can be sent electronically and pushed to the document store. If the patient has already been discharged, then the document can be sent immediately therefore allowing the completion of a DAL shortly after the patient has been discharged.

In other settings, a DAL may be created dynamically from a different workflow, potentially making use of the services used by the more generic workflow such as handling the lists of medication, but automatically completing the diagnostic and procedural information.

Importantly, the creation and sending of the DAL is not necessarily bound to the underlying ADT messages signifying a patient is admitted or discharged, but instead linked via a workflow system that allows flexibility in approach depending on the requirements. As the creation, signing and sending of a DAL is inextricably

linked to the clinical process, those pieces of functionality should form part of that workflow and not be isolated from it causing information model mismatch.

For this, team and ward-based workflows will need to conceptualise the admission and discharge of patients and handle the management of a DAL for each patient. As such, in those contexts, users must be able to see a list of their patients, the status of a DAL if it exists and an ability to create, edit and sign-off that document. In addition, those lists should include lists of discharged patients without a discharge advice letter, acting as an important nudge to clinical users to complete these vital documents. Aggregated data on DAL completion and the timeliness of correspondence must be made available in order to appropriately manage this component of our clinical services.