Rethinking the Electronic Healthcare Record

Why the EHR failed so hard, and how it should be redesigned to support doctors and nurses effectively in their work

J. Martin Wehlou MD, CISSP

Man In the Middle Books

MAN IN THE MIDDLE BOOKS

First published in book form June 2014.
Cover design by Gilles Vandenoostende.
Printed edition ISBN 978-91-981706-0-3.

Composed in Scrivener®
Formatted for LaTeX
by MultiMarkdown

Dedicated to Danielle, Charline, Milena, Julian, and Hania. You made this worth doing.

Martin Wehlou 2014

Contents

List of Figures

List of Tables

Table 0.1: Draft versions

Version	Date
1	January 23, 2014
2	February 2, 2014
3	February 6, 2014
4	February 12, 2014
5	February 16, 2014
6	February 24, 2014
7	March 23, 2014
8	March 24, 2014
9	March 25, 2014
10	April 1, 2014
11	April 8, 2014
12	April 15, 2014
13	April 22, 2014
14	April 29, 2014
15	May 6, 2014
16	May 13, 2014
17	May 20, 2014
18	May 27, 2014
19	June 3, 2014
20	June 10, 2014
21	June 17, 2014
22	June 24, 2014

Download links

From version 9 and onwards, I'll post preview versions every Tuesday to the following URL:

`http://wehlou.com/ehrbook/draft.pdf`

Older versions will be preserved as:

`http://wehlou.com/ehrbook/draft_X.pdf`

... where "X" is a one or two digit number. For instance:

`http://wehlou.com/ehrbook/draft_9.pdf`

and

`http://wehlou.com/ehrbook/draft_10.pdf`

... and so on. The first one in the series is number 9.

For a while, I also published diff files, making it easier to see the difference between the current and the previous version, but I had to stop doing this as Adobe Pro gave me so much grief. The diff function really isn't up to snuff, and I know of no other software that can do this with pdf files.

Contributing comments

You'll make me absolutely ecstatic if you contribute with comments of all kinds, such as errata, arguments for or against what I'm saying, or suggestions for expansion or other reading.

You can do that by any means that suits you, I'll find a way to use it. So go ahead and write me emails, send me scribbled notes on birch tree bark, a mix tape, video confessions, or whatever, but one way that's easy both for you and for me is annotated PDF files. You can annotate my PDF using any of these suggested tools:

- Acrobat Pro.
- Preview on OSX (annotation bar).
- GoodReader for iOS.

... or any number of other tools, some of which must exist for Windows. If you want, I could prepare a special "for comment" version PDF that you can annotate with just Acrobat Reader. Let me know by email, so I know it's worth doing.

You can send the annotated file back to me at:
martin@wehlou.com

If you *don't* want a mention in my "Acknowledgements", please say so. Else I'll include you.

Thanks!

In a nutshell

If you're not sure if you want to invest the time or effort to read the whole book, this summary is for you.

Medicine today is far from fulfilling its potential, and the reasons are numerous. One of the most important reasons is that the documentation system used today is fundamentally identical to the system used half a century ago, and it was known to be defective even then. Nothing much has been done to improve on it. The chance to improve it during the transition from paper records to electronic records was squandered.

There are two steps in the handling of every patient case, the diagnostic and the treatment steps. Both are very problematic and very much lacking in completeness, rigor, and consistency.

The first step, arriving at the right diagnosis, is usually a very haphazard process, relying on first impressions, intuition, and experience of the doctor. The potential solutions to the diagnostic problems have been thoroughly described by L.L. Weed [5] [6] , and this book incorporates many of his findings and points, and adds to them in significant ways.

The second step, executing the right therapies and followup, has until now not been systematically analyzed and converted to a form amenable to automation and direct use by doctors. This analysis and how it should be implemented form the major part of this book.

Medicine needs both these parts to be built, but they have to be built in such a way that they seamlessly integrate with each other and form a consistent whole. In what follows, I will show not only that these two steps can be integrated, but that they enrich one another, and that together they form a whole that can be gradually introduced into a working healthcare system, while also being scalable. All the science and knowledge we've acquired means very little unless we have tools to put that knowledge into practice, in a safe and consistent manner.

Doctors today are expected to keep the huge and ever changing knowledge base of medicine in their heads, performing both the complicated diagnostic and therapeutic procedures without any substantial knowledge-support systems. The only automation with any significant evolution is the

documentation part, which, in comparison, has a relatively limited impact on the quality of diagnosis and care.

First, we need to make the initial data collection about the patient much more complete. This can best be done by using software to guide the gathering of symptoms and clinical signs.

Then we need to use the initial set of clinical data to make a complete list of possible diagnoses (or, as named in this book, "issues"). This candidate list of issues further informs us of clinical signs and tests that should be performed. This interaction between clinical findings and candidate issues continues until a small set of likely issues have been identified.

Each issue comes with a template that not only describes the diagnostic criteria for the issue, and thus informs the issue selection process, but also contains all the information needed for further investigations and therapies. This relieves the doctor from the hard and error-prone task of memorizing every possible therapy and its details.

The interaction of different issues in the same patient also becomes amenable to automated discovery and handling. All these features result in a change of the role of the doctor, from an unreliable source of memorized facts, to someone who, together with the patient, can judge and select between all the available diagnostic and therapeutic tools. The end result is a much higher consistency of care, and a reduction in errors of commission and omission.

A better document hierarchy that relates findings and conclusions to each other in the same way that the doctor actually reasons and decides, makes the documentation side of the EHR more amenable to understanding, and allows auditing of the record. At the same time, this structure solves several problems related to confidentiality and distribution of consistent parts of the medical record.

Introduction

These notes started out as a brief outline for my yearly lecture at the Karolinska Institute in Stockholm, to the students of the "International Masters Program in Health Informatics". As time went by, I began to suspect that these notes harbored in them something bigger, maybe even a book. After some procrastination, I finally started expanding on these notes in January 2014, and what you see here is the result.

The target audience for the original notes consisted of students well-versed in both medicine and technology, which explains why the notes tend to slide from one area into the other without much of a transition.

Converting these relatively limited notes to a full book implies that the target audience also changes in composition and character. Except for the aforementioned students, I include doctors, nurses, and developers into the intended audience. I think it absolutely necessary that developers learn to appreciate the nuances of doctors' use of the medical record, while I also think it essential that doctors learn to understand the technical limitations and possibilities inherent in these systems. Real solutions will not come from two or more professional groups working together, but will only come from each professional group reaching into the other's knowledge area and grabbing onto the stuff that they actually need. The design coherence needed for fully useful system designs must sprout from minds that can bridge the gap, and if these are in short supply, we must either produce more of them, utilize them better, or both.

At the beginning of each chapter, I've added a little graphical indicator of how technically difficult I think the chapter is. I see this indicator mainly as a warning to doctors not to feel too discouraged if the contents seem impregnable.

Another very important point is this: don't look back! We have built IT systems based on how paper records work, and that didn't turn out well. We also have to stop looking at current systems for inspiration on how to build the next generation system, else they'll also be dismal failures[1]. Forget about the past. Think up new systems from first principles, and

[1] Yes, I take that as a given.

it wouldn't even hurt if you assumed that every solution that looks most like what we have today is a bad idea and should be scrapped. The more different from that, the better.

There are also a number of evolutions in the development of the EHR that worry me. The structure of the medical records in their *current* form is often done to "make the computer understand" the healthcare process. If we think about the interaction of man and machine in healthcare, there are three possible ways to go:

- Man does his job and then feeds the machine data for safekeeping and management analysis purposes. This is what we have today for the main part.
- Man feeds the machine sufficiently understandable data, so that the machine can take the responsibility for the intellectual work. This is the road a number of systems, including OpenEHR, seem to take, except I find there is no basis for thinking the machine can take over that job just yet. In the future, yes, but today?
- Man feeds the machine the minimum of data it needs to locate and gather the resources *man* needs to take responsibility for the intellectual work. This way we would improve man's ability to work correctly, while automating away man's main weaknesses, namely memorizing massive amounts of data, and consistent attention to detail. This is the way I think we must progress for the foreseeable future.

One point I need to make clear: I have a lot of opinions about IT in healthcare, but I've also spent a lifetime almost evenly divided between the two fields forming the basis for those opinions. I don't expect it all to line up with other people's thinking on the subject, in particular if they have a background in only one of the two fields.

I will not limit myself to what I can prove, since that would make the text far too short. Far too few of the important angles have been the subject of objective research, so there isn't much to put in the bibliography. It seems much of the field is built upon preconceived notions and assumptions by all involved, developers and medical staff alike, with mostly everyone assuming somebody else has it all figured out.

My background, in brief, is in mathematics and medicine, including a residency in general, vascular, and thoracic surgery, and specialization in general medicine both in Belgium and Sweden. I've also developed software for more than 30 years in several domains, including medical applications. I'm still working part time in general practice and with clinical studies, while also writing software. Appendix B consists of an extended description of my background.

The book is organized in a couple of parts:

Part I

The first part covers the major "business cases" for creating EHR systems. It also briefly analyzes what type of knowledge must be instilled into an individual to make her into a physician. A very quick recap of the history of medical records is then followed by a summary of the major ways the current EHR systems fail us.

Part II

I go into how current EHR systems work, and why they work as they do. I can't help but complain about almost everything about these systems.

Part III

I go into how doctors work clinically, which real requirements we can derive from that, and how a correctly designed EHR system built on these requirements should look. I'm making the assumption that the main goal of these systems should be to support clinical work, even though that is clearly not the case today. But it must be in the future.

Appendices

The last part contains one appendix that goes into more technical detail on the document tree design. Unless you are into building systems, or comparing the designs in this book to other designs found elsewhere, you could probably safely skip this.

There is a second appendix with a slightly longer history of the author's life story, at least as far as it applies to IT in medicine. This story does help explain at least some of the design choices made in the book.

In a text like this, one has to try to be clear with terminology. It is often necessary to reduce the number of terms, even at the cost of losing some nuance, just to avoid introducing ambiguity in the text. Here follows a few selected terms I've raised to the level of "housebroken" and have used as consistently as I'm able to.

Architect, designer, and so on

In this book, if not always in real life, we often use the following names of roles of software builders.

Requirements Engineer

The requirements engineer takes the wishes and demands from the stakeholders (users, buyers, owners) and converts them into a list of requirements that can be used by the system architect, the designer, and the coder to create a system fulfilling the wishes and hopes of the user and buyer[2].

System Architect

The system architect, or architect for short, is the person on the development team that takes the requirements and decides on the high-level structure of the system, such as what different software subsystems it should consist of, which types of interconnections should be done and between what, and on which platforms it will run on.

Designer

The designer takes the overall structure as defined by the architect and boils it down to modules and interfaces, together with a description of what interfaces and functions should do.

[2]Yes, that's the idea, but in reality the buyer's wishes greatly outweigh the user's wishes. Money speaks.

Coder

The coder takes the designs from the designer and creates code. His output should be the executable deliverable.

Developer

In this text, the term "developer" is used to mean anyone directly involved in the production of the software, which makes it applicable to the coder, the designer, and the architect. It's useful to regard the requirements engineer as a developer as well, in general. Whenever it's not necessary to define exactly which type of software person referred to, the term "developer" will be used.

EHR

Electronic healthcare records (EHR) are often called "Electronic Medical Records" (EMR), and there seems to be no useful distinction between those two terms, but there are arguments to the contrary[3]. If there is a difference then the "EMR" is more akin to the old paper records, while an "EHR" reaches beyond that to include both other organizations and other tools.

"EMR" occasionally means "Emergency Medical Responder", which can confuse things. It can also mean "Explosive Mishap Report", confusing things even further.

"EHR", on the other hand, has a much shorter list of interpretations. It does include "Explosive Hazards Reduction", which we can only see as a positive.

In this text, only the term "EHR" will be used, to the exclusion of "EMR" for the above reasons.

Issue

We need to have a name for the reason we see patients. We can't call it a "disease", since seeing a one-year-old for a regular growth check, isn't a "disease". We can't call it a "problem", since that insults the women we see for pregnancy follow-up. I've chosen to call these "issues", or sometimes "healthcare issues" for extra emphasis.

[3] http://www.healthit.gov/buzz-blog/electronic-health-and-medical-records/emr-vs-ehr-difference/

Issue Template

The management of a healthcare issue consists of a series of guidelines, references to publications, addresses we can refer to, a range of diagnostic techniques and therapeutic actions, advisories, medication types, products, and dosages, equations, resource scoring, and more. This entire set of information and tools needed in the management of a healthcare issue is collected and organized into an "issue template". This term implies the technical implementation of the management of a healthcare issue, but here the term is used in both senses, as the collection of tools for management of an issue, *and* as the technical implementation of such a collection of tools.

Item

The term "item" is used in two senses. In the context of "issues", an "item" is a one-liner, a certain clinical finding with it's set of possible values. It's the smallest part of a guideline, and therefore the constituent part for issue templates. For instance, "Blood pressure" can be an item, just as "Cardiac sounds" can be one.

In many other instances, "item" is used to mean what most people mean with "item". Such as "I have two items in a basket", or "are those two an item"?

His and hers

When writing a text, we always have the problem of what gender to use in third person. It's all too easy to write "him" all the time, and be revealed for the crypto chauvinists most of us older male doctors are deep down inside. But going to the other extreme, using "her" all the time, would lend a certain creepiness to the text. You can't go writing "him or her" everywhere either, since it simply looks ridiculous in the long run, and breaks the rhythm of the text, if it ever had one.

My chosen strategy is to mix it up a bit. In situations where I have to choose, I'll let the doctor be female, at least in situations where the doctor comes out looking smart. When the doctor is described as confused, or easily distracted, or of somewhat limited ability, I'll often use "him". Patients will usually be represented as males, too.

Yes, I realize this is also a prejudiced way of thinking and writing, but I'm less afraid of men than of women.

Acknowledgements

In no particular order, I'd like to thank the following people for constructive criticism and corrections of the text: Kim Nevelsteen, Peter Olsson, Göran Agerberg, Johan Månflod, Jack Holleran, Ingrid Eckerman, Karin Lindhagen, and Anders Westermark.

I'd like to thank Mary Brown of Capella University for an insightful perspective on the US medical informatics scene.

Chris Bunch and Jeremy Dwight were kind enough to allow me to use the CHF guideline as an example, and both contributed important viewpoints on the creation and management of guidelines in general.

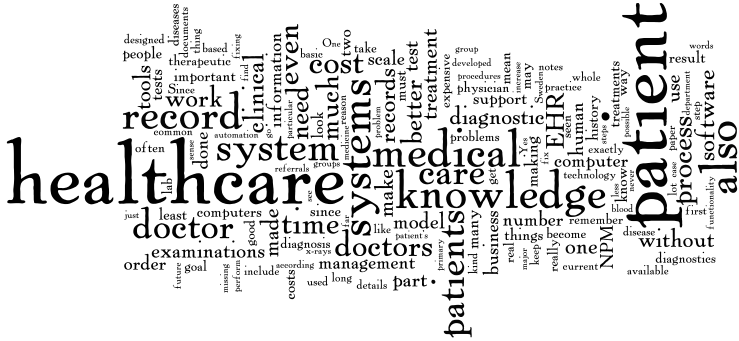
Bengt Dahlin contributed with his knowledge on the history of the medical record.

I'd also like to thank Jerker Green for many inspiring discussions on this theme.

And last, but not least, Hania Uscka-Wehlou, my wife, for inspiration, discussions, corrections, both linguistic and contentwise, LaTeX-assistance, and providing the encouragement to do this in the first place. And a lot of other things I'm not going to discuss here.

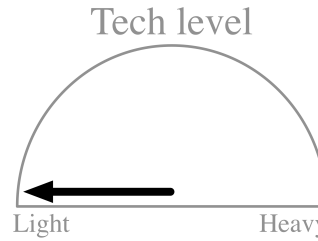
Part I

The basics



Summary

Before going into what's wrong with current systems, or how to build the next generation, we have to look over the basics of medical record keeping. These basics include identifying the driving business models, the stakeholders, the motivations of the stakeholders, and how doctors are trained and kept up to date. It must also include the history of the medical records, and an outline of the relationship between the medical record and the professional, in other words, how does the medical record actually help the medical professional do a better job?



Chapter 1

The business model

What are the economic incentives for doing the right thing? What is the "right" thing?

There's a business model behind every decision on how to structure healthcare and its support services. That business model could be based on the overall cost to society of healthcare issues, versus the cost of providing relief and prevention. These are the concerns at a large scale, by region or nation. On a smaller scale, the business model could include measures intended to reduce or displace costs, such as moving them to other actors, and measures to optimize handling of cases.

Let's take a few examples to illustrate the difference between the large-scale business model and the small-scale business model.

1.1 Large-scale business model

If we calculate the cost to society of a case of polio, including treatment cost, assistance cost, and loss of productivity, we find that it's much more expensive than the alternate cost of prevention[1]. It's obvious to anyone, except maybe to the anti-vaccination nut cases, that global vaccination and ultimate eradication of the virus is an amazingly good idea.

The same calculation has also been made for certain types of cancer where we have useful screening methods, such as for breast cancer, and colon cancer, and early detection and treatment is the winning proposition,

[1] http://www.polioeradication.org/Portals/0/Document/Resources/StrategyWork/EconomicCase.pdf

even without considering the human cost of contracting cancer and having it detected too late for curative treatment.

We can also show that the correct and effective treatment of joint disease, diabetes, hypertension, vascular disease, and a host of other problems, is a generally good idea from a national economics standpoint. All these things cost less to society as a whole if treated according to the state of the art. Even bleeding edge, and often extremely expensive, treatments are economically defensible once we take the evolution of the treatment into account over a period of time. Many of these initially expensive treatments lead to much more affordable and much more applicable treatments in the future, treatments that would never have been developed if the initial expense wasn't made.

In short, paying for the best healthcare you can provide to your population is one hell of a good investment for any nation, even without considering the human cost of disease. No civilized nation would argue otherwise[2].

If healthcare is managed primarily according the benefits to the nation as a whole, the systems developed for healthcare will focus on providing prevention and treatment from a medical perspective. The management of local expenditures will still play a background role, while the calculation of "profit" makes no sense. There's no immediate and local payback for each treated patient; the payback is on a national scale and in a longer perspective.

1.2 Small-scale business model

The small-scale business model comes into play for a hospital, a department, or a practice. It's all the rage in models like New Public Management (NPM), where departments are made into cost centers, and each is incentivized to increase profits and reduce costs, on the assumption that if all departments become more cost conscious and more profitable, the organization, even the nation, as a whole benefits.

This model has been used for a long time in industry, and is becoming increasingly discredited even there, especially since the most profitable information technology companies are abandoning this model. Companies are increasingly seeing the benefit of having all departments work for the common good of the company as a whole, instead of artificially competing with one another. Healthcare seems to be a few decennia behind on this learning curve.

When applying this cost center model to healthcare, at least in Sweden, a fictional cost is assigned to diagnostic and therapeutic actions. Medication prescribed is assigned the retail cost and "charged" to a fictional budget

[2]Except the USA, but you guys are slowly getting the message, too.

the prescribing organization has. Every radiology exam, or lab order, is similarly "charged" to the provider that ordered it. Every visit from a patient ends up on the "profit" side. The idea is that providers should be incentivized to save on costly examinations and treatments, while at the same time seeing as many patients as possible.

In this whole arrangement, two important incentives are missing. First, referrals are "free"; they don't incur a "charge" on the referrer. Secondly, there's no "reward" for actually making patients better. See where this is going?

In order to keep organizations in pretend money[3], the trick is to refer patients to someone else if it looks like their treatment is going to "cost" more than their visits bring in. On the other side of the referral, the referee is incentivized to refuse as many referrals as possible, especially if they look like they'll need a lot of expensive care. Or, alternatively, they're incentivized to only accept referrals where the originator of the referral has done as many expensive diagnostics as possible *before* referring the patient, on the referrer's own "dime". This turns the referring procedure into a confrontational game, instead of the cooperative effort to help and serve patients it should be.

Just to take an example: orthopedic surgeons take a lot of X-rays, since that is one of their most important diagnostic tools. They need them in most, but not all, patient cases. X-rays are therefore a large "cost" item in their budget, and they'd love to get patients referred who already had all the required X-rays taken beforehand to save on cost. In some places, they've taken this to the logical extreme. A department of orthopedics may not even accept a referral without accompanying X-rays. This causes a lot more X-rays than necessary to be made. The result is a hugely increased total cost of healthcare, longer waiting times for other types of X-rays that are really necessary, and a loss for everybody's budget, *except* the department of orthopedics, which comes out looking more efficient.

Other examples are: the emergency department sending patients to primary care for writing prescriptions, cardiologists referring patients back to the referrer with a recommendation for the referrer to prescribe expensive medication the cardiologist prefers but can't "afford", and more of that kind. In each instance, the patient is given a run-around, while the *total* cost of care goes up, and the originator of the problem is rewarded for being budget conscious. It's important to remember that the Swedish healthcare system is firmly "single payer", so all these costs are ultimately covered by the same agency, the state. The whole exercise is futile, idiotic, and expensive.

[3]The departments with most pretend money left and the end of the year, get preferential treatment when it comes to staffing and costs the year after.

If healthcare is provided under a small-scale business model such as NPM, the supporting systems become essential to the process of calculating and controlling costs, and keeping tabs on profitability, *for the department or institution using them*. Any benefits on a national scale or long term will become invisible and ignored. This explains why there is *no business case* for better healthcare under these ideologies, only in local sub-optimizations. Small business scale practices make sense in small businesses, make very little sense in major corporations, and are absolutely toxic to large-scale population concerns such as healthcare.

Finally, we have to note that regardless of budgets, the tax paying citizens may want to pay for healthcare, even if it isn't economically efficient. Human wellbeing consists of more than financially measurable aspects of life. It's absurd to only invest in measures that result in a monetary return on investment. If that was all we were concerned with as humans, we wouldn't procreate.

So, how come patients aren't worse off than they are in Sweden? With a model like the one I just described, one would expect the healthcare to be really crappy, but it isn't. The explanation for this lies in a little bit of compensatory socialism; doctors aren't paid according to the outcome of the budget numbers, or the number of patients. Doctors, in general, receive a fixed monthly salary, making them independent of the NPM game. So even though it's a real hassle having to argue with the orthopedics department, and others, over referrals, doctors have no incentive to save on these fictional charges, usually preferring to get the patient taken care of, regardless.

In short, it seems we're lucky that doctors don't need to care about the NPM numbers and budgets, partly neutralizing the best efforts of public management to corrupt and destroy the healthcare delivery process. One of two things can happen in the future: either the whole NPM idea is scrapped and replaced with a larger scale motivational system that measures and rewards healthcare outcomes, as it should do, or doctors move off the fixed salary regiment and become rewarded according to the NPM based measurements, including small-scale monetary rewards, moving Swedish healthcare to a similar system as the USA, almost certainly making healthcare as inefficient, and unequally distributed as it is there [4].

1.3 The stakeholders

A number of different professional groups have a stake in how the electronic healthcare record system works. Each of their primary uses of the system differs, and the demands on functionality are often in conflict.

Physicians mainly use current systems to record and retrieve patient histories, and to send and receive referrals. Other important uses are: the creation of prescriptions for medication, orders for radiology, orders for other technical diagnostic procedures, retrieval of radiology reports, and retrieval of lab reports. The system is also used to create correspondence, such as letters to patients.

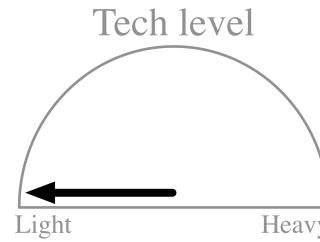
A physician working in her own practice will also use the system to handle billing and payments.

The above reflects how current systems are used by doctors, but if the system was better, it could be used to support in the management of diagnosis and treatment of diseases, too.

Nurses use the system to find out which diagnostics and treatments the physician has ordered, and to take notes on the progression and results.

Administrators mainly use the electronic healthcare record to measure the production as number of patients, severity of caseloads, how many beds are free, and more such. It is in administrators' interests that as much data as possible is coded in a way that allows statistical analysis of the organization and its throughput.

Since there are several stakeholders, all the groups should have their interests represented when designing the systems. As it is today, the interests of the management group not only overshadow the interests of the clinical groups, but is also increasingly displacing them. As a result, healthcare systems turn into management systems where the role of doctors and nurses, as far as the system is concerned, is progressively reduced to that of data input clerks. There's very little, if any, effort to increase the utility of healthcare systems for better diagnostics and treatments.



Chapter **2**

What are doctors made of?

How do we train doctors? How do we keep them trained and sharp after that?

Naturally, we're all made of some calcium, water, neural cells, and not an inconsiderable amount of intestinal content, but what we really should discuss is what kind of knowledge and training is necessary to "build" a functioning physician. How do you go from being just a regular person to a medical doctor able to make decisions affecting the health of a patient?

The knowledge we need can be divided into theoretical knowledge of the healthy human, knowledge about the mechanisms of disease, dexterity in clinical examinations, craftsmanship in diagnostic and therapeutic procedures, and current knowledge about diagnoses and therapies for a range of problems.

2.1 Theory of the healthy human

The initial courses at medical school are all focused on teaching the basic normal functioning of the human body. This includes stuff like biochemistry, anatomy, physiology, and more. As years go by and science advances, this body of knowledge tends to increase rather rapidly. We have long left behind us the time when an individual doctor could more or less have a grasp on all we know of the normal human body, so we have to assume that the practicing physician will only have at his fingertips the most rudimentary facts about the body. Any clinical work that requires a more detailed

knowledge must also be supported by tools that provide the clinician with the knowledge needed.

Except for the occasional old anatomical atlas and well thumbed biochemistry book, you won't find many such tools in most clinician's offices, simply because very few of these tools exist. Even if they did, the clinician usually has just the one computer for the medical record, and those tools either don't work on that system, aren't allowed to be installed by less than helpful IT support staff, or don't work together with the existing medical-record software in any case.

2.2 Mechanisms of disease

Knowledge about mechanisms of disease is rapidly evolving. It's evolving so fast that any course knowledge the doctor may have is quickly outdated. There is a need for constant education about these developments, not on a yearly basis, but on a monthly or even weekly basis. Reading medical journals and participation in medical symposia also helps keep us updated, but it's a very hit-and-miss proposition if what is learned comes to practical use before it's forgotten again.

2.3 Clinical examinations

Clinical examinations include routines such as taking a blood pressure, listening to the chest sounds (and recognizing what you're hearing!), palpating the abdomen, testing reflexes, evaluating joints and tendons, and so on. *How* to do most of these clinical examinations must basically be taught in person, during classes or during work in the clinic with a real patient and a tutor. Some of these examinations can also usefully be taught to an already experienced physician with the aid of diagrams and explanations. An example can be seen in the excellent webpages from the American Academy of Family Physicians (AAFP[1]), about the examinations of the shoulder joint, a pretty complex subject (see figure 2.1).

The illustration looks as if it came out of a jujitsu manual, but actually demonstrates one of the important clinical tests for instability of the shoulder joint. It is described in text, with an image how to perform the test, and the meaning of a positive test (anterior glenohumeral instability) is also mentioned briefly.

Anyone who is not a shoulder specialist, would be hard pressed to remember this test, how it's done, and what it means. We've got hundreds of other clinical tests to remember, and this is one of those that are easily forgotten.

[1] http://aafp.org

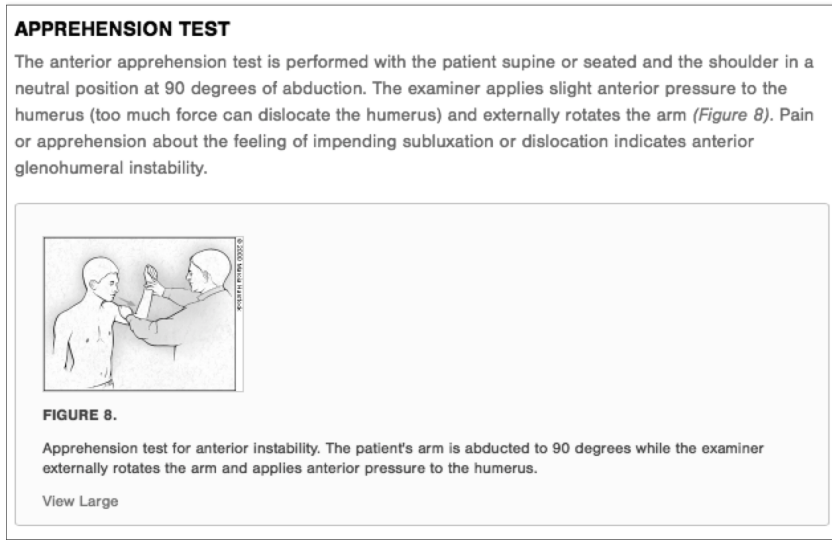


Figure 2.1: A clinical examination, as shown on the AAFP.org site

What we need in practice is a quick reference that pops up when we have a patient with shoulder problems. That reference should include this and other relevant tests, so we know what to do. As it is today, doctors have to know about the existence of the AAFP website, know where to look and go find the test. Not only that, we'll also have to describe the test, at least briefly, in the medical record together with the result. If we don't, a future reader of the patient record will either not know what we're talking about, or will have to look up the test and the meaning anew, and this process will have to be repeated each time the record is read.

Clearly, what we need is a direct and current link between the documentation for how the test is done, with the result of performing the test on the patient, and have all that become a part of the patient's medical record[2].

2.4 Craftsmanship

Craftsmanship comes into play when it's time to draw blood, open the abdomen, hack off limbs, check the eyesight or retina, or aspirate earwax.

[2]Spoiler alert: the solution is an iotaMed-type application, as I describe on page 123 and onward.

Depending on speciality, this craftsmanship can go from the very limited, as for psychiatrists[3], to the very extensive, as for surgeons.

Most, if not all, of this craftsmanship must be learned the same way ironsmiths learn their skill, in a master-apprentice setup. When these skills need updating, doctors again go watch each other do whatever it is that is being done, then try it themselves under initial supervision. There is some help that IT systems could provide for this process, but nothing earth-shaking[4]. This type of learning process probably needs to stay the way it is for the foreseeable future.

2.5 Diagnostic and therapeutic knowledge

Knowledge about diagnostic tools falls partly into the same category as "clinical examinations", but it encompasses much more. Knowing which radiology studies to order for confirmation of exactly which diseases, is a skill that needs constant updating. Knowing which lab tests are available and what they mean, also takes some considerable work to keep up to date. Even knowing which diseases *can* be diagnosed using available tools is far from simple. Worse, even knowing which diseases *exist* is a challenge[5].

The therapeutic arsenal, i.e. the number and kind of methods we have at our disposal to alleviate diseases, is changing at a fantastic pace. There is no way a practicing physician can keep track of even a fraction of what is going on in this area, except for a tiny part of a single speciality. Even knowing which other speciality to refer a patient to, or even that the patient should be referred at all, is complicated enough to be regarded as a significant problem in its own right.

The sum total of all diagnostic and therapeutic knowledge is so vast, and so rapidly changing and expanding, that it's impossible to rely on human memory alone. There's not enough hours in a day for any doctor to even barely keep up. This is where we need knowledge-based tools the most. But to make them even remotely useful, they have to be linked to the medical record in such a way that searches are triggered and enhanced based on the information about the patient in the records, *and* that the results from such searches also becomes part of the patient's information,

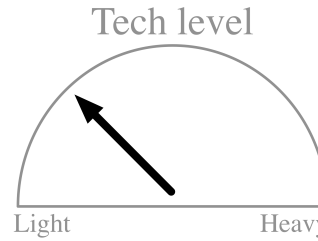
[3]Practically all of a psychiatrist's skill lies in other areas than the purely manual.

[4]There's a lot of hay being made of computer simulations for training surgeons, but it's out of proportion to its relative importance, at least compared to other urgent problems in medicine that need computerization. But there's more money to be made more quickly selling simulators, I guess. Or, just as likely, it's easier for journalists to understand.

[5]Which, by the way, is far fewer than the internet would have you believe.

including the conclusions the doctor drew from the searches in view of the patient's particulars.

Having searches of the universe of knowledge unconnected to the patient record not only increases the cognitive load on the doctor beyond the bearable, but also loses a large part of the advantage of automation, since the relationships found, and decisions made by the doctor cannot be persisted into the record. This implies they can't be used as a base for future analysis and decisions. The waste of human intellectual effort is simply epic.



Chapter 3

Encapsulation

Why do we need specializations? How do we keep them independent enough to evolve?

If we forget about healthcare for a minute, and look at how information is handled in other parts of science and society, we can see that as the information and knowledge volume increases, compartmentalization and delegation compensates for complexity and allows us to keep evolving. As an example that should be easy to visualize, let's take programming. Without going into the entire history of computing, we can claim with confidence that one of the most important steps in the evolution to highly complex systems was the invention of object oriented programming (OOP). OOP is based on the following ideas:

- The details of an implementation should remain invisible to everyone using an object in a larger context, so much so that the internal coding, the implementation, can change as long as the use of it by other objects does not change.
- The interface to the object should be minimal and contain nothing that depends on the exact inner workings of the object.
- At each level of abstraction, the programmer composes objects and creates a new object with a higher level of abstraction.

This way, OOP leads to ever higher levels of abstraction, each level being free from knowledge of details of objects at lower levels of abstraction.

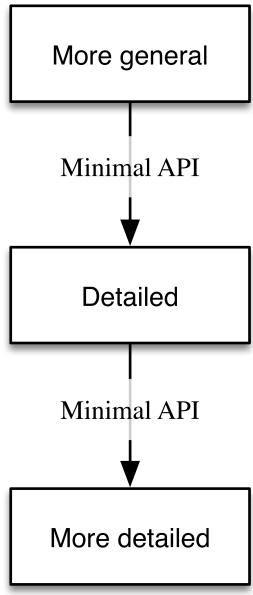


Figure 3.1: Encapsulation flow

Correctly done, OOP removes from view all the internal complexity you don't need at any particular level of complexity. This staged reduction of detail, opaque encapsulation, is what turns a potentially exponential growth of detail into a linear process that can be handled by human minds.

In medicine, the same process applies. The patient talks to the general practitioner (GP) using a high-level API: "I'm sick. I think it's my liver. It runs in the family."

The GP reasons to himself in much more detail: *"The liver. Right... What he probably means is that he's prone to nausea and stomach aches, but that's probably because his whole family is living off hamburgers and get into drunken fist fights over the TV remote. Anyhow, I'll check his transaminases, not forgetting the gamma-GT."* But what he says to the patient after poking his abdomen for a bit is something like: *"Hm... your liver is maybe a little tender, we'll run some tests."*

If the GP had gone to a continued professional education (CPE) class and learned that there is now a virus causing a deadly disease involving symptoms of nausea, fights, and remotes, and that it was brought to earth by the moon landing crew[1], he would still have responded the same way:

[1]Don't worry. I'm lying.

"We'll run some tests". In other words, the change in the GP's internal implementation of how to do medicine in a family practice in the space age does not lead to a change in his information interface towards the patient. The GP is fully encapsulated and OOP compliant.

Let's push this example a bit and assume something is terribly wrong with the lab tests and the GP refers the patient to a gastroenterologist with the general question: *"What's up with this liver? Maybe a biopsy would be a good idea?"*

Now, let's further assume that the gastroenterologist agrees[2], goes on to make an appointment for a biopsy, chooses the most suitable ultrasound transducer, the right needle gauge and length, etc, etc, and does the biopsy. The gastroenterologist, in turn, sends off the specimens to a pathologist for microscopic, and possibly histochemical, analysis.

The pathologist also does his thing coloring, embedding, slicing and dicing, etc, etc, and all this results in an answer from the pathologist to the gastroenterologist detailing the microscopic findings, and elaborating on the type of tissue damage seen. The terms "cirrhosis" and "steatosis" may have been used in this report. A few other minor types of cell damage or proliferation is also noted. Since the pathologist does not report all the different preparation steps or solutions and equipment used, the pathologist is also encapsulated and has a restricted interface versus the gastroenterologist.

The gastroenterologist looks at this report, decides from the clinical picture and the pathology report that the cirrhosis is significant, the steatosis less so, since it's very common in the general population, and that the other findings of minor cell damage are coincidental and irrelevant for the current major complaints. So the gastroenterologist reports only the most significant findings to the GP, and just takes notes of the less significant findings in his own notes, just in case they will turn out to be relevant later.

So, in summary, the gastroenterologist reports to the GP something like this: *"The biopsy showed a moderate degree of cirrhosis with some steatosis"*.

At this point, the GP is supposed to understand *"cirrhosis"* and *"steatosis"* and more or less know what to do about it (cut out the alcohol and the hamburgers; the fist fights present no problem). But the GP does not need to know how to do a liver biopsy or how to prepare the samples for microscopy or even how cirrhosis looks in a microscope. Even if the specialist buys new equipment and then does his biopsies in a different and better way, this makes no difference in the interface between the GP and the specialist. In this example, the gastroenterologist is fully encapsulated versus the GP.

[2]This does happen.

This encapsulation allows every layer of abstraction to evolve independently. The GP can change and improve his methods without the patient noticing[3]. The specialist can change and improve his methods without changing his interface towards the GP, and the pathologist can modify his methods without changing his interface towards the gastroenterologist. This is the only way to allow medicine to evolve, going from the "super GP" who knew all of medicine in the middle ages[4] to the super specialists of today.

It is relatively easy to see that the same process of layered levels of abstraction of knowledge applies in all intellectual human endeavors, not only programming and medicine.

And here comes the moral of this story:

To allow medicine to work efficiently, we must mirror the same levels of abstraction, encapsulation, and separation of concerns in the EHR as the EHR becomes our primary tool. Only high-level information should be shared by default, not the details. If we keep flattening the EHR as is generally done today, with access to every detail at every level, we're moving medicine back into the middle ages instead of forwards into the 21st century.

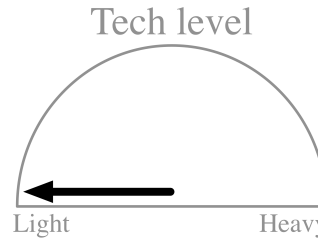
And, more bluntly:

Large, unified EHR systems are a really bad idea. A much better idea is loosely coupled specialist systems, each with a narrow interface, mirroring object oriented systems and allowing full knowledge encapsulation.

Exercise for the student: how does this destroy the idea of allowing the patient access to the EHR as it is implemented today?

[3]Except as better, quicker, and perhaps even more gratifying encounters with primary care.

[4]Which wasn't much.



CHAPTER 4

The History of medical records

We have to know where we came from to understand how we arrived where we are now.

As in all books, there is this "history" thing. But in this case, the history is essential to understand why things are as bad as they are.

In what follows, remember that I'm old enough to have actually lived through the described evolutionary stages myself, and I ain't dead yet.

4.1 The absence of records

Not long ago, some family practitioners had no medical records at all. When I first took over a practice in Belgium in the 80's, the "records" I got consisted of two stacks[1] of documents:

1. Letters, lab reports, and other documents that my predecessor had not yet seen and discussed with the patient, in reverse chronological order. In other words, they were dumped on top of each other as they came in.

2. The same kind of documents, after they'd been seen, in order of their processing. In other words, they were dumped on top of each other in the second heap once seen and discussed with the patient.

[1] Literally stacks: one of them was a meter high, the other a third of that.

The system, if you can call it that, worked as follows: the patient comes in and asks what the specialist said or what his blood tests showed. The doctor then asked around what time the visit to the specialist occurred or the drawing of blood, then proceeded to locate the document in stack number one. After reading it and discussing it, he prescribed something or other and off the patient went. The document ended up on stack number two and was never seen again. The whole incident then lodged somewhat loosely in the memory of the doctor and hopefully more permanently in the memory of the patient.

Obviously, this "system" can only work for a limited number of patients, and only if they stay with the same doctor for a long time. This kind of patient-doctor relationship was common back then, but is rapidly becoming extinct as both doctors and patients become much more mobile. Also, the number of doctors, both general practitioners and specialists, involved in the care of a patient, is increasing, resulting in the number of contacts between each doctor and any particular patient going down.

But the worst aspect of this old type of doctor-patient relationship is that it is dangerous. Important details are forgotten, and the patient's history can't effectively be transferred from one doctor to his successor.

4.2 Paper-based mementos

Obviously, this was a terrible state of affairs. Around this time, most GPs in Belgium started keeping a real medical record for two reasons:

1. Fear of lawsuits. If you're sued for malpractice and you have no records at all, you're doomed.

2. To memorize details, such as exactly which medicine was prescribed for exactly what period of time, exactly when. And blood pressure measurements, and such.

In other words, the paper record evolved to store hard to remember details in diagnosis and treatment on the one hand, and simultaneously to be a log of all interactions with the patient for legal reasons. The knowledge about the patient as such, his diseases, preferences, and most of all the overall plan in the diagnosis and treatment, was not so much written down as memorized by the doctor. After all, the patient always went to the same doctor anyway, so why write it down?[2]

Sweden had proper records for patients much earlier than Belgium. General practitioners ("provincial doctors") had an obligation to keep notes on

[2]It was also a great incentive for the patient to stick to the same doctor.

both patients and events in the environment, and this started roughly 200 years ago. They had to report yearly to the authorities on the state of health and risk factors. About a hundred years ago, there were already regulations about the minimum content of medical patient records, but not on its form. Most records were extremely brief, and most information about the patient was never written down, but maintained in the head of the doctor. Even as late as 50 years ago, most records were kept on index cards, since that was ample space for the brief information.

In the 1970's, it became more common to have several doctors managing the same patients, so more extensive records were needed. A Swedish standard form of medical records was developed (the Spri project), and a number of elements of the records became well defined.

The problem-oriented medical record (POMR)[3] was introduced in the 1970's. As part of POMR, the subjective-objective-assessment-planning (SOAP) model for the record was introduced. The SOAP model, without the underlying POMR model, can be seen in widespread use in other parts of Europe, but is fairly uncommon in its pure form in Sweden today.

During the 1980's, computers came into medical practice on a large scale. The applications were largely based on the preexisting paper records and were in fact treated as a faster and more advanced version of the same basic information carrier. It was never conceived as anything more that a record that allowed efficient retrieval of essentially the same information as had previously been kept on paper and in folders.

A very good source for much more detailed information on the history, the structure, the events, and the systems (in Swedish), can be found on Bengt Dahlin's site[4].

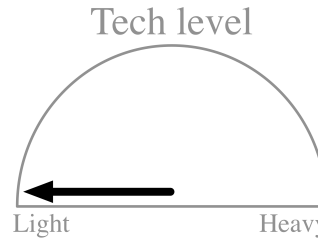
The key thing to remember is this:

The classic paper-based medical record was only intended to support the family doctor in the maintenance of hard to memorize details and was never designed to contain the overall picture of the patient or any diagnostic or therapeutic plan. Since there is no assigned and permanent doctor in most practices anymore, it makes no sense to automate the paper-based records without considerable adaptation to the new medium. But that is exactly what has happened. The electronic record is designed as if it was a paper record, but in digital form. All the opportunities for improvements that the digital form brings have been missed.

[3]The problem-oriented medical record was described by Lawrence L. Weed in his book published in 1970 [5]. A summary of the idea can be found at: http://medical-dictionary.thefreedictionary.com/Problem-Oriented+Medical+record

[4]http://www.bengtdahlin.se

Current medical records are characterized by a torrent of largely useless details without a unifying context.



Chapter 5

How does the EHR fail to assist us?

As a doctor, what help do I expect from a well-designed medical system? And why am I not getting it?

It's useful to compare the tools and methods we use in medicine with how other professions evolved. All professions have in common that they use knowledge, methods, and tools to achieve a goal. Medicine is more lopsided than most professions, lacking many of the tools we need for optimal performance. Not so coincidentally, it is unique in the sense that most tools for medical professionals are specified, designed and developed by lay people, while most other professions aren't that unfortunate. Pilots have a say in how cockpits should look, architects have influence on the software they use for design, but doctors often get medical-record software that no doctor would have specified. The result is a system with entire missing areas of coverage, and we need to look into what those missing areas are.

5.1 Compare to other knowledge areas

We don't have to make a choice between a system that assists us in decision making and a system that documents our actions, since the very process of working through a problem using knowledge support can be automatically documented, and in itself covers a major part of that documentation. One could argue that any documentation that covers a part of the medical events that are *not* part of a process, that is of a decision tree, except the patient's subjective history, by definition has no consequence and is thus of less value. The exception being, of course, parts of the medical process that are not

covered by the tool currently used for knowledge support, but which should have been.

With this reasoning, we arrive at the conclusion that any free-form medical-record notes are a symptom of defective knowledge support functionality, which is a good description of *all* of our notes as done currently, since there is no knowledge support functionality at all worth the name in any of our systems.

To see how the EHR fails in assisting doctors in their work, we need to compare it to other known processes that do work much better, and the example I'm choosing is "fixing a computer", since that is what most of us do far too often.

If we put the process of "fixing a computer" right next to the process of "fixing a human", we can find the same five stages or phases in both processes:

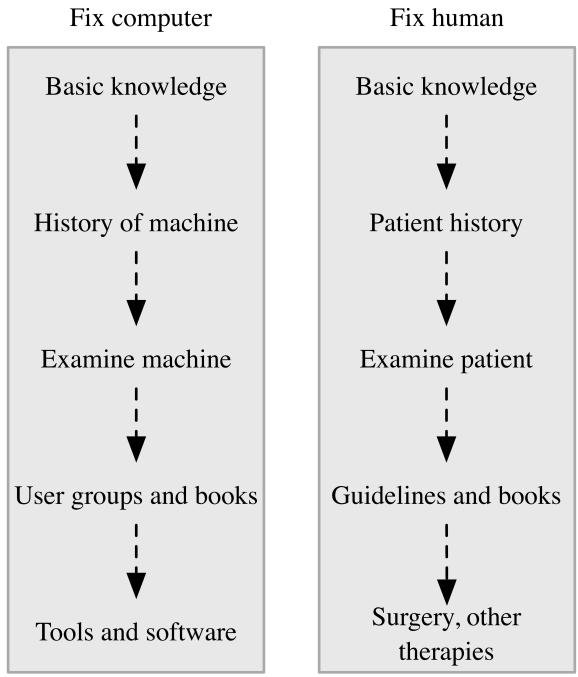


Figure 5.1: Comparing how to fix two similarly complex things.

The first part, "basic knowledge" (of computers), has the corresponding

"basic knowledge" (of humans) in the other column. These requirements are somewhat analogous for the two knowledge areas.

The second step, the "history" part is the only thing we have developed to a significant extent in EHR systems. Similar to the "computer fixing" process, this is not the most important step. It's nice to have, sure, but surprisingly easy to live without.

When we get to step three, "examine machine" vs "examine human", we find support for some, such as radiology, lab, and most therapies.

The real "meat" of the process to fix a computer is in the fourth step: "user groups and books". Not many computers would be fixed today without the ability to reference what other people have seen and how to fix it (or not, as may be). Trying to fix a computer without any reference to these resources is doomed to failure except in a few trivial cases.

Actually, the same is true for medicine. Since we don't have the same kind of easily available resources, medical practice is still in the 1980's if we compare it to fixing computers. We're still offline, so to speak. We can fix humans, but the only knowledge we can use is what was hammered into us at medical school, or that we can fortuitously remember from a more recent CPE class.

That fourth step, the step that really makes a difference, is all about gathering recent knowledge about a problem and its solution. That part is missing entirely from the EHR systems we have today. Yes, those resources are available elsewhere, but they are both hard to locate and hard to use *while examining and treating a patient*, the very time when these tools could make the most difference.

Let's make another comparison: the architect's productivity has certainly increased with the introduction of email and software that allows him to both write and maintain textual documentation, but the real advance of architecture is enabled by computer-aided design software. Where is the computer-aided design software equivalent for medicine?

Since our EHR systems are specified by civil servants, it's only natural that any provision for providing better healthcare, quicker diagnostics and more consistent therapies, have been left out entirely. None of these things matter to the administrators, so they simply don't care to require any such functionality.

You get what you pay for, or rather, you get what he who pays for it asks for. And healthcare functionality is not part of that.

5.2 What should I do?

Building on the example in the previous section, we can translate that into the steps that follow.

The first thing the software should help me with is a guideline on basic stuff, such as:

- What history elements must I query the patient about?
- Which clinical signs and examinations are basic to most problems and should be considered[1]?
- If I have a working hypothesis, the software should present me with signs and techniques that can be helpful to confirm or exclude that hypothesis. It should also present the major criteria for a diagnosis, without drawing any conclusions automatically. For more on this, see my discussion about active software (chapter 15).

5.3 How should I do it?

When I do a clinical examination or determine clinical signs, the software should:

- List the signs and examinations related to this, their names and a list of possible outcomes.
- Give me an informational page showing me how to perform the clinical examination.
- Give me information about what the clinical examination results mean and imply. Included here should be pointers to other examinations that could be valuable to complement the result.

If I decide to order a diagnostic test in the form of lab tests or technical diagnostics such as X-ray or ultrasound, the software should assist me as follows.

- Show which diagnostics are available.
- Show what prerequisites apply to the diagnostics, and when they shouldn't be performed (contra-indications).

[1] "Considered" means just that; you should think about it, and do it if you think you must. The point here is that if you don't do something in a guideline, it should be because you had a reason not to do it, not that you simply forgot.

- Show the relative cost of the diagnostic, both in resource use and in costs to the patient (radiation load, pain, risk for complications, time, etc).
- Show *where* the test can be done, and what provider to send the order to.
- Help me fill in a form for ordering the test, including all the elements the intended recipient of the form has determined is needed for the order.
- Helpfully propose the right documents from the records to include with the referral.

5.4 What did I forget?

The application should point out to me what I forgot, such as:

- Which other diagnoses should I consider and exclude for this patient? That is, present me with a list of "differential diagnoses".
- Which lab tests or diagnostic tests have I forgotten to perform to confirm this tentative diagnosis?
- Which therapy have I forgotten to start, or stop, for this diagnosis?
- Which reporting have I forgotten to perform?

As before, the system shouldn't tell me what to do, only what to *consider*. As a user, I should feel comfortable that I've considered all the angles and that my choices are made not from ignorance or happenstance, but from weighted judgement of all the elements.

5.5 History in context

When retrieving the history of the patient, the EHR system should present a list of issues for the patient, and all notes, conclusions, examinations, and other documents *in the context of their respective issue.*

Any notes, conclusions, or findings relevant to more than one issue should be presented in the context of *all* those issues.

Part II

Current systems

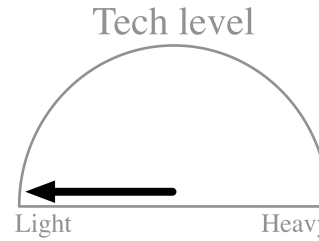
Summary

In these chapters, you will find an example of a current electronic healthcare record system, and how we work with it. We'll also describe which knowledge support systems are available and make a few points about their usefulness, or lack thereof, and try to identify what is missing to make them more useful.

We'll also analyze how a doctor works, and how she interacts with the electronic healthcare record, including the limitations due to time and place of use.

Finally, we'll go into the information model that is used in current systems, and what's wrong with it.

This part is mainly catering to the non-physician reader, providing the background necessary to understand why our current systems don't satisfy the real needs of doctors, and by implication, what needs to change. Doctors are also encouraged to read these chapters to see the assumptions and the background I have when writing this text. Contexts vary, and yours may be different.



Chapter 6

The goal of the system

What seems to be the goal of current systems? Are we happy with how that turned out?

The goal of the system should be to help the healthcare professional do a better job. Some functions of EHR systems support data entry and communication; functions which are generally fairly well developed in current systems. What is almost entirely lacking, however, is knowledge and process support, as I described on page 39. Some simpler processes for nursing can be found here and there, but nothing really significant is going on.

Note well: I'm not saying there are no knowledge or process support initiatives out there, but what I *am* saying is that there are no significant such initiatives that are fully a part of the EHR. There's a great number of such stand-alone initiatives, fragmented and each covering just a part of the problem area. But even if there were wall-to-wall coverage in such a tool, it's still not a part of the healthcare documentation process. If you have to interrupt the regular documentation process to go look up something, and what you find does not automatically become documented, it's not only harder to do, it also does not enrich the documentation with any reasoning resulting from the knowledge you looked up. It's not enough to define in the documentation what you do to the patient, it's essential that you document *why*, or *why not*, you are doing it, and that motivation is lacking if the knowledge support is not integrated into the same tool as the documentation.

One could argue that the EHR has only a documenting function, and that the knowledge support function should be separately implemented

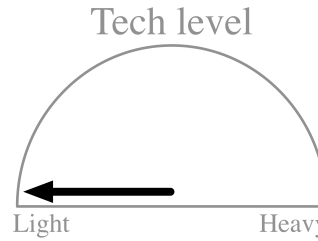
and provided, but that would by necessity imply duplication of effort and redundant information. One could also argue, as I do, that the EHR has erroneously been viewed and developed as a documenting tool instead of a supporting tool. Current systems are also increasingly subverted to become data gathering tools for management purposes *instead* of a tool for healthcare provision, due largely to a power grab of administrators in healthcare.

A nice illustration of how badly current systems are conceived, at least from a practical healthcare perspective, came in an article celebrating the joy of having *someone else* update the EHR:

> "Without much fanfare or planning, scribes have entered the scene in hundreds of clinics and emergency rooms. Physicians who use them say they feel liberated from the constant note-taking that modern electronic health records systems demand. Indeed, many of those doctors say that scribes have helped restore joy in the practice of medicine, which has been transformed — for good and for bad — by digital record-keeping."[1]

If you introduce a new IT system to help a particular professional do a better job, and one of the most celebrated advances in the use of that IT system is having someone else manage it so you don't have to come into contact with it, your system is pretty much condemned as nothing but a drag on the user. It's hard to think up a more damning verdict than that.

[1] The New York Times, January 14, 2014, "A busy doctor's right hand, ever ready to type", http://www.nytimes.com/2014/01/14/health/a-busy-doctors-right-hand-ever-ready-to-type.html



Chapter 7

Legacy EHR Example: Cosmic

To show what I'm complaining about, let's deconstruct a typical current EHR system.

We'll illustrate how current systems work by using examples from Cambio Cosmic, a system the author has used quite a bit. The descriptions and screenshots aren't of the most recent version and some improvements have been made since these were taken[1], but fundamentally, it's still working according to the same principles. It's important to stress that Cosmic is certainly not the worst system out there; it could conceivably even be the best. But it is definitely representative of the sorry state of systems we have.

In the first screenshot (figure 7.1) we see a window showing the notes for the current patient. At the top of the screen we have the demographics, showing the personal number and name of the patient (a fake test patient). The top right shows three buttons in different colors indicating different kinds of warnings. The right large pane shows the contents of notes selected from the left pane.

The left pane lets you select what to show in the right pane. It's a list of sources of documentation, typically different departments within primary care or hospital care. It's basically an organizational chart. County-owned primary care is one section, while privately-owned primary care centers have their own sections. Each speciality department, such as "urology" and "orthopedics", is grouped under a larger umbrella (such as "surgery"). This

[1] I would like to use more recent screenshots, but I very much doubt they'd let me take them.

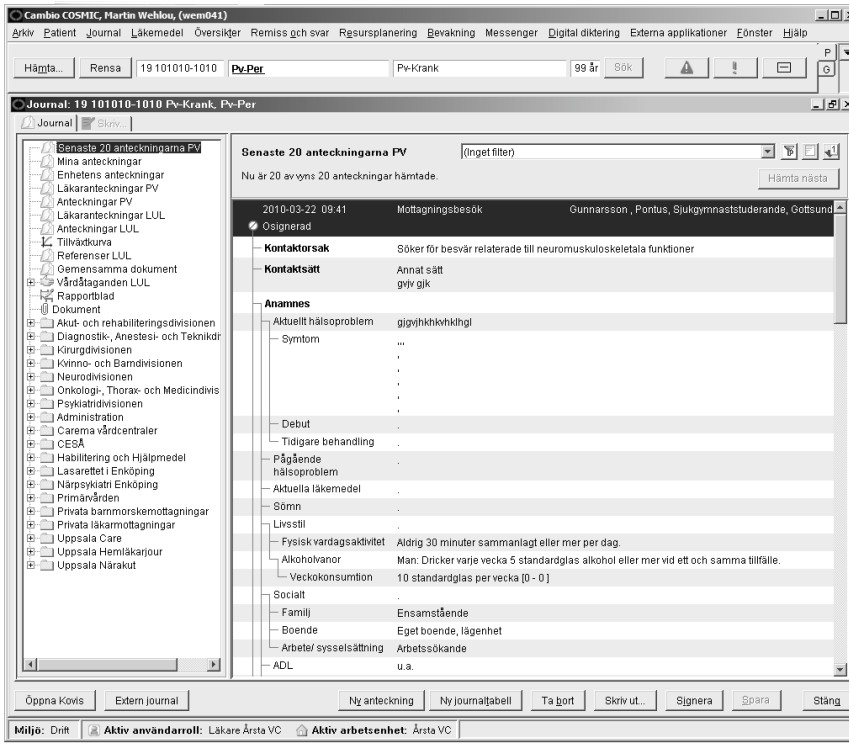


Figure 7.1: Cosmic, viewing the notes in the record

must be wonderful for a civil servant to see, but is pretty much pointless to the work of a doctor. Yes, we *do* want to know occasionally *where* the patient has been, but generally we're much more interested in *what* and *why*, which this list doesn't help with at all.

This left pane is a fantastic illustration of what's wrong with current systems. It's pretty clear it was designed, or at least specified, by a lay person *imagining* what he would like to see there if he was a doctor, and then forcing doctors to see just that. But I very much doubt that a properly informed and independent doctor would have chosen that view to occupy such a dominating part of the workflow.

If you look at the top of the left pane in figure 7.1, you'll see two tabs. The left one says "Journal" ("Notes"), while the right one says "Skriv" ("Write"). This is where you switch both panes over from reading to writing.

When in writing mode, the left pane changes to show all the "keywords" or subtitles, you can enter information into (see figure 7.2). Whatever you

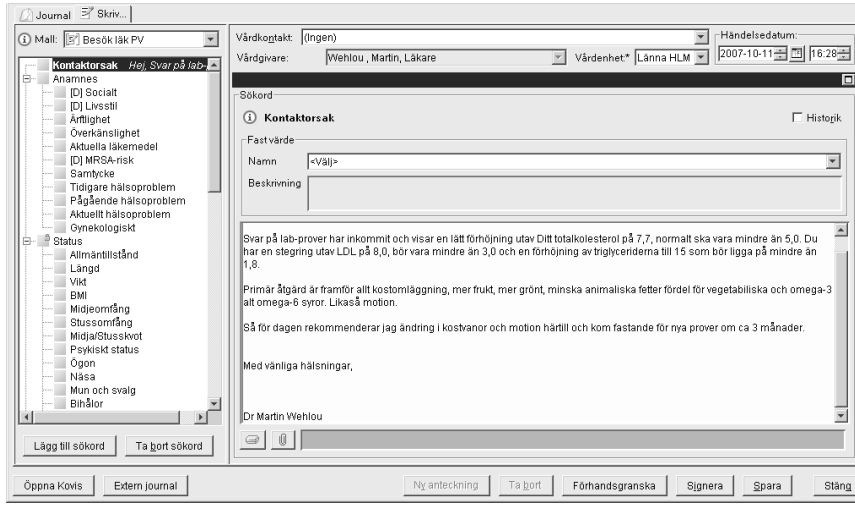


Figure 7.2: Cosmic in notes writing mode

write in the right pane is entered into the field contents of the keyword you have selected on the left. At the top, above the entry field in the right pane, you can select date, time, contact (encounter), and where you are. Again, in this screen, you see an emphasis on "who", "where", and "when", but not so much on "what" and "why".

What we also see is a structure determined by those keywords to the left. This neatly shows what the developers have understood when we say "structured records", namely a text divided into sections according to some fairly random list of discernible terms. That's not what most doctors would regard as a medically sensible structure. Anyone can slice any text any number of ways without improving its utility. The "structure" we'd like to see is a division into diagnostic plans, intentions, deductions, findings, and the underpinnings of those. There's nothing of that in Cosmic, or in any other current EHR systems.

When creating a prescription in Cosmic, it looks like in figure 7.3. Clearly, the entire working space on the screen is occupied by all the stuff you need to prescribe a medication. The only other things that still appear are my name, the patient's name, and the contact date and time. That's it. It's obvious that the developers didn't see that prescriptions have anything to do with notes, referrals, lab, or just about anything else in the records. Or that the doctor would need to refer to anything else while creating prescriptions. When seeing a patient for hypertension, for instance, it's as easy (or difficult) to prescribe a blood pressure lowering medication as it

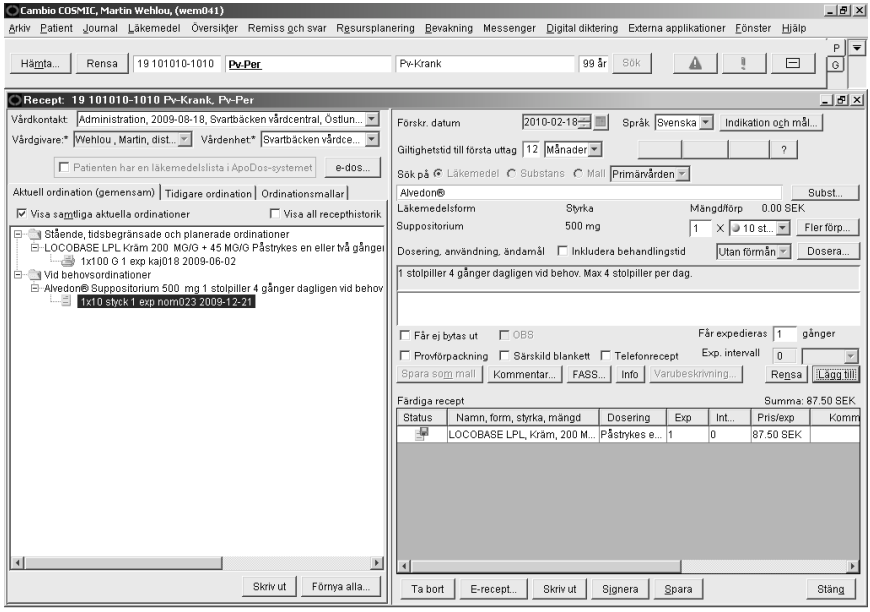


Figure 7.3: Creating a prescription in Cosmic

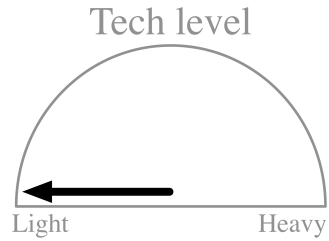
is to prescribe the pill, or morphine, or a drug against Parkinson's disease. The system is totally oblivious to what disease is being managed. It doesn't help and it doesn't hinder (too much). If I send a referral, likewise the system allows me, with exactly the same level of support or hindrance, to send it to a cardiologist, a kidney specialist, or, why not, a psychiatrist, or a hearing aid service. It has no preferences and is completely oblivious to the medical context.

If you had created a system for photographers with this attitude and domain implementation, it would contain functions for finding models, taking pictures, developing film, making copies, and invoicing clients, flushing toilets, and clearing gutters, but the picture taking would have no relationship to either models or film, the development task would involve bottles of solutions, but no film or copies, while the invoicing would have no relationship to anything else, making it as easy to invoice a photo shoot as a dozen dead rabbits. No photographer in his right mind would buy an application like that.

I have no screenshot of how you write referrals with Cosmic, but you're not missing much. It's the same kind of thing, where once you start writing, you see nothing else, and cannot look up anything else. The referral itself is not connected to any particular disease or other documents, and is, for

all practical purposes, as integrated into the patient's narrative as a dozen dead rabbits.

I said it in the beginning of this chapter and I'll say it again: *Cosmic isn't worse than anything else out there, it may even be the best system of them all. It's probably pretty much representative of the current market.* Which, when you think of it, isn't a very comforting thought.



Chapter 8

Knowledge support

What kind of knowledge support do we have today? Do we use it? If not, why?

As doctors or nurses, we can't possibly keep everything we should know just in our heads. And even if we could, things change as science advances. There is such an enormous amount of data we have to make use of, and an ever growing avalanche of new findings. There's no way we can drink from this fire-hose and do real work at the same time.

Depending on how fresh the information is, and the level of detail, it is available as original articles, review articles, textbooks, and/or guidelines.

8.1 Original articles

According to the Medline Fact Sheet[1], there are more than 19 million references to original articles in their database. Another 2,000 - 4,000 are added *daily*. Even though everything we need to know in our daily practice is in there *somewhere*[2], this information isn't useful in clinical practice in its raw form. Researchers working on a particular topic have great use of this resource, but for a clinician having a patient in her office, it's practically useless. You can't just say to the patient: "I have to go read up on your problem for a bit. Grab a coffee, and I'll be back in a couple of weeks."

[1] http://www.nlm.nih.gov/pubs/factsheets/medline.html

[2] I'm glossing over the practical techniques, the judgement, and the experience we need as doctors, focusing solely on the scientific findings.

At times, publications containing remarkable and important discoveries are widely read by doctors, but this is just an infinitesimally small fraction of the total published mountain of articles. And we can't just ignore the rest. Most of them form the basis of future diagnostic and therapeutic principles, so we have to use a system to reduce them to a more digestible form.

8.2 Review articles

Well-informed authors regularly write reviews of the most important papers in their field and publish those in the same type of medical journals where the original papers are usually published. These review articles allow non-specialist doctors to get a good overview of a particular subject and the current state of knowledge about it. While writing these reviews, the authors use their best judgement to sift through the underlying original articles to bring forward the most relevant and trustworthy findings, so the rest of us don't have to do that.

These review articles are still too detailed and specialized to be directly useful in patient care in general, but are often just fine as a tool to keep up in a field in which the doctor sees a lot of patients, but not necessarily does research.

8.3 Textbooks

Textbooks constitute the next level up, and the quality of the information is similar to reviews, but as a collection of related subjects for a certain medical speciality.

While original articles and review articles presume the reader is already familiar with the subject, the textbook does not, and starts from basics. Textbooks are rarely useful for a practicing doctor, except as a reminder of what she learned in medical school. Additionally, text books are much too expensive to buy just to keep up with the science. They're also not updated quickly enough for that use.

8.4 Guidelines

Guidelines are based on original research and reviews, and turn that content into practical use advice. If there's a study that says it's advantageous to the patient to apply therapy X, the guideline recommends therapy X, at least if it's available in the relevant region.

In theory, guidelines are entirely based on science, the interest of the patient, and the resources available, and are short and sweet enough that they

can be read, understood, and applied while the patient is present. In practice, however, these guidelines are often influenced by considerations that have nothing to do with the patient's immediate interest, but by political and economic incentives. If those economic considerations are of the kind that optimizes healthcare in a larger perspective, it's reasonable to accept that, but manipulative bureaucrats sometimes compromise the integrity of these motives to such a degree that guidelines are getting a bad rap as being a tool for civil servants to control medical professionals, rather than a tool to improve the quality of the direct patient work. If we can't trust the motives behind a guideline, we're not likely to seek it out and use it, either.

Somewhat unexpectedly, it's hard to find useful guidelines in English, but one decent example can be seen at a site loosely connected to Oxford University (according to the text on the site). The guideline chosen for the purposes of this discussion, is their guideline on the work-up and management of chronic heart failure, and was found at this URL[3], which, of course, has a good chance of not existing anymore once you try it. That's the internet for you.

This guideline is in the form of a PDF file and summarizes the causes, the diagnostic procedures, and the recommended treatments stratified into levels of seriousness. The guideline document includes text, flowcharts, tables, and forms. Even though the information is solid and useful, the document is a mashup of different kinds of functionality, all squeezed into one single document, severely impacting its practical usefulness. Let's look at a few aspects of this guideline[4] and its construction, and it'll become clearer just what I mean by that.

In figure 8.1 we see the beginning of the guideline, where there is a short overview of the basic diagnostic procedures. As doctors we're supposed to know all this by heart, but we don't always remember every detail. It could be too long ago, too detailed, or we can have a bad day, so this is excellent as a brief reminder.

Figure 8.2 shows a part of the therapeutic discussion from the guideline. Again, it's terrific as a reminder, but it is so much more than that. It's also an *up to date* reminder, (or it *should* be[5]), potentially alerting us to changes in recommended therapies caused by new discoveries that may have shown newer agents to be more effective than older agents, or shown some agents not to be as effective or free of adverse effects as we thought. In other words, however well you know past recommendations by heart, checking

[3]http://static.oxfordradcliffe.net/med/gems/CHF.pdf

[4]The CHF guideline was reproduced with permission from Dr Chris Bunch and Dr Jeremy Dwight, NHS Oxford, UK.

[5]It's absolutely essential that the guideline includes the date it was last updated, so that we can judge its actuality, but this guideline does not. That is a serious shortcoming.

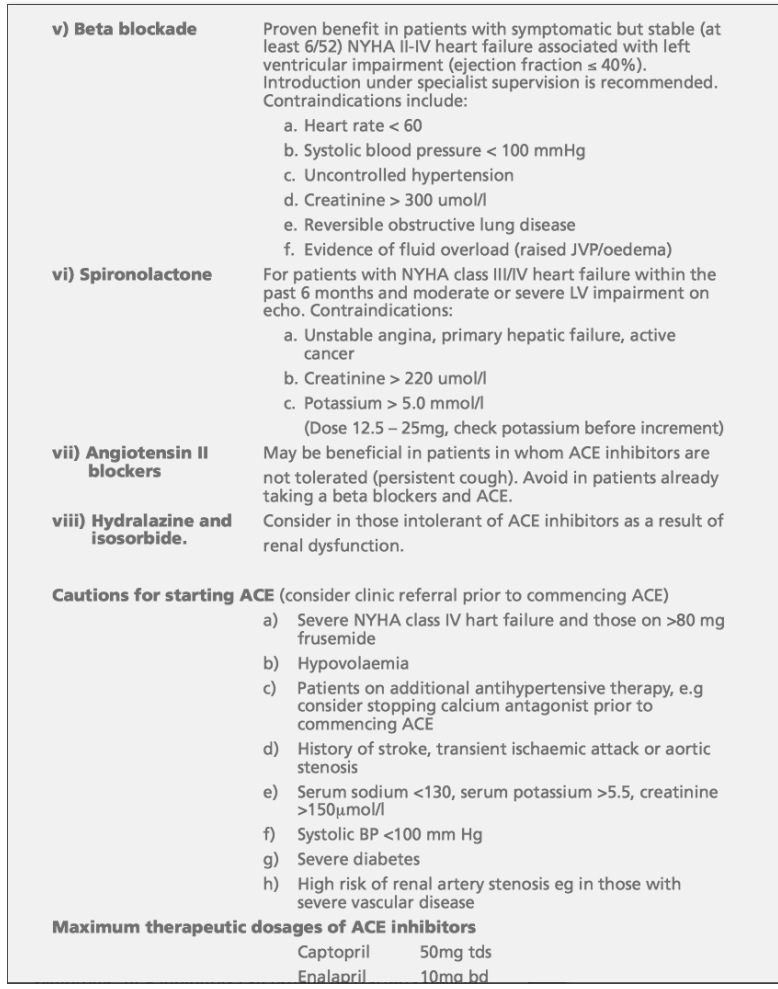
5 CHRONIC HEART FAILURE
Diagnosis and investigation

Consider aetiology	**Ischaemia** -History of angina or myocardial infarction, presence of vascular disease elsewhere, eg. Carotid bruits or claudication, multiple risk factors for IHD eg hypercholesterolaemia, smoking, diabetes.
	Hypertension - History of hypertension, presence of left ventricular hypertrophy on ECG or echo
	Valvular Heart disease - Cardiac murmur, history of rheumatic fever, history of valve replacement
	Others – Alcohol, thyroid disease.
Symptoms and signs	The symptoms and signs of chronic heart failure (CHF) are often non-specific (eg tiredness, shortness of breath, chest crackles, dependent oedema). The most specific signs are a 3rd heart sound and a displaced apex beat
Baseline investigations	*(essential for referral for echocardiography see Figure 6)*
	ECG (A normal ECG makes a diagnosis of heart failure unlikely.)
	CXR, FBC, electrolytes and creatinine, thyroid function, gamma GT, glucose.
Echocardiogram	An **echocardiogram** is the key investigation in suspected CHF. It identifies potentially treatable causes of heart failure (e.g. valvular disease) and can confirm the presence or absence of left ventricular systolic dysfunction.
Assess Severity – *New York Heart Association Classification*	
	Class 1 No limitation. No fatigue dyspnoea or palpitations on physical exercise
	Class 2 Slight limitation in physical activity. Comfortable at rest, but fatigue, palpitations or dyspnoea on exertion
	Class 3 Marked limitation in physical activity, but comfortable at rest
	Class 4 Symptoms present at rest

Management *(see Figure 7)*

i) Aggravating factors	Aggravating factors can include
	a. Drugs : beta blockers, NSAIDs
	b. Diet: excess alcohol, excess salt.
ii) ACE inhibitors	ACE inhibitors are first line treatment for established heart failure. They prolong life, delay progression and improve symptoms. ACE inhibitors can usually be started safely in primary care if introduced in low dose and care is taken to avoid hypotension (see Dargie et al referenced overleaf for guidelines for starting ACE inhibitors in general

Figure 8.1: Guideline for management of chronic cardiac failure, diagnosis and investigation.



v) Beta blockade

Proven benefit in patients with symptomatic but stable (at least 6/52) NYHA II-IV heart failure associated with left ventricular impairment (ejection fraction ≤ 40%). Introduction under specialist supervision is recommended. Contraindications include:

a. Heart rate < 60
b. Systolic blood pressure < 100 mmHg
c. Uncontrolled hypertension
d. Creatinine > 300 umol/l
e. Reversible obstructive lung disease
f. Evidence of fluid overload (raised JVP/oedema)

vi) Spironolactone

For patients with NYHA class III/IV heart failure within the past 6 months and moderate or severe LV impairment on echo. Contraindications:

a. Unstable angina, primary hepatic failure, active cancer
b. Creatinine > 220 umol/l
c. Potassium > 5.0 mmol/l

(Dose 12.5 – 25mg, check potassium before increment)

vii) Angiotensin II blockers

May be beneficial in patients in whom ACE inhibitors are not tolerated (persistent cough). Avoid in patients already taking a beta blockers and ACE.

viii) Hydralazine and isosorbide.

Consider in those intolerant of ACE inhibitors as a result of renal dysfunction.

Cautions for starting ACE (consider clinic referral prior to commencing ACE)

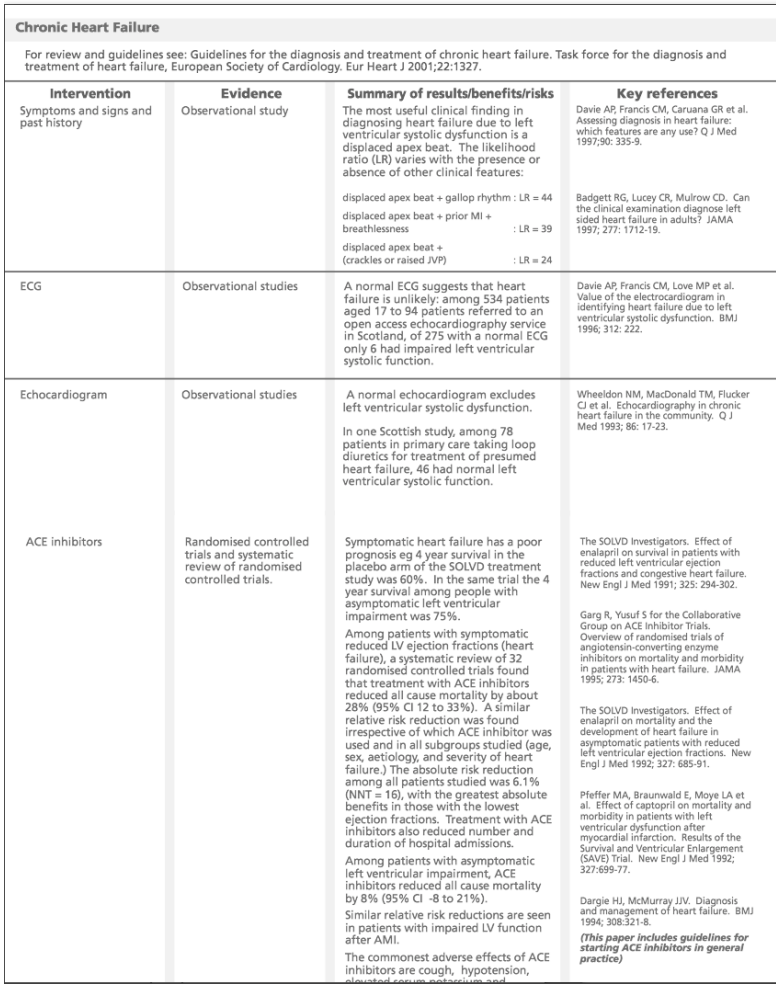
a) Severe NYHA class IV hart failure and those on >80 mg frusemide
b) Hypovolaemia
c) Patients on additional antihypertensive therapy, e.g consider stopping calcium antagonist prior to commencing ACE
d) History of stroke, transient ischaemic attack or aortic stenosis
e) Serum sodium <130, serum potassium >5.5, creatinine >150μmol/l
f) Systolic BP <100 mm Hg
g) Severe diabetes
h) High risk of renal artery stenosis eg in those with severe vascular disease

Maximum therapeutic dosages of ACE inhibitors

Captopril	50mg tds
Enalapril	10mg bd

Figure 8.2: Chronic cardiac failure, part of the treatment description.

through this list can always teach you something new and bring you up to date quickly and easily.



Chronic Heart Failure

For review and guidelines see: Guidelines for the diagnosis and treatment of chronic heart failure. Task force for the diagnosis and treatment of heart failure, European Society of Cardiology. Eur Heart J 2001;22:1327.

Intervention	Evidence	Summary of results/benefits/risks	Key references
Symptoms and signs and past history	Observational study	The most useful clinical finding in diagnosing heart failure due to left ventricular systolic dysfunction is a displaced apex beat. The likelihood ratio (LR) varies with the presence or absence of other clinical features: displaced apex beat + gallop rhythm : LR = 44 displaced apex beat + prior MI + breathlessness : LR = 39 displaced apex beat + (crackles or raised JVP) : LR = 24	Davie AP, Francis CM, Caruana GR et al. Assessing diagnosis in heart failure: which features are any use? Q J Med 1997;90: 335-9. Badgett RG, Lucey CR, Mulrow CD. Can the clinical examination diagnose left sided heart failure in adults? JAMA 1997; 277: 1712-19.
ECG	Observational studies	A normal ECG suggests that heart failure is unlikely: among 534 patients aged 17 to 94 patients referred to an open access echocardiography service in Scotland, of 275 with a normal ECG only 6 had impaired left ventricular systolic function.	Davie AP, Francis CM, Love MP et al. Value of the electrocardiogram in identifying heart failure due to left ventricular systolic dysfunction. BMJ 1996; 312: 222.
Echocardiogram	Observational studies	A normal echocardiogram excludes left ventricular systolic dysfunction. In one Scottish study, among 78 patients in primary care taking loop diuretics for treatment of presumed heart failure, 46 had normal left ventricular systolic function.	Wheeldon NM, MacDonald TM, Flucker CJ et al. Echocardiography in chronic heart failure in the community. Q J Med 1993; 86: 17-23.
ACE inhibitors	Randomised controlled trials and systematic review of randomised controlled trials.	Symptomatic heart failure has a poor prognosis eg 4 year survival in the placebo arm of the SOLVD treatment study was 60%. In the same trial the 4 year survival among people with asymptomatic left ventricular impairment was 75%. Among patients with symptomatic reduced LV ejection fractions (heart failure), a systematic review of 32 randomised controlled trials found that treatment with ACE inhibitors reduced all cause mortality by about 28% (95% CI 12 to 33%). A similar relative risk reduction was found irrespective of which ACE inhibitor was used and in all subgroups studied (age, sex, aetiology, and severity of heart failure.) The absolute risk reduction among all patients studied was 6.1% (NNT = 16), with the greatest absolute benefits in those with the lowest ejection fractions. Treatment with ACE inhibitors also reduced number and duration of hospital admissions. Among patients with asymptomatic left ventricular impairment, ACE inhibitors reduced all cause mortality by 8% (95% CI -8 to 21%). Similar relative risk reductions are seen in patients with impaired LV function after AMI. The commonest adverse effects of ACE inhibitors are cough, hypotension, elevated serum potassium and	The SOLVD Investigators. Effect of enalapril on survival in patients with reduced left ventricular ejection fractions and congestive heart failure. New Engl J Med 1991; 325: 294-302. Garg R, Yusuf S for the Collaborative Group on ACE Inhibitor Trials. Overview of randomised trials of angiotensin-converting enzyme inhibitors on mortality and morbidity in patients with heart failure. JAMA 1995; 273: 1450-6. The SOLVD Investigators. Effect of enalapril on mortality and the development of heart failure in asymptomatic patients with reduced left ventricular ejection fractions. New Engl J Med 1992; 327: 685-91. Pfeffer MA, Braunwald E, Moye LA et al. Effect of captopril on mortality and morbidity in patients with left ventricular dysfunction after myocardial infarction. Results of the Survival and Ventricular Enlargement (SAVE) Trial. New Engl J Med 1992; 327:699-77. Dargie HJ, McMurray JJV. Diagnosis and management of heart failure. BMJ 1994; 308:321-8. ***(This paper includes guidelines for starting ACE inhibitors in general practice)***

Figure 8.3: Summary table with literature references for chronic cardiac failure.

In figure 8.3 we see more or less the same information, but in tabular form, with indications of trustworthiness in the "evidence" column, and with literature references in the last column. That last column is essential, since it ties the recommendations to the underlying original data. The guideline

should never be the expression of just the author's personal preferences, it must *provably* be based on neutral scientific evidence, and that's where this last column comes in. Without an explicit base in original scientific work, the guideline loses most of its value.

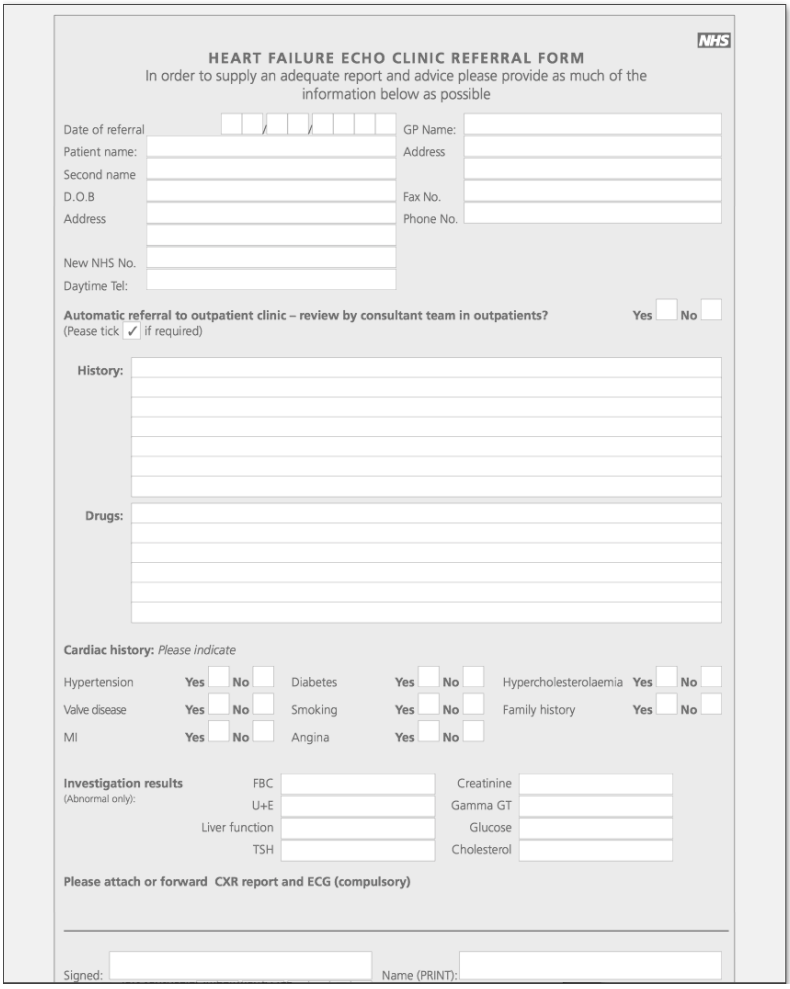
Not everything in a guideline can be based on explicit scientific material, however. Some of it is based on generally agreed good practices, but it's important that those parts of the advice in the guideline are clearly recognized as such. The reader should always have the ultimate say in how much authority she lends to the guideline in the individual patient case, and should not have to take anything in the guideline on faith, or as a result of belief in the authority of the author alone.

When we're working through a guideline, in particular the recommendation parts, we always weight the recommendations against the totality of the patient we have in front of us. Some recommendations can't be followed due to other diseases the patient has, personal preferences of the patient, and many more factors. It's impossible, and unproductive, to even attempt to include all those considerations into a guideline. If we as doctors weren't able to take those considerations into account ourselves, we shouldn't be doing this job, anyway.

But here's a problem. Reading the guideline and selectively applying the advice based on considerations that are not part of the text leaves no trace in the medical record. While the user is reviewing the guideline and reacting to it, she is performing the real essence of her profession, and is taking the important decisions about the patient's care, but it's not persisted anywhere. She can't scribble on the screen, striking through stuff, or underlining, adding checkmarks, etc. The guideline is her worksheet, the very embodiment of her reasoning, and she can't save it? That's ridiculous! In other words, however great the content of the guideline is, the form is wrong. It shouldn't be presented as a read-only webpage or even paper document, unless the user can scribble on it, modify it, and make it part of the patient's ongoing record.

The form we see in figure 8.4 only underlines this fatal flaw. The form is just sitting there, totally passive. Yes, the user can print it out and fill it out by hand[6]. The idea is fine, providing us with the right form, but the delivery is deeply flawed. This form should be different according to the organization and geographical location of the user, be sent automatically to the destination, become part of the patient's record automatically, and become connected to the reply, once it arrives. Clearly, this is too much too ask of a plain dumb PDF file on a website.

[6]Since the death of the typewriter, there's no other way to fill in forms, really. Unless you count spending hours installing and *cursing* Adobe Reader to kingdom come, for the pleasure of *trying* to hit those fields, then *going ballistic* as it can't save the filled in document anywhere, least of all in the medical record.



HEART FAILURE ECHO CLINIC REFERRAL FORM

In order to supply an adequate report and advice please provide as much of the information below as possible

Date of referral / /
Patient name:
Second name
D.O.B
Address
New NHS No.
Daytime Tel:

GP Name:
Address
Fax No.
Phone No.

Automatic referral to outpatient clinic – review by consultant team in outpatients? Yes No
(Pease tick ✓ if required)

History:

Drugs:

Cardiac history: *Please indicate*

Hypertension Yes No | Diabetes Yes No | Hypercholesterolaemia Yes No
Valve disease Yes No | Smoking Yes No | Family history Yes No
MI Yes No | Angina Yes No

Investigation results (Abnormal only):
FBC | U+E | Liver function | TSH
Creatinine | Gamma GT | Glucose | Cholesterol

Please attach or forward CXR report and ECG (compulsory)

Signed: Name (PRINT):

Figure 8.4: Referral form for chronic cardiac failure.

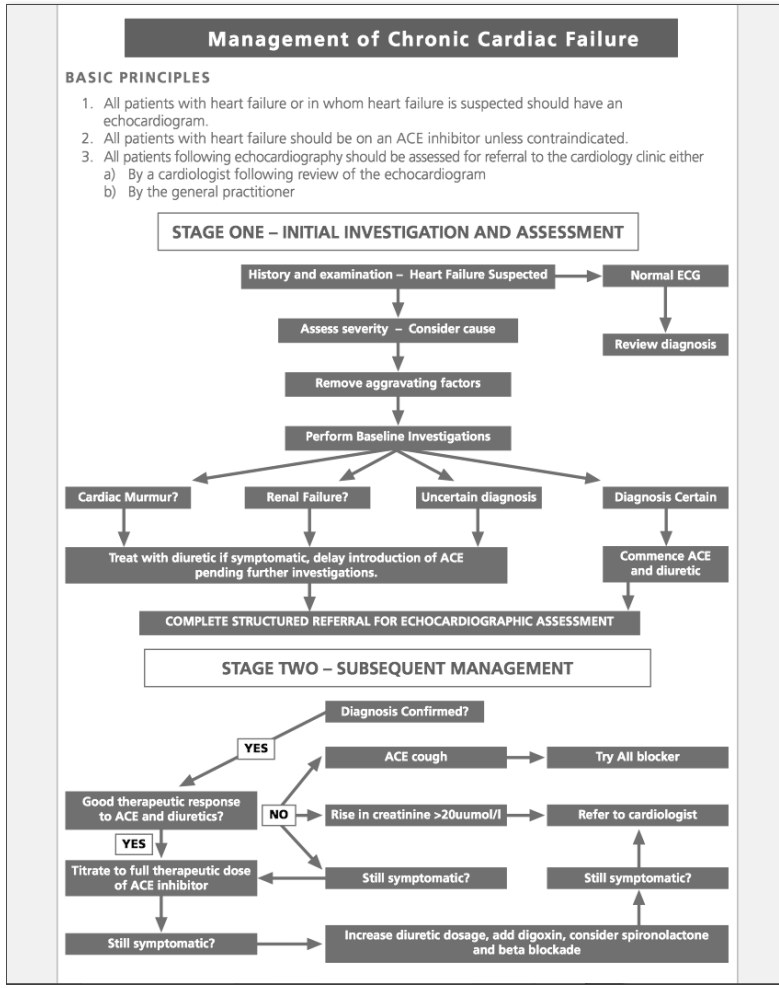


Figure 8.5: Flowcharts for investigation and management of chronic cardiac failure.

In figure 8.5, we see the final page of the guideline, with a neat flowchart representation of the management of chronic heart failure. This representation is great for a review of the guideline by the user, but since it's entirely passive, it has no immediate use in any software implementation.

There is a more philosophical problem with the flowchart representation, namely that it invites to being implemented as a process in medical records, and that would be tragic for two reasons. First, we have the "keyhole" effect discussed in section 15.1, that is an incentive for the user to hunt around in the program, answering questions in different ways just to see what conclusions and recommendations come out of the software. The other reason is that it is an invitation to lay management to try to automate away the need for doctors. The flowchart, after all, seems to indicate that the entire decision process can be reduced to just a few input factors and a few decision branches. Lay people tend not to see what we're using our training and experience for, once it's ostensibly reduced to a flowchart. In short, I think the flowchart here has more potential for damage than for good use.

8.5 Continued Professional Education

In most countries, doctors must follow a minimum amount of continued professional education per year[7]. The idea is to bring us up to speed, at least in our own speciality. This idea falls down when scrutinized, though.

The number of diseases we manage as doctors is too large to be comprehensibly covered in any particular year, so a choice must be made as to which subjects are covered. These choices have little relationship to the actual case load of each participating doctor, as a large part of the selection of cases is based on random factors. It's left largely to chance if you will receive training in any one particular subject with any regularity.

Also consider how much you can possibly remember from an education session, and for how long. The information is at the top of your brain, and in a useful state, only for a short time, maybe a couple of weeks at the most. After that, it sinks down into the same primordial ooze all the other knowledge from university days bubbles around in, and becomes effectively almost useless. Any patients with that disease you see while still having the information fresh in your mind will benefit, and possibly make the information lodge more permanently in your memory. But how many Parkinson patients will you see in the three first weeks after that one session on Parkinson you had last year? One?

[7]Sweden has no such obligation. I don't know what to think of that, really.

I'm not entirely alone in this pessimism. A paper from the Wellcome Trust[8] says:

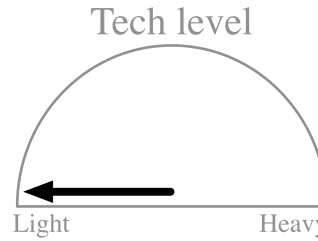
> "The researchers estimate that the time lag between research expenditure and eventual health benefits is around 17 years."

That number, 17 years, is pretty much the midpoint of the average career for a doctor, leading me to my own informal conclusion:

Nobody learns much of anything after graduating from medical school. Continued professional education, in practice, makes no useful difference.

Of course, there are exceptions to this very pessimistic rule. Any training in manual dexterity, what is called "craftsmanship" in section 2.4, is best taught using in-person meet-ups in clinics and operating theaters, or at nice hotels or spas with excellent restaurants and bars. But for theoretical knowledge, it is very doubtful this is an effective method.

[8] http://www.wellcome.ac.uk/stellent/groups/corporatesite/@sitestudioobjects/documents/web_document/wtx052110.pdf



CHAPTER 9

How is the record created?

We're entering data into the systems, but how and when do we do that?

Before designing an interface, and even before analyzing the meaning of the entered data, we have to know how and when the user creates the input. It is the physical environment and context around the place and time of input that largely decides the character and value of the data produced.

9.1 The input method

There are several ways data can be entered into the medical record. There are three major workflows, each with their own advantages and disadvantages.

Writing afterwards

Some doctors take short handwritten notes during the encounter, then transcribe those into long form notes in the medical-record system after the patient leaves. This implies that the doctor has some time alone between patients, or that she spends time at the end of the day transcribing notes for several patients.

Usually, referrals, reports, and attestations of different kinds are written in that same period, after the patient leaves. Some doctors write referrals and attestations with the patient present, but leave the dictation or writing of the record for after the patient has left.

Dictating afterwards

In this workflow, the doctor also takes some brief notes during the encounter, then arranges his thoughts and proceeds to dictate the notes after the patient has left, or possibly after hours. It's easier to mix up the history and findings of different patients if the dictation is all done at once like this.

Simultaneous writing

In this workflow, the doctor takes notes during the actual encounter, as the information is provided and develops.

This way of working is surprisingly easy on doctor and patient. As the doctor reacts visibly by typing to almost everything the patient says, the patient feels he is being taken seriously, is less prone to repetition, and also becomes more succinct and to the point. That's all good.

The major obstacle to this is that the doctor needs to learn touch-typing to be able to maintain eye contact with the patient, at least part of the time. But that isn't hard to achieve with a little training. It's orders of magnitude easier to learn to type well, as compared to learning to become a doctor in the first place.

Another advantage of writing while the patient is present is that if any point is missing or unclear in the narrative, it is much more straightforward to get clarification of these points as the patient is still there to help out.

Finally, since all note taking and writing of referrals is done before the patient leaves, there is no work left to do in the interval between patients or after hours. The patient also gets to be with the doctor for a longer period than if the writing or dictating needs to be done in his absence.

Curiously, in some countries doctors almost always write the record themselves, while in other countries it's rare. It seems to be a cultural phenomenon.

Obstacles to simultaneous writing

The major obstacle is the lack of typing ability in doctors. This can and should be overcome with suitable incentives and training[1].

Another more serious obstacle is the poor design of most EHR systems today. In most of them, it is impossible to refer to all the source documents you need while creating typical output. Often the editor window you use to write notes is the same window you use to look up older notes. The same

[1] In Sweden, at least, I'm getting a lot of pushback on this from several reviewers who *can* touch type but are adamant that it's still better not to be distracted by typing while the patient is present. It would be interesting, useful, and entirely feasible, to do an observational study on this, but I'm not holding my breath.

happens with referrals; while writing a referral you can't go back and read another referral, since these functions utilize the same window. You can't save a referral as a draft to go look something up, so you need to throw the draft away, go look up something, then start over. Adding insult to injury, you can't reuse an earlier referral, modify it and send it again. Since one of the most frequent workflows you need to handle is re-composing a referral that was bounced back from a referee[2], these defects combine to form a perfect storm of aggravation.

The systems are much easier to use if you *either* read documents *or* write them, but not both at the same time. It is also often extremely complicated to copy existing information into new forms and notes. Copy and paste is often poorly implemented, and even if it works, it's a very cumbersome method. If you dictate your records, this is not a problem, since the doctor then reads, while the secretary only needs to write while listening to the dictated recording, which explains how this terrible design came into being. The systems are in effect designed for a workaround; having a secretary type the medical record is a workaround for bad designs and defective medical records, and the medical-record system vendors keep designing for that workaround, perpetuating the problem.

The way the EHR systems are designed needs to be changed to take simultaneous writing into account. This, again, is probably different according to the culture of the country you're in.

9.2 The different results

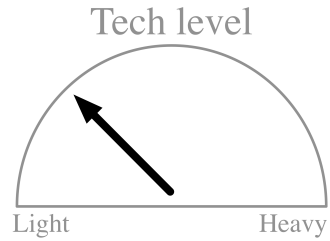
If the notes are created *after* the actual encounter with the patient, the contents of the notes will be different from those that could have been written *during* an encounter. The reason for this is that in the first case, the notes are an after-construction, where findings are already filtered through the conclusion the doctor reached. It's, in other words, colored by prejudice. As an example, we'll take the following scenario. If the doctor listens to a patient telling her about his sore right knee in full detail, and the patient also mentions he had a rash and a fever two weeks before his knee acted up, the rash and the fever will probably not end up in the patient history notes that the doctor writes down, unless she sees a connection between that and the sore knee, the principal complaint. There is no way for another doctor, or a computer, to diagnose any problem with the knee that also involves the rash and fever, by analyzing the notes only. The information needed to do so will simply not be there.

[2]Yes, a "referee" is someone in black and white stripes wielding a whistle, but it also means a person receiving a referral. Look it up.

If, on the other hand, the doctor writes down the notes while the patient tells his story, she'll probably write down the part about the rash and the fever long before reaching a conclusion and a diagnosis that does not really involve the fever and rash. As long as the doctor doesn't go to the trouble of going back and erasing that part of the history, intentionally screwing with the veracity of the record, the findings will indeed remain part of the record and may in the future lead to a different conclusion by another doctor or a machine, from the same data.

The greatly increased veracity and usefulness of the medical record in this last scenario, is a strong argument for having the doctor create the record *while* the patient is telling his story, and *during* the clinical examination, not afterwards.

It's of ultimate importance for the objectivity of the data that the record is created as early in the process as possible, during the actual encounter. The only method that can be usefully employed when the patient is present is typing, or dictating, if the quality of the doctor-patient relationship allows it.



Chapter **10**

The information model

Current systems are built on a model of the clinical reality. What does that model look like? Is it correct?

So, how does the data model[1] look in the EHR, and what can we do to improve on it?

Legacy EHR systems generally provide a couple of major categories under which they sort the documents belonging to a patient. These categories are typically:

- Daily notes.
- Referrals and replies to referrals.
- Radiology orders and protocols.
- Lab orders and results.
- Prescriptions.
- General correspondence.
- Forms of all kinds filled in for the patient.

Let's first look at how these legacy systems relate these categories to each other. Basically, their data model shows the patient as the top, or

[1] "Data" is "information" in a form that a computer can process. "Information" is derived from the "data".

root, element[2], and we then have a list of encounters, and another list of documents. If we're lucky, the EHR system relates referrals, prescriptions, notes, and other documents to the encounters where they were created, but in many systems even this is too much to ask.

Since the notes from one encounter can relate to more than one problem, while documents such as referrals or prescriptions relate to one particular problem (there are exceptions), the relationship between encounters and documents, if it exists at all, is wrong and misleading. Additionally, documents often arrive outside the context of encounters, such as replies to referrals.

In fact, this organization into separate stacks show a frightening similarity to my Belgian predecessor's stacks I described on page 35. We don't seem to have progressed much beyond that, except we now have a significant number of stacks per patient, where there were only two stacks for the entire practice previously.

For the rest of this discussion, just to keep the diagrams simple, I'll limit myself to four stacks per patient, namely: notes, lab, X-ray, and referrals, as in figure 10.1. The first stack with the notes should really be represented as several stacks if we're talking about large unified systems, since each speciality has its own stack of notes, but still largely share the other stacks with lab, X-rays, referrals, etc, with each other. I'm almost starting to long back to simpler times when we had just two stacks in my Belgian practice. Yes, they were larger, but there were just the two, and they made a twisted kind of sense.

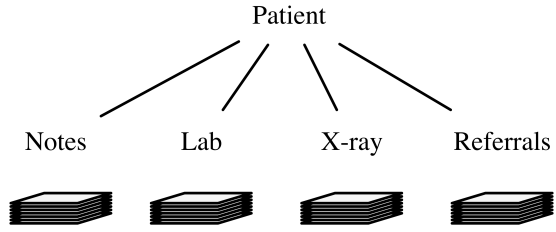


Figure 10.1: The "a few stacks" oriented architecture of legacy EHR systems

The legacy EHR systems have another dimension as well: time. Each of

[2]In computer science, "trees" have their "roots" at the top. Go figure.

the stacks is organized in reverse time order, so the most recent documents or notes are uppermost (again, just like my Belgian predecessor did it). If we expand the stacks in figure 10.1 along the time axis, it will look something like figure 10.2.

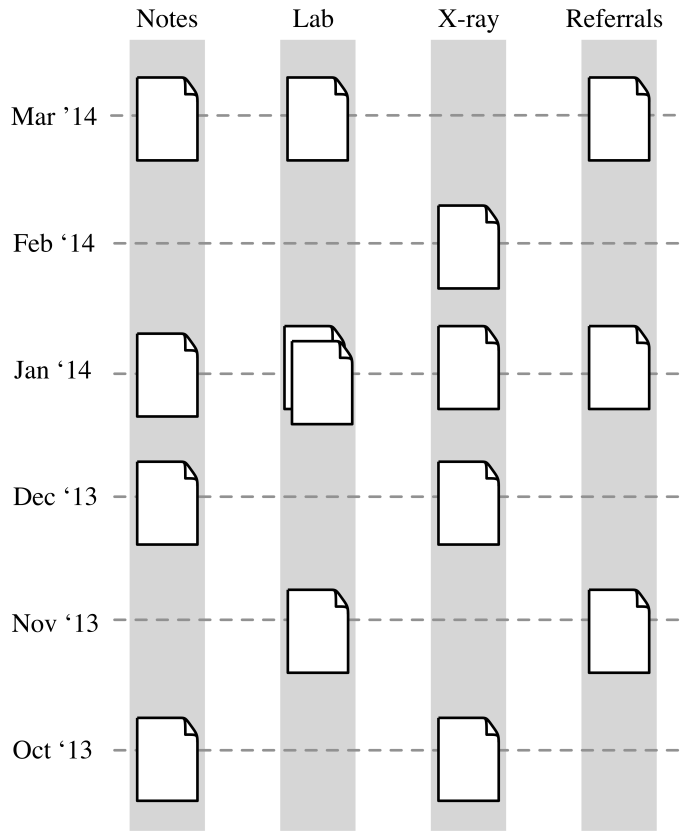


Figure 10.2: Legacy EHR document architecture

There is nothing explicitly connecting any particular documents in one stack with related documents in another stack. In other words, if we as user of this system try to gather all the documents and notes that are relevant to a particular problem, it quickly degenerates into an error prone and highly frustrating stack-walking procedure. Let's assume that the patient records in figure 10.2 contain information on three diseases, one of which is

hypertension, which we mentioned in two notes, and for which we've done two lab orders, a chest X-ray, and two referrals (one for fundoscopy, the other to a vascular lab). In figure 10.3 I've marked the documents relating to hypertension with a crosshatched pattern.

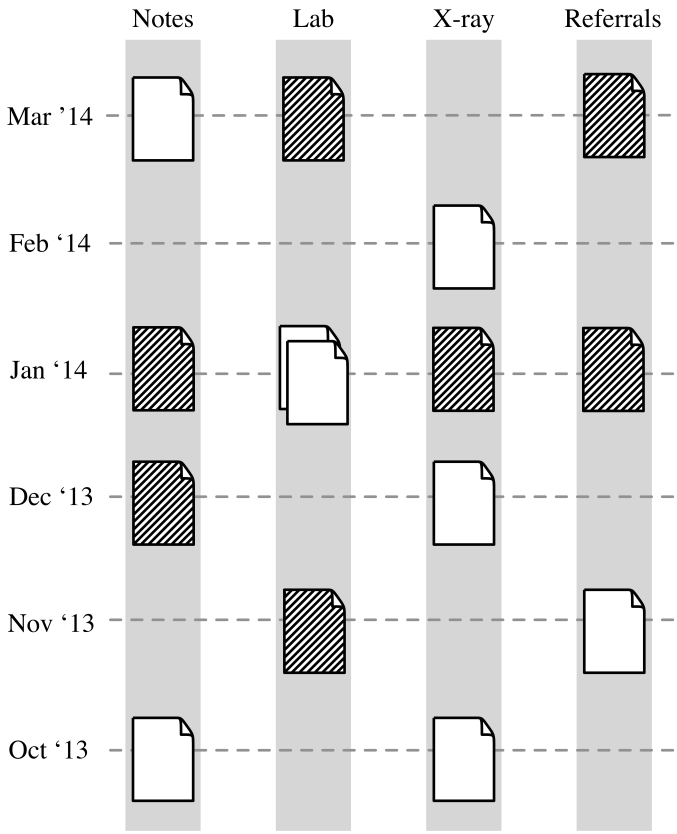


Figure 10.3: Just that one disease is here, here, and here...

The only way to locate the relevant notes is by reading through the stack of notes, top to bottom. Then the only way to locate the other documents that may or may not be mentioned in those notes is by searching through those other stacks from top to bottom. And once you've done that, and have them all neatly arranged inside your head, you just throw it all away again, since there is no way to persist the relationships you just found. The

next time you go through the same process, there's no guarantee you'll come up with the exact same selection of documents from the different stacks, since it all comes down to how attentive you are and how selective, so there is no guaranteed consistency in your view on the record. It's largely a matter of chance. The more documents there are in the record, the less reliably will you be able to find relevant documents.

Having the documents ordered according to the type of document made sense before we had IT system and it was all on paper, since that document could only be in one place[3], and if you had to choose between filing that X-ray among other X-rays, or in a folder "hypertension", it made slightly more sense to file it under "X-rays". Not *much* more sense, just a little bit.

But now, when the number of copies of a document means nothing, it makes no sense at all. An X-ray of the thorax taken during workup for hypertension has its primary meaning as an exemplar of a study of the *hypertension*, not as yet another X-ray. In other words, the fact that it is an X-ray is just an indication of a method, not of meaning. What we actually were looking for was a possible enlargement of the heart, and to determine if that was present, we used an X-ray. We could just as well have done a cardiac ultrasound, or an ECG, but we did an X-ray.

Now, what happens in the head of the doctor when he reflects on the records and wonders if the patient has a cardiac enlargement? What we would prefer to do is go look in some spot labeled "cardiac enlargement", and find the answer: "no, as shown on thorax X-ray", or maybe "no, as shown on ECG", or even "no, as shown on cardiac ultrasound". But with the stack-oriented EHR systems of today, instead I must assume[4] that the possibility of cardiac enlargement has been verified or excluded using *either* thoracic X-ray, cardiac ultrasound, or ECG, and then go searching through each of those three stacks to look for any trace of such an exam[5]. Even worse, if I didn't know of BNP, a new lab test for cardiac failure, I would not go looking in the stack for lab results after this, and would miss it.

What I *really* want to know is if cardiac enlargement or failure has been verified or excluded, and I really don't care how. But what I'm *forced* to do with these systems, is try to read the mind of other doctors, then backtrack from there.

Most doctors simply repeat findings in the notes to clarify how they drew their conclusions, but this is a workaround for an incredibly poor system

[3] Unless we copied it, but that was too much work and too expensive.

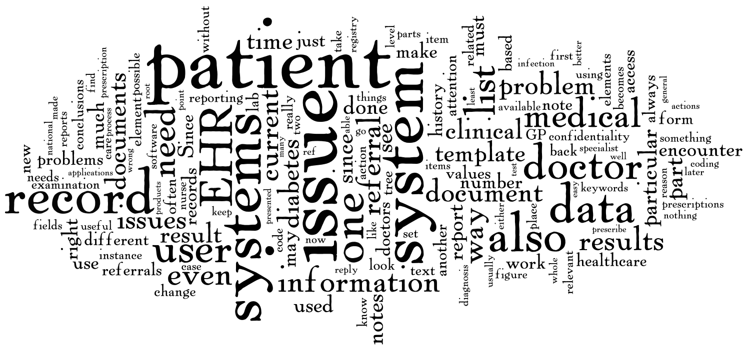
[4] That assumption builds on knowledge entirely outside the records, such as conventions in my area, or that the doctor who did the workup came from the same university as me, or that we both ought to be equally up to date, or out of date, so can be expected to use similar methods, etc. There's a lot left to chance, as you can see.

[5] Unless we cave under the pressure and ask the patient, but that defeats the purpose of the medical record, doesn't it?

design, and it shouldn't be necessary.

Part III

A consistent design



Summary

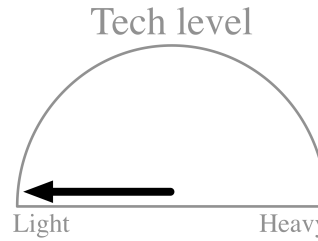
_In the foregoing, we've discussed what the problems in healthcare related to IT are in general, and how current EHR systems attempt, but fail, to solve them. We've seen a number of ways these systems are badly conceived and fail at their task.

In what follows, we'll try to reason out what we need, and how we could go about achieving that. It's clear that we need more than one thing, and that there are a number of aspects of healthcare, that each could use its own solutions.

When we work with software applications in general, there are distinct aspects of the application we can and should discuss in isolation from each other[6]*.*

One such aspect is the composition of the data, what form the knowledge about the patient takes. Is it numerical or textual? Is it structured hierarchically, or by references, or relationally, or a mix of these patterns?

[6] In this discussion, the developers among the readers will recognize the Model-View-Controller pattern, and a bit of Presentation-Business-Data layer abstraction. Good for you.



Chapter **11**

Necessary, but not sufficient

Any solution must satisfy this short list of conditions, else it won't be used. At least, not for long.

There's a number of specific requirements on any system that has a chance of becoming useful and used in practice. Below I'll briefly discuss each of these core features the system must have.

11.1 Effective use

The system must help the user do a better job, do it quicker, and with less errors. The difference must be clear, the payback immediate. The most obvious actions to be taken for a particular issue should be presented first, and all details that can be taken care of by the system should be done automatically.

Finding the right issue, or the right set of issues, to consider, should be highly automated. Just like an experienced ER nurse who knows which instrument the surgeon needs next, the system should serve up the most likely information and tasks before the user needs to go searching for them. The system should be an assistant to the user, not just another chore.

11.2 Context sensitive

The system must be "aware" of the issue at hand. It must adapt and change its content and mode of presentation according to the set of issues that the patient has, and which issues are most important at any particular

point in time. The system must therefore filter out everything but the most important, and most likely, actions or inputs. At the same time, *all* possible actions or inputs should be reachable in some fashion, of course.

11.3 One single system

Legacy EHR systems are a serious block around the ankle for users, and any new system should *replace* it, not add another system to the side of it. The new system should assist the user, *and* produce any documentation that is needed for other purposes. The new system should be the only system the user needs to interact with, and it should form the bridge to any information or functionality it cannot provide itself. There should be one focus, one central management point, one place to find it all, even if some of the content originates in other systems.

11.4 Under the user's control

The knowledge based content (the "issue templates" in my terminology) should *not* be implemented as code. Instead, these must be in the form of files that any user can create or modify for her own purposes. We can't have a staff of programmers and a development process between any changes in medical practice and the actual implementation of those changes in our tools.

11.5 Derivation of issues

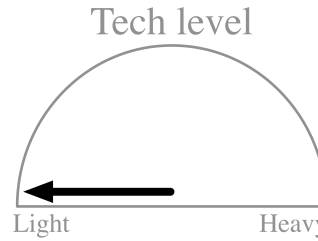
The "issue templates" that determine the diagnostic criteria and therapeutic actions in the tool, should be derived from parent templates and allow child derivations. What this means in practice, is that any user can take a preexisting issue template, adapt it to her own processes and knowledge, and still keep a connection to the template she derived from. This allow the easy spread of new knowledge for a certain "issue" to all users of the original or derived template.

Example: if an institution has derived their own variation of the issue template for "hypertension", their specific issue template should still be updated automatically if a country-wide change of a first choice therapeutic agent is rolled out.

11.6 Cover the full process

The system should cover the full process from initial findings, to a short-list of likely issues, and to guidelines for therapy. There should be full

interaction and feedback from each of those stages to the other stages.



CHAPTER 12

The phases of the clinical process

Observing a doctor working. This is how it looks.

In what follows, we'll discuss the different steps and actions in the clinical process one by one. For each step, we'll first describe the step as such, independent of any IT systems. Then we'll describe how current EHR systems help or hinder this step, and finally we'll go through what future systems could and should do.

12.1 Clinical encounter

The encounter can be an ambulatory visit to the doctor, a phone call, a five minute administrative time set aside to review reports for a patient. It could also be a visit at the patient's bed during rounds. There isn't that much to discuss about the encounter as such, since it mainly consists of a coming together of doctor, patient, at a certain place and time. The encounter could be said to consist of the elements described in the following sections.

How it is

In current systems, the encounter is the main element in the medical record, as far as the notes are concerned. It is reasonably clear what was done

during an encounter[1], but it's much harder to find what was done in relation to a particular issue if you don't know when it was done or by whom.

The encounter is related to a doctor or nurse and a place, so if you're mainly interested in that information, it's there. From a medical standpoint, however, this information is practically useless.

Some systems record the creation of referrals and prescriptions right in the notes, but even then, they rarely link directly to the created documents. In other systems, that information isn't in the notes, so it is left to the user to look for documents with the same date and time as relevant notes to try to piece together what really happened[2].

Even if this all works fine, which it doesn't, it helps very little in finding the notes relating to a particular issue, which is really all that matters.

How it should be

The clinical encounters should still be ordered chronologically in future systems, but that is a view that will rarely be used. A better organization is to have notes, documents, results, and prescriptions, ordered according to issue. If I'm opening up the records for a particular patient, the very first thing I should see is a list of issues, such as "diabetes", "hypertension", "headaches", and so on. If I select one of those issues, I should see a list of plans, notes, documents, results, and prescriptions related to the selected issue.

If I, on the other hand, select a document, a note, or a prescription, I should see a list of issues related to that information element, and through that list of issues, I should again be able to see related information elements, as described in the preceding paragraph.

12.2 Overview of patient history

As a doctor seeing the patient, the main things I want to know about the patient and the encounter are:

- The main problems the patient has or has had over the years.

[1] "Clear" for arbitrary values of "clarity". What is often not clear are which orders were created or prescriptions written.

[2] As if that horrible state of affairs wasn't enough, at least one system I know has a tendency to show the wrong date and time in either or both of those places (notes and documents), making it well nigh impossible to figure out how they relate to each other. That same system, by the way, also records users differently in medication lists and notes, the former with a user login short code, the latter with the full name. In neither place can you see both forms of user identification, so it is *impossible* to match up notes and prescriptions. That reflects some *amazing* dedication to screwing things up.

- Which problem or problems we as doctors need to consider and manage today.
- Which other problems are of importance when considering the current encounter-related problems.
- Which plan is being followed for each of the problems and where those plans are coming from.
- What has been done and decided so far concerning our encounter-related problems.
- What has happened that isn't in the record but is relevant[3]. These are the things the patient should be able to tell us about.

How it is

In current systems, there is nothing but the "history" and the "assessment" fields in the notes to reconstruct the patient's history, unless you count going through all referrals and prescriptions and trying to deduct from those what could have initiated them.

2014-01-10	**Martin Wehlou, Doctor's notes**
Contact	New visit
Reason	Shortness of breath
History	Had similar complaints in 2011, got Salbuterol and improved on that. Last spring also got Pulmicort. Patient still uses both, but is still more affected in cold weather. Pulmicort 200: 1+1 doses per day. Uses Salbuterol every now and then, several times daily. Patient feels pressure over the chest during effort.
Status	BP: 145/85, sitting, left Cor: regular rythm, no souffles Pulm: ves. sounds bilat. ECG in rest: normal. PEF: 225
Assessment	Needs more Pulmicort. I'll order treadmill ECG. Also giving patient an electronic PEF meter for a couple of weeks to collect everyday data.
Referral	Referral sent to UAS, Physiology lab
Prescriptions	Pulmicort Turbohaler 200 micrograms/dose 3x60 doses, renew x2
2014-01-16	**Martin Wehlou, Doctor's notes**
Contact	Followup

Figure 12.1: A typical entry in a classic EHR

History fields are present in each encounter note, so gathering this information implies that you have to scan through all the notes, or all the notes

[3]Note that I carefully avoid saying "since the previous encounter", since all too often even important events that have happened *before* the previous encounter are not in the record, and only the patient can inform us of them.

you can stomach, and mentally assemble it all into a consistent whole. If the record system contains notes from a number of different departments, you can choose to go through them separately or all together, if the system gives you that choice. Assuming you're allowed to scan through the notes for other departments without specific authorization and/or reason.[4]

During most encounters, we simply skip this step, hoping we didn't miss anything important, but sometimes we painstakingly have to assemble a consistent story for some particular purpose, like a referral or a report to the health insurance, and then it's convenient to copy that fairly comprehensive history into the record for later use. The problem, however, is that there is no place to put it except inside the current note, which means that this very useful summary slowly recedes over the visible horizon as later notes accumulate. The only way to keep it current is to keep copying it over to more recent notes, and updating it as you go along, but this is, even at best, a really horrible solution. Some systems have a function to copy old history notes over to the current note, but that's a really terrible idea, since it in reality means a large amount of duplicated information, which doesn't get updated anyway. It simply results in outdated information being added under a current date. You could call it "automated misinformation".

How it should be

If there is a list of issues, aka health problems, over the years, this in itself constitutes most of what we're looking for in a "patient history". The next level down in detail can be found by opening any particular issue and seeing only the notes and documents referring to that one issue. Each issue should contain the *plan* used to diagnose and treat that issue, with a reference to the source of that plan.

For some issues, a timeline can be a good visualization. For example, the long term parameters and therapy of diabetes have something to gain from a well-designed timeline presentation. Other examples are recurring ear infections in children, or vaccinations, or the followup of pregnancies. But there are also a lot of examples of issues where a timeline doesn't make much sense, such as surgery for hemorrhoids or ingrown nails, or a pneumonia. Since the utility of timelines is so heavily dependent on exactly *which* issue we're talking about, it should be a part of the issue template

[4]Since we built up a picture of the patient's history by scanning the notes, it's obvious that we don't really know what we're looking for until we've found it. So we often can't have a known reason for scanning a particular set of records, even though the reason for doing it would be clear if and when we could see those records. Interesting circular reasoning going on here, and the reason for *that* is that the record is oblivious to the concept of "disease" or problem, so it can't present any information in a cohesive and relevant frame.

set for those issues. This has the added beneficial characteristic that the timeline can be designed optimally for each issue type.

There is also the possibility that we'd like to see *all* issues in an overview in the form of a timeline, which could have some utility. In this case, the timeline will be much simpler and not convey much information about each particular issue. Some issues will not be there at all, not in a useful form at least. Just think of congenital diseases like sickle-cell anemia. Yes, each complication of sickle-cell anemia may occur there, but the sickle-cell trait itself can't usefully be represented.

For some issues, an anatomical overview could also be useful, such as for a multiple trauma, where damaged areas and organs can be highlighted. In rheumatic polyarthritis, an anatomical drawing with each affected articulation highlighted can also be very useful.

12.3 Clinical examination

The clinical examination consists of a number of manual actions to check on signs and findings. Examples are:

- Listening to heart sounds.
- Listening to lung sounds.
- Palpating the abdomen for masses, tenderness, percussion[5].
- Examining ear drums, eyes, throat, etc.
- Measuring blood pressure.
- Checking reflexes.

Exactly which clinical examinations we do depends on what we're looking for. If the patient presents with an upper respiratory infection, the examination is directed towards respiratory elements. If the patient comes for a yearly checkup of diabetes, we have another set of clinical examinations we ought to perform. Weed et.al argues [6] convincingly that the initial set of questions and clinical examinations should be very exhaustive, so "what we're looking for" should be seen as very broad and inclusionary.

Every doctor knows how to do the most common examinations like using a stethoscope to listen to heart sounds, palpation of the abdomen, and ear inspection, but other examinations are harder to remember how to do. Examinations for shoulder problems or nervous system diseases can be quite

[5]Percussion: tapping and listening to the sound. This can tell you the size of the liver, fluid or air in the intestines, and more.

intricate and for these the doctor may need reference material to remind him of which examinations he should do, and exactly how to do them and what they mean.

How it is

In current systems we usually have only a template containing a list of keywords. In some systems you can add keywords or sets of keywords on the fly. If you're seeing a patient with diabetes for a yearly checkup, you'd add the set of keywords for just that. But notice that even if you add that set for diabetes, the system doesn't really take note that you're doing an encounter for diabetes, and doesn't save that fact in any useful way, even though that information is clearly available from the user.

Other systems (or often the same systems, but configured differently) instead have a single or a very few templates with keywords. Since you can't easily add in keywords on the fly, these templates tend to be huge, and to include all kinds of keywords that are only rarely used. Often only one in ten keywords are used during an encounter, so much so that the hunting for the right keyword to enter information becomes a significant time consuming task in its own right.

Both kinds of systems usually only save keywords that do contain entered information into the notes, avoiding saving long lists of empty keywords.

As to how information is entered into the keyword data field, there is very little attention paid to that. Most fields are plain text, with a few fields having a numeric type, which in general is more hindrance than help. Some systems allow for graphing of values over time if the field is numeric, but this has very limited utility since it's done for so few fields. At times it could be useful, such as showing a graph of HbA1c values over time for a diabetic, but this has then to be done by switching over to the lab results module, then picking out the HbA1c value and graph it from there. It's not a part of the clinical overview for diabetes, simply because no such overview exists.

Another aspect of the use of the IT system in conjunction with the clinical examination is the office layout. A physician's office usually contains a desk with the computer on it, a chair for the patient, and an examining table. These can be arranged such that the computer can be used while at the desk, or easily reached while examining the patient on the table or gurney. But in some offices, the examining room is physically separate from the office, and there is no computer in the examining room.

If the examination is done in a room without a computer, the doctor is back to either memorizing his findings or taking notes, the exact workflow we try to eliminate. If the examination room does have a computer, we're still out of luck, since we usually can't log in to the same EHR system from

two places with the same user, either because that wasn't a part of the design criteria, or because we need a smart card to log in, and it can only be used in one machine at a time.

How it should be

Since the choice of suitable keywords depends on the healthcare issue we're seeing the patient for, it becomes natural to connect the issue to a list of clinical findings. Since national registry, and communicable diseases reporting *also* depend on the healthcare issue, it is obvious that those clinical findings should also be determined by the issue.

Regardless of the reason the user defines the issue, be it in order to get the list of clinical findings right, or to do reporting, or to maintain a correct patient overview, the end result is the same: a descriptive issue in the history overview *and* the right clinical findings for the clinical notes and the reporting.

The "healthcare issue" is such an obvious integrating element that it is hard to conceive how systems continue to be built without that element in their design.

Not only should the set of clinical findings (or "items" as I prefer to call them) be determined by the issue, but the contents the user enters as values into items should be determined by the item. The user should almost always be presented with a default normal entry value, a series of common alternate values, and the option to enter free text. This not only speeds up entry of normal values, but also mildly suggests standard values such that the record becomes more easily exported for reporting purposes, automatic translation to other languages, and coding systems.

The user should never be limited to predetermined choices, only invited to use them. Whenever he bypasses them and enters free text instead, that effort will always be greater than choosing a preexisting alternative, clearly signaling a defect in the list of available codes. Any such manual workaround of the coding system must be caught and lead to updating of the coding system to fill in the holes.

If the examination room is separate from the office, it has to be provided with a desktop computer mirroring the one in the office. Only one of the machines should be active at any point in time, and the ideal way of achieving this would either be by smart card, which when removed just freezes the screen and keyboard, not logs out the user. Alternately, a proximity activated system using NFC cards or similar could work.

When designing this, keep in mind that the exam room may be shared by several doctors, so when a doctor activates the desktop, it needs to show

the same active session that particular doctor had available moments earlier in his own office.

Another solution entirely could be a portable unit such as a tablet computer, which the doctor can simply carry back and forth and do all of his work on.

12.4 Creating referrals and orders

Referrals are created when the doctor wants to have another doctor examine the patient or perform needed therapy. For instance, if we have a patient with headaches, and we can't say with certainty if they're tension headaches or something worse, we may want an X-ray or MRI of the brain, or maybe we want a neurologist to take a look and give an opinion.

In these cases, we start with a problem we can't solve. We have an idea of the general problem area, like "headache" in this example, and what kind of doctor is a specialist in that.

The next step is to find out where there is a neurologist in our geographical or administrative area we can send the patient to. If there are several, we may want to select them according to the patient's preferences or which one has the shortest waiting times.

Once we decide on which neurologist to consult, we write up a referral with the following elements:

- The actual question, such as "headaches of uncertain origin, please advise".
- A short history with major reasons why I can't figure it out, which examinations I've done, and including any results from MRI, lab, etc.
- Related documents in the form of earlier referrals with replies, and documents included with those replies.

After that, we send off the referral using the communication means at our disposal. Sooner or later, the patient will be seen by the neurologist and we'll get a reply back. (See section 12.8.)

How it is

In current systems, referrals, lab orders, X-ray orders, and creation of general documents such as letters and attestations, are completely separate from the notes and from each other. The creation of notes regarding a particular healthcare issue has no influence at all on the document or referral-creating process. Even if I include the template keywords for diabetes,

nothing in referrals or prescriptions is tuned to diabetes care. Everything is separate and starts from scratch at each use.

If I'm having an encounter with a patient for his yearly diabetes followup and I open up the prescription module, I'm presented with the same choice of thousands of products as always. There is no preselection of products or product classes that are more relevant to my patient. The same thing happens with referrals and orders. The range of presented choices is totally independent of the actual problem the patient has, and is therefore always excessive and confusing.

Each document or referral I need to produce starts out with a mandatory decision on where to send it. There is no smart preselection of relevant addresses, just a full list of *every* address the system knows of. Also, there is no indication of which addressees are interested or able to handle the problem I wish to present them with, except possibly as indicated by the name of the department. For instance, the "cardiology" department is clearly doing something related to the heart, but do they also take children with heart problems, or do those go to pediatrics? Does the cardiology department do ultrasounds of the heart, or is that the radiology department, or even the clinical physiology department? As a user, I'm clearly supposed to know all these rules and exceptions, but I don't. This is something the system should do for me.

Next, I need to fill in a question (in the case of referrals), and even though I've probably already formulated that in the notes under "assessment" or "planning", there is nothing in the system that makes it easy to reuse that information. Worse, in some systems there isn't even a way of copying that information via the clipboard, necessitating retyping it verbatim.

Finally, I need to rehash the main history of the patient, and the background information I may have. Again, the system won't help me copy over history or assessment notes, or even let me copy over other related documents.

After I send it off, there's a significant risk I'll get the referral back, either because it was sent to the wrong place, or it doesn't carry with it all the information elements the recipient requires. Since these referrals are based on the paper model, the electronic referral now becomes useless, *and has to be written all over again*[6], even if the change is minimal or only the destination address needs changing.

[6] Accompanied by piercing pig squeals from undersigned, who *still* can't believe how completely moronic these systems can be.

How it should be

Since we're working with an issue, the most probable scenarios for referrals should be already provided in the system, with a brief synopsis of which circumstances warrant referrals, and what elements should be included.

Since the issue template is aware of the reasons for referrals, it will also know where to send them, and what should be put into the "cause" field of the referral. It is also aware[7] of the required extra information that the recipient of the referral will need, so it includes that as well from the other sources in the EHR. The only thing left for the user to do is to verify and possibly add a personal touch to the different fields in the referral, then send it off.

The chances of the referral being sent at the right time, to the right place, for the right reason, and with the right information, will be hugely increased.

If the recipient, against all odds, still sends the referral back due to misaddressing or insufficient information, the user should be able to update it and send it again, to the same or a different recipient, without starting from scratch. We've had that functionality since forever in plain email, so it's clearly not rocket surgery.

12.5 Creating prescriptions

When we institute or extend treatments with medications, we need to create prescriptions. When we start a new prescription, we do that with the following considerations:

- The problem or disease we want to treat determines the *therapeutic class* of products we want to use.[8]
- We check that there are no *contra-indications* for use of this therapeutic class for this patient. In other words, does the patient have some other problem that precludes the use of this class of medication?
- We check that the patient isn't already receiving any other medications that may *interact* with this medication. If so, either adjust dosage of one or both medications, or avoid giving one of them.

[7] Can't help anthropomorphizing the poor computer. I hope it forgives me.

[8] To see the whole list of therapeutic classes more formally, see the "Anatomical Therapeutic Chemical" (ATC) classification system: http://www.whocc.no/atc/structure_and_principles/

- We determine the *right dosage and duration* of therapy, depending on the patient's weight, and kidney or liver function[9], and which problem or disease we're treating.

- We locate the information on which *commercial preparations* of this therapeutic class that this particular hospital or region supports or recommends[10].

- Finally, we get around to writing the prescription.

How it is

In current systems, the prescription module is independent of the rest of the system. Yes, it *is* related to the patient and the prescribing doctor or nurse, and department, but not to the current issue, referrals, results from lab, or anything else that is a significant concept in healthcare.

This means that when I'm seeing a patient for a middle ear infection, and I want to prescribe an antibiotic, the system presents me with *all* products it is able to prescribe. It is as easy, or as difficult, to prescribe penicillin as it is to prescribe an anti-psychotic medication at this stage. This also means that if I'm not certain of the name of the product I want to prescribe, I have the universe of all possible products to plow through, instead of the relatively limited subset of products relevant to ear infections.

In many systems, you can select products according to ATC grouping, limiting the searchable universe somewhat, but simply finding the right ATC group is largely duplicated work. After all, we just told the system what is ailing the patient, why do I have to keep telling it that in one fashion after another?[11]

Once I've located the product, penicillin V in this case, I need to decide on a dosage. Since the dosage differs according to indication, i.e. it's not the same if the penicillin V is prescribed for ear infections, as it is if it's used for pneumonia, and still different from the dose for sinusitis. Even worse, none of these dosages are in the EHR system, anyway. So I have to find the reference information on the product, usually through a web site. Most EHR systems connect to a reference site and do look up the product

[9] Many pharmacological products are eliminated by the kidneys or broken down by the liver, so if these have reduced function, the dosage may have to be adjusted accordingly. In some cases, we must do lab tests to determine this, measuring either the liver or kidney function, or the actual concentration of product in blood.

[10] Many healthcare organizations make deals with pharmacological companies to get better pricing. These deals result in lists of "recommended products" that doctors are asked (or required) to follow.

[11] Yeah, I know, it's rhetorical. It's because the system is idiotic and doesn't have the concept of "issue" or "disease".

for me at the click of a button, but I have to take it from there, scanning the info, deciding on which table of dosages is applicable, switch back to the EHR and enter the dosage in the form. Then switch back and forth a couple of times to make sure I copied it over correctly.[12]

How it should be

If we have selected "middle ear infection" as the current issue while examining the patient, the issue template will already contain a list of recommended therapies, including a few different antibiotic classes, so I can choose either the most likely to work, or the one the patient is known to tolerate.

When I select one of the products in the issue template, it already comes with a recommended dosage for "middle ear infection", since it's a part of that issue template. It can easily calculate from there according to age and weight, or ask for the weight if it is needed and not yet available to the system. The duration of therapy is also dependent on the issue, and can be automatically proposed to the user.

In some cases, including middle ear infections, the duration of the therapy can be influenced by prior diseases. If the infection is a first occurrence, the recommendation could be five days of antibiotics, while if it is a recurrence within a few months of a previous middle ear infection, it is recommended to take the antibiotic for 10 days. If the EHR is using "issues", it can easily detect that a similar issue was active less than, say, two months previously, and then suggest a 10 day therapy to the doctor.

We can go even further. For the sake of argument, let's assume the doctor changes his mind about the diagnosis a couple of days later when the therapy doesn't seem to work. For whatever unlikely reason, he wants to change his diagnosis from "middle ear infection" to "hairline fracture of the skull", after discovering that the patient fell down the stairs and having done an X-ray study. He then deletes the issue "middle ear infection" and replaces it with "skull fracture". As he does that, the system can then ask him if he wants to continue the antibiotics, since the system knows that "middle ear infection" is the indication for the antibiotic. It's aware of the *why* of the prescription.

If the doctor removes the issue "middle ear infection", all the data entered into the corresponding template remains available in the EHR, it's only the "framework" of the issue template that is removed. Naturally, both the addition and removal of the template for "middle ear infection" is kept in a

[12] And if my institution is stingy, as most large institutions practically always are, I'm stuck with a far too small screen with a far too big EHR window, so I can only see *either* the EHR *or* the penicillin V documentation, not both at the same time, hugely increasing cognitive load and risk for errors.

history log, but as the diagnosis changes, it doesn't need to remain at the front of the visible record.

12.6 Creating the note record

The note record mainly consists of free-form text. One can argue that this is one of both the best and worst aspects of the medical record. It's one of the best aspects since it allows a fully unbounded and expressive description of the patient's condition, wishes, fears, *and* the doctor's assumptions, vague intuitions, and decisions. It's also one of the worst aspects, since it won't allow the computer to anticipate actions, locate suitable support tools, and warn for missing actions and errors.

The note record usually contains the patient history, the clinical examination, and the doctor's conclusions, and open questions, but in this discussion I'll limit myself to the "subjective" part[13], the "assessment" and the "planning" parts of the note, since the remaining "objective" part is discussed separately as "clinical examination" (section 12.3) and "results" (section 12.7).

How it is

The "subjective" ("history") and "assessment" parts of the note are currently free text only, which is really all they should be. The problem is mainly that they're mixed in with the clinical examination, results, and planning parts, which they shouldn't be. Taken together, this creates a mixed bag of free text, and text that really should be more structured.

In many cases, systems structure a few clinical examination fields with drop-downs or numerical masks, but this doesn't really make any significant difference, while only being irritating in the cases we need to add something that doesn't fit the preconceived notions the developer had of valid values.

How it should be

We should preserve both the "subjective" and the "assessment" parts of the record as unstructured text. Humans are unparalleled at describing exactly what is perceived, including the degree of uncertainty and vagueness. This description of impressions, state of well-being, and intuitions is very valuable and should not be hindered by excessive structuring.

When attempts are made to force the user into using a stricter and more well-defined language in these parts of the record, we force the human

[13] I'm using the SOAP terms here, but the same applies to non-SOAP structured records.

user to do the work machines are better at, namely syntactic and semantic accuracy, in order to give the machine the task humans are much better at, namely weighting facts and drawing conclusions. In other words, it places both the machines and the humans at their maximum disadvantage.

The "subjective" and "assessment" parts of the record must therefore largely remain textual and unstructured[14]. Since it is hard, or even impossible, to determine exactly which subjective complaints belong to each "issue"—the patient's well-being depends on *all* his problems to some degree—this part should be common to all issues. In other words, whichever "issue" I'm focusing on, I should always see the same "subjective" history in full, for that encounter.

The "assessment" part should be shared the same way, since the assessment *must* refer to all the issues the patient has, else it's not a good assessment.

Taken together, this implies that the complete data for an encounter include *one* "subjective" field aka "history", a set of one or more "issues" which contain the clinical examinations, results, planning, etc, and *one* "assessment" field.

12.7 Finding results

There are several situations where we need to find replies to referrals, X-ray reports, lab reports, and so on. That could happen during a meeting with the patient, when responding to queries from other doctors, or during planning.

There are three main "angles of attack" when looking for documents, and that is:

- By issue, i.e. everything related to headaches or diabetes. When we try to form an overall picture of the particular issue we're seeing the patient for, this is the most important view. But also, when referring a patient to a specialist for a particular issue, this is also the most fruitful search method.

- By type of document, i.e. X-ray reports, lab reports, replies to referrals, and so on. When considering ordering a test of some kind, it becomes important to see if we have already done that test or examination, or something very similar to it.

- Chronological, that is everything that happened recently, or during a certain time span. This method is more a way of figuring out exactly

[14] Colleague Johan M, and others, beg to differ, thinking that even the patient history should be structured. Time will tell who is right.

what was done by us or someone else lately, to see where things are leading. It is often the last choice in searches if the more directed searches don't work.

We refer to results during encounters, since our conclusions are based on these results, among other things. Everything we base our conclusions on, including these results, should be part of the record at that point.

How it is

During the encounter, with or without the patient present, we retrieve a number of results, some of which will be significant for our conclusions and actions. Since it's important to include in the patient record not only which conclusions we reach, but *why* we reach them, we have to refer to results somehow. In current systems, there's generally no other way to refer to them except as a textual description. There is then no way for a later reader of the record to with certainty determine what results we used, except by painstakingly going through the list of results and matching them up against our, possibly inexact, textual description. I haven't seen any system where the textual notes allowed a direct embedded link to another document in the records, but if that could be done, it would certainly help some.

How it should be

If we work in an environment with "issue" templates, the template itself will have items where it refers to results. If the diagnosis of an infection involves an estimate of CRP[15], the template has the wherewithal to open up the lab reports, then go look for the most recent few CRP values for you. Similarly, if the template contained a possible referral in an earlier encounter, it will now go look for a possible answer to that referral and present it in the context of the same issue.

When the user then uses the issue template to look up a value and selects the value he chooses to be most significant, that choice is preserved in the issue, making it clear for later readers exactly what result the doctor did use to base his conclusions on.

12.8 Receiving results

We also receive results outside the context of a patient encounter. Lab reports and replies to referrals are brought to our attention when they

[15]C-Reactive Protein, a blood test that gives an indication of infection and inflammation.

arrive, regardless of what we're really doing at that point in time. Most doctors set aside some time during each day to go through these new results.

When I see a result from the lab, an X-ray report, or a reply to a referral, the first thing I need to do, even before reading the result, is find out why the tests were ordered, or why the referral was written. I need to locate and assimilate the context in which this examination was required, else I won't be able to understand the implications of the results. Ideally, the result should be presented in a reproduced context, with all the notes, other results, and assessments, that were a part of the thinking when the order was made, regardless of if it was ordered by me or another doctor.

If I don't see that context, it becomes uncertain if I will fully appreciate the meaning of the result and there's a chance I will fail to react to it as I would have reacted if I had gotten the result while being in the state of mind I was when I ordered it. If I fail in that way, the result will not be optimally useful, and the care of the patient will suffer.

How it is

Most systems present new results in a list of "new results" or "unsigned results", paralleling the old paper-based workflow. Back then, we usually got a stack of results dumped on our desks by a secretary, each result attached to the complete medical record of the patient. We were expected to scan the result, do something, then sign off the result in the right lower corner or some other predetermined place on the paper, showing that we did pay attention and from now on, we take the blame if the appropriate action isn't taken.

So now we're in the 21st century, and we *still* are expected to sign off on the result in a very similar fashion. There's no integrated idea of *why* the test or referral was done, or *what action* we could or should take and if that action did indeed result from it all. The system expects one thing, and one thing only: my signing off on having seen it.

The act of signing off doesn't really imply anything. I could act on the result without signing off, or I can sign off without doing anything else with the result. In some systems, simply *viewing* the result sets a flag, in other systems, the user has to click a button or equivalent to "sign off", but there is no relationship between this act and a medically significant action resulting from viewing the information.

Worse, the system presents new results in the context of new results, not in the context I had when ordering the test or referral. This leaves it up to me as a user to go back into the record and figure out why the test was ordered and what to do with the result. Far too often, the test or referral is done by someone else, under assumptions I don't share, with intended actions that were never written down, so I don't know what to

do with the result. This is particularly a problem when I'm not private to my predecessors plan of action, and when *my* plan of action wouldn't have included the ordered test or referral. Due to the lack of context and reasoning, the test or referral result will turn out to not only waste my time, but also not result in any useful new plan of action.

How it should be

If a referral or test is ordered from the context of an issue template, that same issue template gets called up and presented together with the result. In this context, the reasoning behind ordering the test is clear and explicit, and the template also contains suggestions on what to do with the result. What the template does is simply making the plan before, during, and after the test, explicit and clear.

If I'm getting the results of a test ordered by a predecessor for reasons I don't agree with, this comes down to not agreeing with the issue template, i.e. the guideline, my predecessor chose, and that is entirely possible and fair. But then at least I know my predecessor followed a plan, which exact plan that was and why, which gives me all the material I need to decide on which plan to follow in the future in a fully informed and considered fashion.

Having the originating issue template pop up when viewing a new result also saves huge amounts of time, since all the tools for prescriptions, new referrals, letters to the patient, etc, are right there, for *that* issue. It also saves brain power, reducing cognitive load, and reducing the risk of missing details and making mistakes.

The result should only be taken off the list of "new" results when it has been used for other actions or documents. There is no point in having a mechanism flag the result as "seen", since having been "seen" means nothing. The user isn't guaranteed, or even likely, to remember having "seen" anything, and even if he did, being remembered isn't something that in itself helps the patient.

We should make a direct connection between removing the item from the "new" list and the appearance of a new referral or action in the system. These two events cannot be unlinked from each other.

The list should in fact not be called a list of "new results". It's more usefully referred to as a list of things requiring some kind of action, a "todo" list. Note well, that even "ignore" is an explicit action for which the user needs to take responsibility. Doing absolutely nothing is *not* an action, so in that case the item in the list remains in the list until some user takes responsibility for assigning it to an action or an "ignore" category.

If a result or report applies to several different specialities or users, then assigning it to an action by one user, may leave it in the unassigned list (the incoming "new" list, if you will) for any other users considered relevant as receivers of the result or report. This avoids having the result disappear from a user's view even in the absence of action.

12.9 Reporting

Reporting encompasses mandatory reporting of communicable diseases, but also statistical reporting, and case-load related reporting.

To do the reporting right, all the elements needed in the report must be available. These elements often aren't part of the clinical examination and history taking of any particular encounter. To avoid poor reporting or extra work, either all the data must be captured when it's available and used for later reporting, or the reporting should be done during the encounter with the patient.[16]

How it is

Reports and attestations of all kinds are usually implemented in current EHR systems as a collection of forms with pre-formatted fields. Some of those fields are filled in automatically, such as patient demographic data, current date and time, doctor's name, and so on. Most other fields are left empty, such as "history", "clinical signs", "conclusions", "recommendations", etc. These fields are often of a character and intention that is dependent on the use of the document.

The same problems referred to while discussing prescriptions (section 12.5) appear when creating these documents as well. If the user is in the process of handling a particular problem for a patient, only a very small subset of documents is relevant. If the user is treating an infection, reports for communicable diseases may be relevant, but a certificate for a driver's license is likely not. In current EHR systems, the user will still be presented with all relevant *and* irrelevant document forms to choose between, greatly increasing the time and aggravation in finding the right one, while at the same time increasing the risk that she'll choose the wrong form. Which, of course, will come back and bite her later.

Once we get past the onerous locating of the right form, we have to fill it in. This usually consists of going back and forth a great number of times

[16]Sometimes the need to report only becomes obvious once results come in, and in that case, it's likely some elements needed for the reporting were not captured, and we have to contact the patient again for that.

to look up information in the records, then copying and pasting it into the form, occasionally editing the text to look better, or to fit the form field.[17]

How it should be

Again, if the user is working in an "issue" template for the current problem or disease, any document forms that can be relevant for this issue will be listed there, and can be activated from there with a click. This limits the set of possible forms to those that actually make sense for the issue. Nothing hinders the user from going to the full library of forms if he needs something exceptional, but that would rarely be needed.

Since the form is linked to the issue, it is easy to link the contents of most form fields to existing items in the issue template, eliminating most of the copy and paste work. Only very few fields will ever need to be entered entirely by hand.

Having the form created from the issue also means that this relationship is preserved. It will be trivial to see in the future that a certain document or report has been created for this issue, and conversely, what issue was the reason for the creation of a particular report.

12.10 Reporting to national registries

National registries are maintained for selected diseases, such as heart failure, diabetes, hip replacement surgery, and more. These registries can be useful for a number of statistical purposes, and in selected cases may even contribute to scientific knowledge.

Often reporting is done after the patient encounter by reading the medical record and extracting the reportable data from it. The problem that often occurs is that the right data was not collected during the patient encounter, necessitating contacting the patient again, or submitting incomplete report forms to the registry.

If the reporting can be done in the presence of the patient, or forms (paper or electronic) are provided for the doctor to use during the encounter, the frequency of missing data can be reduced to practically nil.

[17] As I'm writing this, I feel my rage boil up again at that one system that for years didn't allow switching back to the record, then didn't allow copy and paste, and *still* even with copy and paste can't handle an excess of text for a form field, simply refusing to paste in that case. Oh, and it won't allow copying less than a full field from the notes either. This is the way you turn plain bad software into an epic fail.

How it is

In current systems, there is either no help in creating reports to national registries, or forms are provided in the notes with all the data fields needed for these reports.

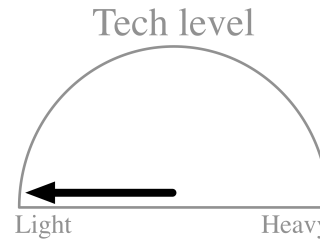
Some systems have fields in the EHR that clearly makes it easier to create complete records with all the required data during the encounter. Better systems also eliminate duplication of entry between the normal clinical examination and the reporting form.

These forms are then checked and completed by a nurse or doctor, then sent to the national registry in binary form (hopefully), and anonymized at arrival. In some systems, the patient identification is transformed using some form of hashing function before being sent to the registry.

How it should be

If all required fields for the national registry are part of the clinical issue template, the creation of a report is a simple question of extraction of those data values, and subsequent packaging into a reporting data packet.

Before transmission, the data should be fully anonymized and provided with a one-time-only patient identifier, that can only be resolved to the right patient by a designated third party. The technology to achieve this is fairly straightforward, but a precise description is outside the scope of this book.



Chapter 13

The full medical process

From illness to cure, every step of the way.

In figure 13.1, we see the steps that together make up the care of a patient. It's very important to keep the full process in mind when discussing any technology or method that has bearing on one of these components or steps. Let's work through the diagram now.

Presentation. This is the patient presenting his problem in the most basic form. It could be after an accident, or it could be a patient complaining of headaches, or fatigue, or loss of weight.

Initial inputs. The initial inputs consists of the set of questions and clinical signs and symptoms that should be asked for all patients of a certain category. All patients presenting with a general medical problem such as headaches or fatigue, should be asked the same basic set of questions, and have the same basic set of clinical signs and symptoms checked. If a patient presents with a stab wound from a knife, the set of questions will be different, much smaller, and more to the point.

Database. The database is the set of *answers* to the initial questions, signs and symptoms. (I'm using the term "database" in the sense used in Weed et.al [6].)

Matching. The set of values in the database is now matched against the set of "issues" in the system, to come up with a shortlist of probable and possible explanations for the patient's complaints.

Candidate issues. This is the shortlist of probable and possible issues that could be relevant to the patient.

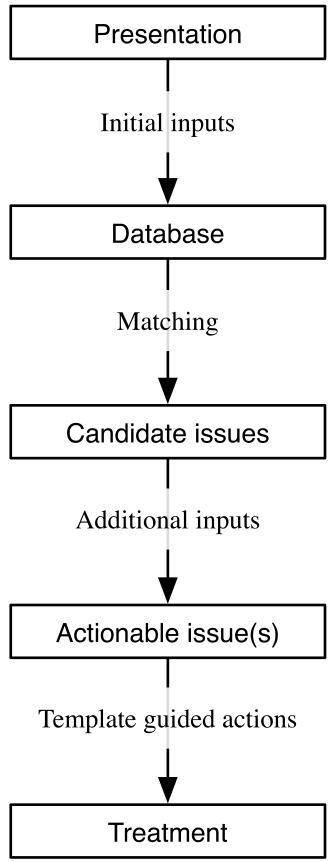


Figure 13.1: Steps in the complete care of a patient.

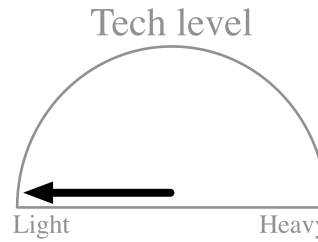
Additional inputs. The candidate issues are, as are all issues, each linked to a set of patient history questions, clinical signs, and findings, some of which are in the set of initial inputs, but some of which are not. These latter, "new", inputs are now added to the list of inputs, and can be answered by the user and/or patient.

Actionable issue(s). The additional inputs can result in elimination, or at least downgrading, of some candidate issues, and introduction of new candidate issues. The process repeats until there is a stable set of issues which we can act on, which we call "actionable issues". More than one issue can often be the explanation at any particular stage due to imperfections in the methodology. Also, patients may have more than one issue at the same time.

Template guided actions. Each issue template contains links to the related issue items (complaints, clinical signs, and symptoms), but also the actions that can be taken, such as referrals, orders for X-rays and lab, prescriptions, and more. Each of these actions are tailored to the issue itself, so that executing these actions will be much simpler and much less error prone.

Treatment. Once the definite issue or issues have been determined to a satisfactory degree, the actual treatment is chosen from the set of possible treatments available in the issue template (or templates). The choice of exactly which treatments to perform, and exactly how, is something the patient and the doctor do together, where the patient is the primary decision maker. The issue template advice, however, presents a complete list of options to enable an informed choice, while also allowing a more secure and complete execution of the chosen alternatives.

To make this a living, evolving system, it must be possible to develop each issue template in isolation, and still have the totality of issue templates and input definitions behave as one unified system.



Chapter 14

The real requirements

After removing all the fluff, what's left?

With the lessons of the previous chapters in mind, how do we formulate the *real* requirements for an EHR system, that includes the features we need, while not needlessly limiting the solution space? I'll touch on the areas I think are most important.

I'll formulate these requirements as "awareness" key points, underlining the importance of keeping the doctor or nurse informed of the right things, not *how* that information is brought across. It does not matter in the least if it is presented on paper, on a screen, by needle pricks on the back of the hands, or by telepathy. Those are all possible (or impossible) implementations fulfilling these requirements, and it's up to the fantasy and ability of the architects and designers to choose the method.

In the same fashion, input to the EHR system can take any conceivable form, such as typing, speech, gestures, telepathy[1], interpretive dance, or music, but the main point is that it is up to the architects and designers to invent suitable methods. The key requirement is that there *is* an input, not *how* it's done.

14.1 Awareness of issues

When seeing the patient, the doctor or nurse must be made aware of all issues the patient has or has had that can have any relevance to the current

[1] You wish.

encounter. The doctor or nurse must also be made aware of the issue that is the subject of the current encounter.

An "issue" consists of a short description of a medical problem or fact that forms the subject for diagnosis or treatment. It should be distinct enough to allow identification in the literature, as indication or contra-indication for medication, and as the basis for public health reporting or statistics. It should be of a form that is, or could be[2], available in standard coding systems such as ICD–10 or SNOMED CT.

As the user is made aware of the issues, the user should, for each of those issues, be made aware of all the diagnostic and therapeutic steps that have been taken in relation to that issue *and* the reasoning behind it. That "reasoning" includes at least the planning and what scientific basis that planning has, including both diagnostic and therapeutic planning.

Any "summary of care" documents belonging to any issue should also be clearly presented. There should be a way of finding more details than the summary of care presents, but those details could be provided through other means.

Any documents forming the basis for conclusions in summary of care documents, or in responses to referrals, should be included directly or indirectly in these summaries or responses. See appendix A where I discuss referring to sources in the document-tree design, on page 167.

14.2 Awareness of patient history

The doctor should quickly and painlessly be made aware of the patient's history in general, i.e. those aspects not tied into a single healthcare issue, but more related to the whole patient. Things like general well-being, ability to lead a normal life, and the major obstacles to that, including social and financial. This history should not be fragmented and contradictory, but be presented as a consistent whole, where not only the different aspects, but also the evolution over time is clearly shown.

14.3 Awareness of planning

The doctor needs to be made aware of the plans used in diagnosis and treatments, what these plans consist of, the sources they are derived from, and how far along in these plans the diagnosis and treatment have come.

These plans must be explicit and detailed enough so they can be compared with other plans and "current best practice", both by the doctor and the patient. The sources must also be explicit enough so that it can be

[2]If it could be, but isn't available in standard coding lists, one should take possible mechanisms into account to add the issue to these coding systems.

verified that these plans are not invalidated by any sources having been retracted or superseded by more recent science.

It should also be clear exactly why these particular plans were selected for this patient, if it was due to the location, the patient's own characteristics, or a preference by patient or doctor.

14.4 Awareness of outcomes

It should be clear from the overview what the status of issues is. Are they active, and if so, who is responsible for this issue (doctor, provider, institution)? When was it last managed, and is there reason to think that it needs more attention? Has it been forgotten about and left without action for too long? Or is the issue resolved, and, if so, how (briefly)? Or are there outstanding results or responses related to this issue that need handling?

14.5 Ensure action

When results or replies to referrals become available, the user should be made aware of these. The system should keep showing these results or replies as "open", or "not yet attended to", until effective action has been taken, and these results have become an integrated part of further decision making. The simple viewing of the result should not be regarded as effective action.

14.6 Issue-based management

The patient management should be based on healthcare issues in such a way that the system adapts to the actual problems being managed.

It should be easy to locate a healthcare issue and activate it. Once activated, the system should present the user with the most frequently used modules and actions that are applicable to that issue. The system should also help the user navigate through the management in the most optimal way, while also helping the user avoid forgetting steps and remind the user of alternatives, possible other explanations for symptoms, and the expected waiting times and resource costs.

The navigation through the management of a healthcare issue should be defined primarily by the healthcare workers themselves, with the emphasis on correctness and completeness from a medical perspective. The navigation structure and the recommendations are primarily a medical responsibility and form a set of guidelines, but non-medical management can be consulted while setting up these guidelines such that unnecessary delays and costs can be reduced.

The management of a healthcare issue is defined in what I call issue templates. Such a template contains guidelines for diagnosis and treatments in a checklist form, referral addresses and criteria for referrals, recommended diagnostic means, recommended therapies, and patient materials.

The system must be designed in such a way that these issue templates can be used nationwide, or be customized for each region, institution, provider, or even patient. There should be a provision for derivation of issue templates from other issue templates, such that the relationship is maintained and it remains clear which templates are local variations of which other templates.

When the management of a healthcare issue is modified due to new knowledge or new resources becoming available, the corresponding issue template should be easy to modify by healthcare workers themselves. At the same time, users employing templates either identical to the changed template, or derived from it, should automatically be given the opportunity to update their own templates, and therefore management, of that healthcare issue.

The issue template is designed to formalize the management of a particular healthcare issue, so it should be made available to the patient, allowing him to judge for himself, or with the aid of another doctor or patient advocate, the contents of the plan, and therefore the quality, of the provided and planned care.

14.7 Recording of history

The subjective "history" element is by its nature free text. There is no useful way of structuring this according to some predetermined syntax. Its main function is to relate what the patient experiences with as little transformation or interpretation as possible. There's good reason to even let the patient largely write the history part himself.

Since this element relates the experience of the patient as a whole, it cannot usefully be assigned to any particular subset of issues the patient has, so it belongs to no single issue in the record. It can be a part of any or all issue templates, but the content of the element will always be the same across all issues for any particular encounter.

All this implies that the history part of the record cannot be made available in multiple language versions, unless there's a translation tool provided with the system.

14.8 Recording of clinical examinations

Clinical examinations are not in essence different from other items in an issue template. Each clinical item consists of a prompt and optionally a set of preselected values.

When designing the template, one of the preselected values can be indicated as a "default" value, and the system should have a convenient, easy to remember, and fast shortcut or gesture that activates the default choice. Going through a clinical examination, or any other kind of questionnaire, that consists entirely of default entries, should be optimized for speed. (In our *iotaMed* implementation of issues, the user double-taps an item to select the default value.)

Keep in mind that the absolute majority of clinical examination items, even in a very ill patient, will be normal, i.e. correspond to the default value, so the system should optimize for this path.

14.9 Don't lead me up the garden path

Some legacy EHR systems have warnings for potential errors. The most commonly implemented is the pharmacological interaction warning, which I'll take as an example of everything that is wrong with how these systems are built.

What happens is that you are first presented with a list of *all* medication products you can prescribe, then you're allowed to select one, and only after that will the system come back and tell you it's a bad choice[3]. Now, this just serves to make us hate the system.

What the system *should* do, of course, is only present you with the products that are relevant to the issue you are in the midst of working through. And even *if* it would present pharmacological products that have a contra-indication or interaction warning attached to them for this particular case and point in time, that warning should result in a flag that is visible at the point in the issue where you would prescribe that product. In other words, don't waste my time selecting a product the system *already knows* has a warning attached. That warning should be visible *before* selecting the product.

Products that belong to the normal arsenal for the treatment of this issue, but that should not be prescribed due to contra-indications or interactions, should still be displayed, albeit with a flag of some kind. If you hide these products, you'll only confuse the user, since the disappearance of

[3]Adding insult to injury: many, if not most, of those warnings are so wrong they're just a waste of time.

a well-known therapeutic product from the issue template could be misinterpreted by the user to mean that the template is defective, or the product withdrawn. Also, the interaction or contra-indication warning is just that, a warning, and the user may need to prescribe that product anyway, after considering the alternatives and the risks.

The user should always have the ability to choose any product from the total list of existing products, but that choice may reside one or two levels deeper in the interaction hierarchy, since it should rarely be needed if the issue template is well designed.

14.10 Confidentiality

The system must allow setting access limits on issues. This confidentiality flag should limit access in several levels to groups, or roles, of users. The restrictions should not be too detailed, since that makes the system hard to manage. A bare minimum should include the following levels:

- Accessible only to the creating organization (the department handling the problem).
- Accessible to the above, plus designated individual doctors/nurses, or designated other organizations (care centers, departments).
- Accessible to all authenticated staff.

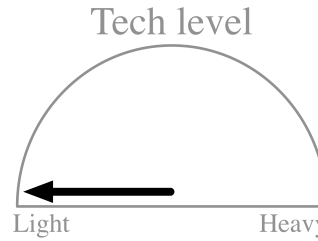
Since the access restrictions are set on the issue, not the department, some issues can be confidential, while other issues are not marked as such, even though both kinds can originate in the same department.

The confidentiality setting also includes all medication, all referrals and lab reports that are originated from the issue in question.

Since the presence of an issue, or any result covered by the confidentiality setting of the issue, can form a warning or contra-indication when another doctor or nurse prescribes medication or orders diagnostic tests or treatments, the system must still be able to warn for that. If the initiating doctor or nurse are excluded from viewing the issue, one of the following actions can be prescribed by policy to occur:

- The originator will be warned about the contra-indication, and told what it consists of.
- The originator will be warned, but not told what it consists of.
- The originating organization for the hidden issue will be told there is an attempt to prescribe an action that could be dangerous, and this organization will have to resolve the issue somehow.

Other combinations could be possible, such as involving the patient, or involving only the actor that created the confidentiality flag in the first place.



CHAPTER 15

How active should the software be?

Who should run the show? We or the machines?

When the software contains information about diagnostic criteria and at the same time has the clinical data pertaining to the patient, it is tempting to assume that we should let the software draw conclusions from the clinical data and establish diagnoses. Making decisions is what computers are for, after all. But designing for that would be a mistake.

15.1 The keyhole effect

If we let the software make diagnostic decisions and pose questions according to past answers, it will lead us through a series of questions and answers that appear to the user in a sequence according to its programming. It proceeds along a path through a flowchart. This hides the overview of the process from the user, giving the user just a "keyhole view" into the exact questions and parameters that the software deems interesting at any particular point of the decision process.

As a user, I'm inclined to game a system like this, simply to be allowed to view the different branches of the decisions tree that are hidden depending on particular data input. The whole thing quickly degenerates into a charade of false inputs just to make the desired information come up on the screen. Putting in false information into the record for this reason, or for *any* reason, is a really bad idea that is bound to come back and bite you later.

15.2 The indiscriminate criteria effect

When the software chooses clinical data to match to criteria, this is often done quite mindlessly, leading to wrong conclusions. For instance, the criteria for the diagnosis of diabetes is, among other things, "two consecutive capillary glycemia values of 7.1 mmol/L or higher". If we let the software make that diagnosis based on the series of glycemia values in the records, it will make that diagnosis in many cases where a doctor would not, and vice versa. Many of these values may be non-representative, the result of other influences that the doctor knows about, but which the software doesn't. Also, if the software does *not* make the diagnosis, but the doctor does, it will probably force the doctor to falsify clinical data to make the software behave as he wishes it to behave.

If the doctor is made to fiddle with data to make the software draw the right conclusions, the set of clinical data becomes suspect. The right role of the software is to present clinical data and criteria together in an easily digestible format, aiding the doctor as he draws conclusions and makes decisions.

15.3 The disempowerment effect

If the EHR system is enabled to make decisions that used to be taken by doctors, healthcare providers may see this as a way to reduce the dependence on doctors, thereby increasing the capacity for healthcare, or reducing the costs for doctors, or both. There is nothing wrong with these goals, but there is a significant probability that doctors will see this as disempowerment and refuse to delegate that power to the IT system.

Since getting the cooperation of doctors is crucial to any successful automation project, you should carefully consider if it is worth it to pursue a transition plan that involves taking power away from doctors by force. It is probably more prudent to have that transition occur further in the future, as it will undoubtedly sooner or later be the case, and have that transition be initiated by the doctors themselves.

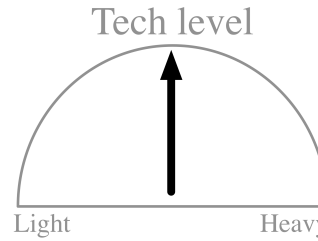
15.4 Nurse vs doctor domain expert

The leading user influence in EHR development is usually either a nurse or a doctor, and it seems that the difference between the outlook of these two professional groups is severely underestimated.

Nurses usually work in a process oriented workflow: start from the top and work your way to the bottom of the list. Typical examples is preparation for operations, postoperative care, post anesthesia checks, etc.

Doctors seldom work in a directed workflow, but tend to work with a list of things that should be considered or done, and where the order, or even completeness, is of secondary importance.

A software system designed by nurses will have a fundamentally different workflow from a system designed by doctors. There is nothing wrong with this, unless you let nurses design systems for doctors or vice versa.



Chapter **16**

The issue oriented record

It's not all doom and gloom. Issue orientation to the rescue!

In an earlier chapter (see chapter 7) I described an example of a current EHR system. You won't find any description of knowledge support there, because there is none. Nil, nada, zip. I also described in chapter 8 how much we actually need knowledge support in our daily work as doctors. What we *don't* need is another tool to use along the medical record, doubling our interactions with the computer, but a tool that *replaces* the medical record as we have it now.

The solution is to take guidelines and other knowledge based support tools and adapt them so they become both interactive, and a mechanism for recording both history and actions. I'll describe one way of doing just this.

16.1 Diabetes, old style

As an example, I'll use the yearly followup of a diabetes patient. There are a number of things the doctor should check, and a number of decisions to take. None of these things are very difficult or far fetched, but you have to think of them, and you have to do them.

For instance, you should check the heart and lungs, and take a blood pressure. You should check the weight, the glycemia values, creatinine, lipids, and the albumin/creatinine index in urine. You should also ask the patient how many hypoglycemia episodes he's had the last year. Every three years, you should have an ophthalmologist check the eyes for complications.

The treatment of diabetes type 2 is primarily tablets, and the first choice is currently metformin. If that isn't enough, we can add glimepiride or glipizide.

There's a lot more things like that you have to think about, but the above considerations are enough for my purposes here, so I'll stop now.

Using a classic EHR system, the doctor has to remember the above items, or look them up in a guideline or book as the patient waits. If the doctor relies on her memory alone, she won't know if the recommendation for, say, metformin as a first choice has changed due to new discoveries. She'll just plow on prescribing it as a first choice forever.

In the classic EHR, the doctor may have a template for diabetes yearly followup, with the keywords "Heart", "Lung", "Hypoglycemias per year", "Medications", "Conclusion", which may remind her of what general areas she should spend some thought on. But that is all. If she wants to make sure the eyes have been checked on time, that's something she'll have to remember to do. And to do that, she'll have to switch over to the referrals module and then wade through it to find any ophthalmology reports about diabetes followup[1].

When the doctor orders the lab examinations through the classic EHR, she has to switch over to the lab module, then remember to order the right lab tests for a diabetes followup. If she's lucky, there's a preselected group of lab tests for that purpose, but she has to remember to look for that. Back in the notes part of the EHR, there is no automatic indication in the notes that she did in fact order the lab tests at all.

Towards the end of the consultation, the doctor needs to consider any needed therapy changes. Once having thought of that for a while, she ought to write down a note in the record about this, mentioning if the therapy can remain unchanged, or if it needs updating, and if so, how. If the therapy needs additions or changes, the doctor switches to the prescription module, then edits the prescriptions accordingly. There's no guidance whatsoever from this module in what is first choice, or second choice medications for diabetes. Or for anything else, either[2].

[1] The eagle eyed reader would maybe notice a dubious assumption here, namely that any examination for eye complications must have been done by an ophthalmologist, and it must have been done due to a referral, so that's where we have to look. What we should have been able to do is find the actual examination results we're looking for, *regardless* of if it was done through referral, or which speciality performed the examination. The problem is the misguided conflation of *what* was done with *who* did it, that is so pervasive in current systems. The same bad assumption causes confidentiality in current systems to be based on medical departments instead of on the actual health care issue.

[2] Some EHR systems indicate "preferred" choices graphically, but these "preferred" choices are determined on price, availability, and deals with suppliers, and have nothing to do with preference from a medical perspective. As always, the only preferences that seem to matter enough to make it into features, are administrative preferences.

I'll stop here, but the rest of the consultation goes on the same way. Everything medically significant, any real decisions and conclusions, only happen in the head of the doctor. The EHR system just sits there and allows recording of anything the doctor decides. It assists in sending off prescriptions and finding documents, but it basically has no idea what is going on and why. It has as much to contribute to the consultation process, as a word processor contributes to the jokes in a Terry Pratchett novel.

16.2 Diabetes, new style

The new style EHR tool adapts to the healthcare issue and the point in time. For each healthcare issue, such as diabetes, it has several variations of templates, each of which I call a "block".

For diabetes, there's a block for the diagnosis and workup, used when the patient initially presents with the problem. There's a block for the yearly followups, which is used over and over again, and there may be a block for the diabetes nurse, and for the foot care specialist, and so on.

The block for diagnosis and workup is intended to be used once only, and that is when you decide if the patient does indeed have diabetes or not. It also contains the initial examinations and treatments for diabetes. It can be used again if the full workup couldn't be done in one consultation, though.

Items you'll look at during the workup are possible infections, dietary habits, weight and weight loss, vascular status, and more. The actual criteria for making the diagnosis are also evaluated in this phase.

The next type of block is the yearly followup block, where we need to go through parameters that indicate how well managed the diabetes is, and if changes of therapeutic regimens are needed.

After a few years, the patient record will contain several diabetes related blocks, typically one for the workup, and one for each year of followup (see figure 16.1).

The workup block

The workup block, as all other blocks, contains a number of prompts, or questions, that can be answered with plain text, number, or one of a series of predefined choices.

We should ask the patient about abnormal thirst, excessive urination, sugar urge, infections, and more. We should also check on the patient's general condition, skin turgor (an indication of hydration), sores, candida infections, or other complications. We need to review lab values for electrolytes, glucose, ketones, and more. If lab results aren't available, we need to order a standard set of lab tests.

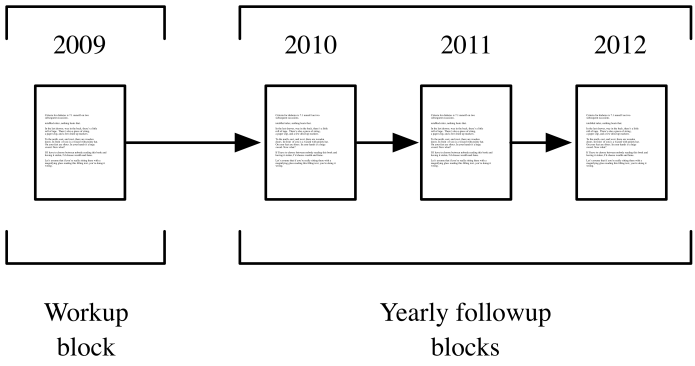


Figure 16.1: The series of issue template blocks in a typical diabetes patient.

Figure 16.2 shows part of the workup block for diabetes, where you can see the relevant input fields for weight loss, thirst, polyuria, blood pressure, and more, already provided. All these are gentle reminders of what to ask and examine with a diabetes patient at presentation. But you also see the lab tests that should be ordered, and right next to the list there's a button to tap which automatically orders just those tests, and then helpfully adds a marker showing they have been ordered. There's no switching back and forth between the EHR system and a guideline system, or between the notes and the lab module in the EHR system.

In figure 16.3 we see another part of the same workup block for diabetes. In this part, we're reminded that the diabetes diagnosis depends on two consecutive values of blood glucose, and by tapping those fields, we get a popup allowing us to select the most significant value from the available lab reports. Again, no hunting around different parts of the EHR system to collect the relevant information. Instead, we get it presented exactly at the point we need it. By selecting one of the values as most significant, we've also inherently documented that choice, eliminating a bit of note taking at the same time.

If we need a reminder on how high the glycemia value should be for a diagnosis, and if we run the system on a tablet, we can tap the info button on the left, and get a brief description of the criteria. Having "information at your fingertips" doesn't get more real than that.

In the lower half of figure 16.3, you also see a set of useful referrals. Since these are part of the template for diabetes, the referrals are already largely filled in with the right destination, the right questions, and the right

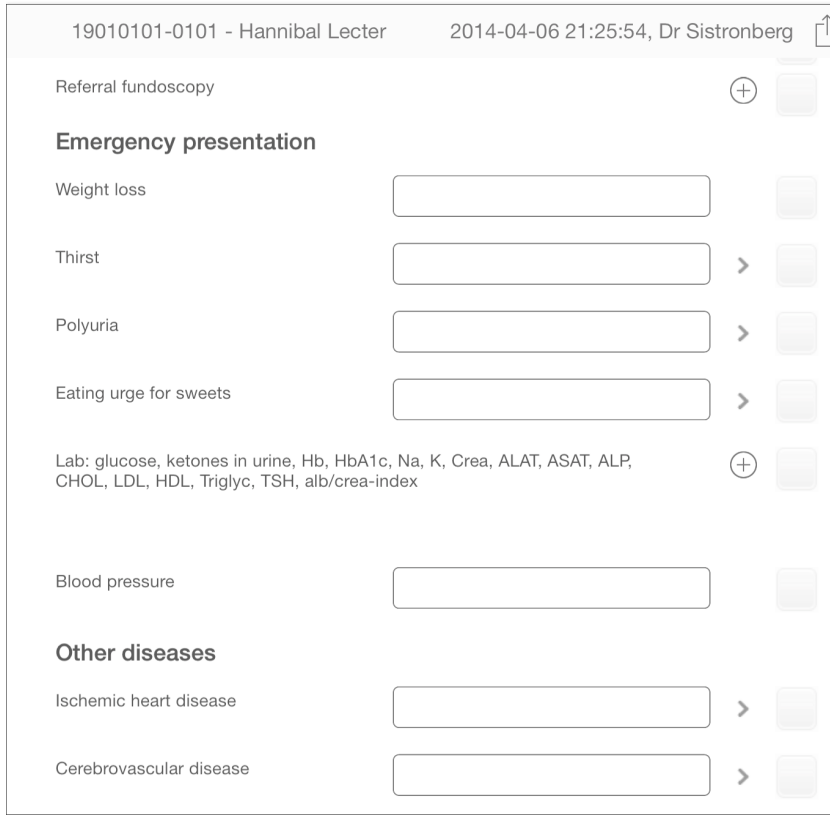


Figure 16.2: Part of the workup block of the diabetes issue template.

included information, as shown in figure 16.4. By touching the ⊕-button to the right in figure 16.3, the referral form is shown, and once sent, it's also marked as completed in the main workup block as in figure 16.5.

Therapy

When we get to the part where we prescribe medication, we use the same principles. Right there in the issue template block we are informed of the first choice medication, with a brief mention of recommended dosages *for this disease* (see figure 16.6). A number of medications are used in different diseases, and for each of those applications, different dosage tables are often used. If the medication recommendation is made part of the guideline in the form of an issue template, that recommendation will be accurate for

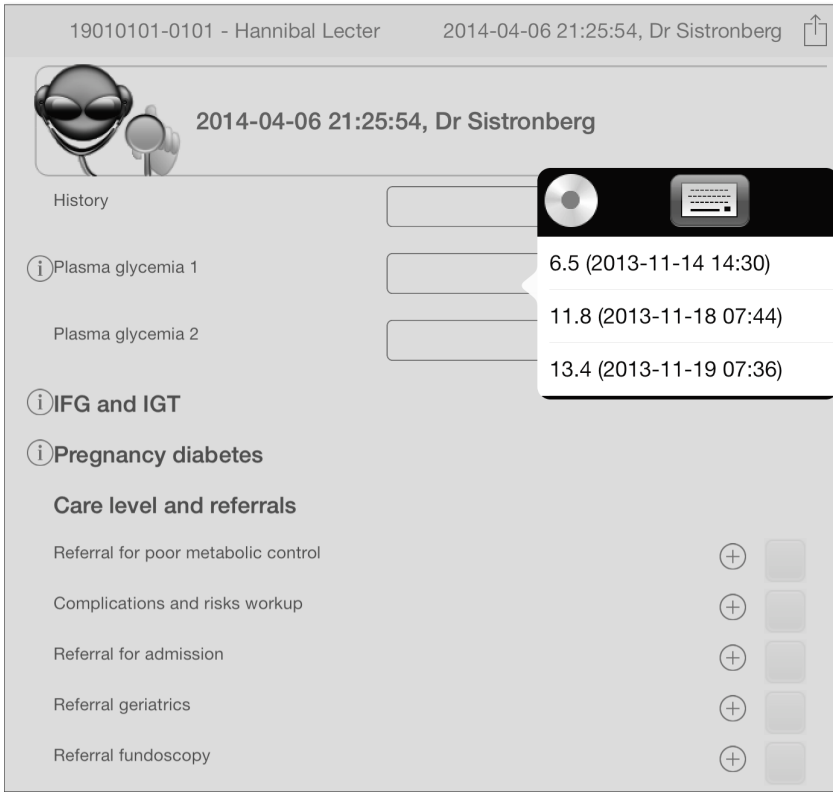


Figure 16.3: Another part of the workup block for diabetes.

the issue at hand, without having to look up the same disease yet again in another system managing prescriptions.

After clicking the ⊕-button in figure 16.6, the system now presents the user with a set of alternate recommended doses for this medication, making the selection very simple. See figure 16.7.

Back in the block, we clearly see the flag to the right of the medication, indicating that a prescription was made (figure 16.8). The square button to the far right is now enabled. Touching that button brings up a history of prescriptions and dosages for that one medication.

With this system, the EHR is "aware" of what disease you are managing, so it can present the relevant medications, and assist in making the right choices. In other words, the most common and recommended courses of action require the least amount of

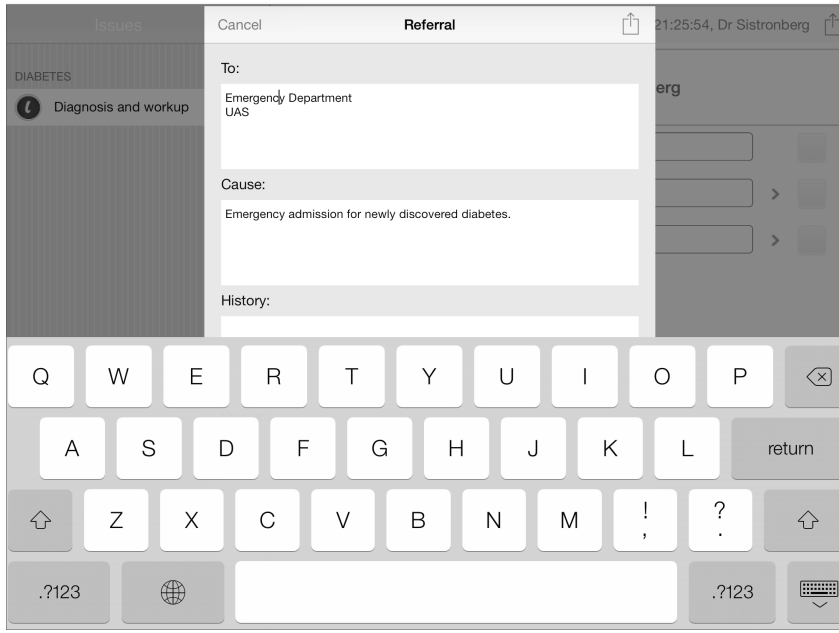


Figure 16.4: Filling in a referral and sending it.

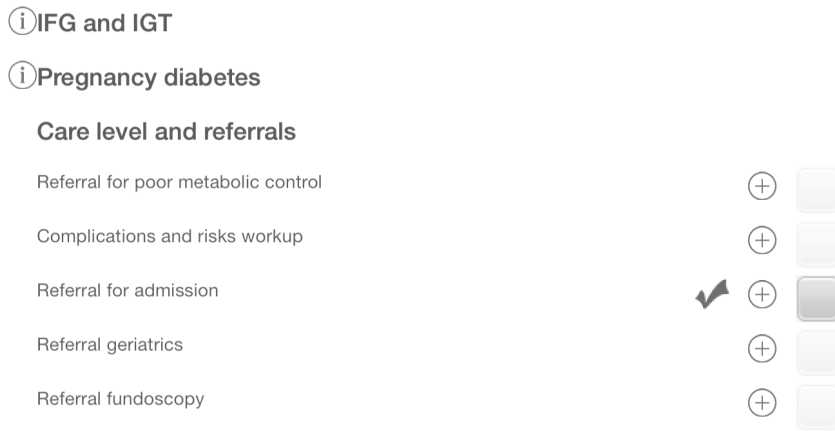


Figure 16.5: After sending a referral, it is flagged in the block.

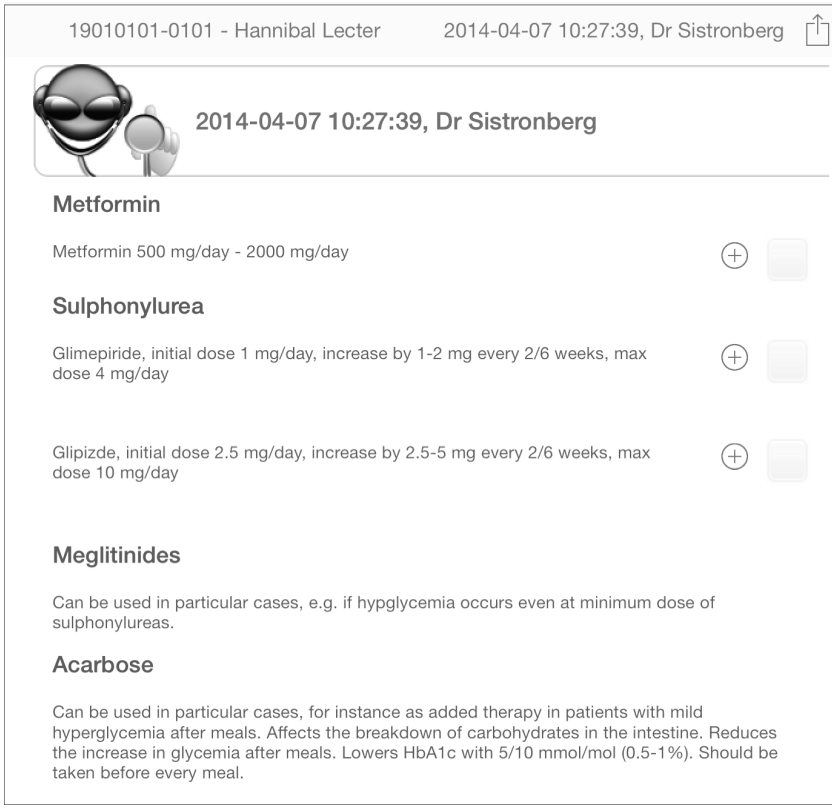


Figure 16.6: The therapy block of the diabetes issue template.

switching around and mental work. The risk for oversight and errors is greatly reduced, as well.

16.3 The template structure

With any guideline-based system like iotaMed[3], we have to structure the templates such that they can easily evolve in time as knowledge about the managed diseases improves. We also have to provide for variations in guidelines depending on country, locality, provider, and even at the level of the individual patient. Since an issue template is a digitized form of a plan, it must allow adaptation of that plan for a whole series of reasons.

[3]I'm being disingenuous here; there are no other systems like iotaMed.

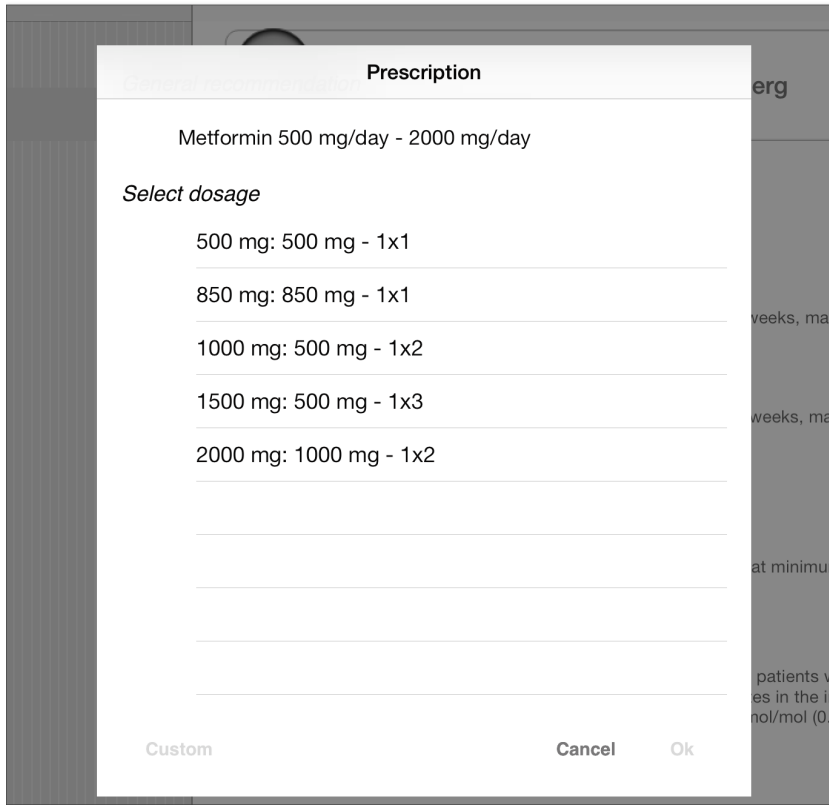


Figure 16.7: Selecting a dose from the recommended set.

Derivation

The way to allow evolution and local specialization of templates is by classic inheritance and derivation. Any new template is explicitly based on a preceding, similar template, or on the root "nil" template. The new template contains the identifier of the template[4] it derives from, so that it becomes possible to check for changes higher up the inheritance tree.

To illustrate why this is a good thing, let's assume we have a parent template block that defines the diagnostics and therapeutics of diabetes. It contains the defined glycemia limits for making the diagnosis, and the recommended oral medications, among a lot of other things. Let's also assume

[4]Only single inheritance is provided. I really see no need for multiple inheritance in this design.

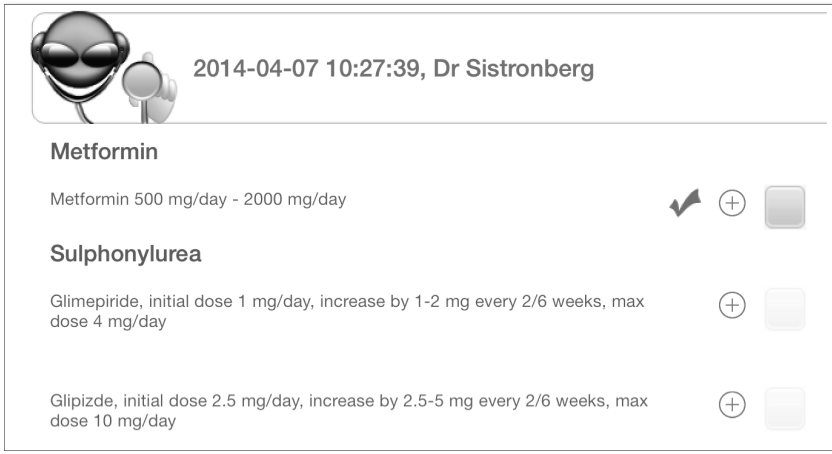


Figure 16.8: Prescription is flagged in the block.

that a new block is derived for a particular clinic, and that the diagnostic limits are different in this new block, but the therapeutic recommendations remain unchanged. The new diagnostic limits were the reason a derivation was done, instead of using the original template block as is.

At some point in time after this happened, the parent template is derived to a new version with a change in the medication recommendations. Whenever the earlier child template block is used, the system can now warn the user that there has been a change in medication recommendations in a parent template, and offer to merge in those changes and create a new child template with both modifications together.

I realize that this was quite a murky explanation, but the elevator pitch for inheritance is as follows:

By updating issue template blocks using the inheritance mechanism, any changes to medical practice recommendations will automatically spread to all users of the system, allowing for efficient and correct knowledge distribution that simply does not exist in legacy EHR systems.

To make this work, all template blocks must be globally indexed, including back pointers to parents and proper security, but the detailed explanation of how this works is outside the scope of this book[5].

[5]If you want to know how, hire me.

Blocks

When we take a regular guideline, such as the one described in section 8.4, and convert it to a template that can be used in an issue-template based EHR system, we quickly notice that the textual information must be subdivided in a particular way to allow it to be useful for a number of patient encounters over time.

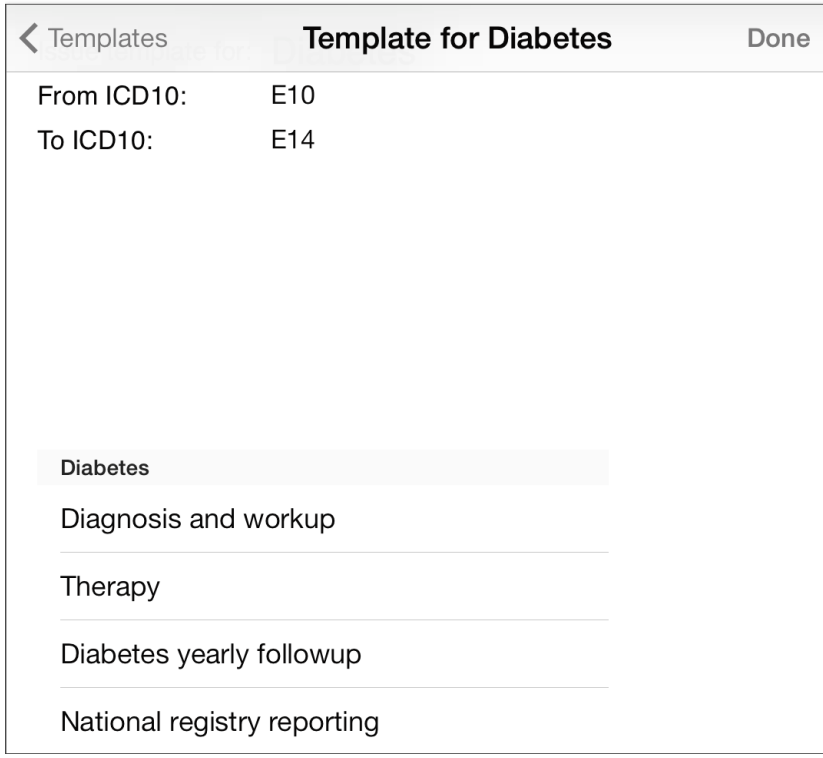


Figure 16.9: The blocks available in the diabetes issue template.

For instance, the items in the guideline that concerns making the actual diagnosis are only used and relevant once, at initial presentation. The items that have to do with yearly followups, however, are revisited at every yearly encounter. When the guideline is used as an issue template, the actual data entered into the template will be different for each encounter. This leads us to divide the issue template into "blocks", where each block is applicable to a different type of encounter. In this example, we'd create two blocks, one

is the "diagnosis and workup" block, while the other is the "yearly followup" block.

Some issue templates will need blocks for "operation" as in orthopedics, "treatments" as in radiotherapy, "hourly stats" as in intensive care units, and even "patient instruction sheets" for helping the patient in his own home care. As an example of available blocks, see figure 16.9.

Items

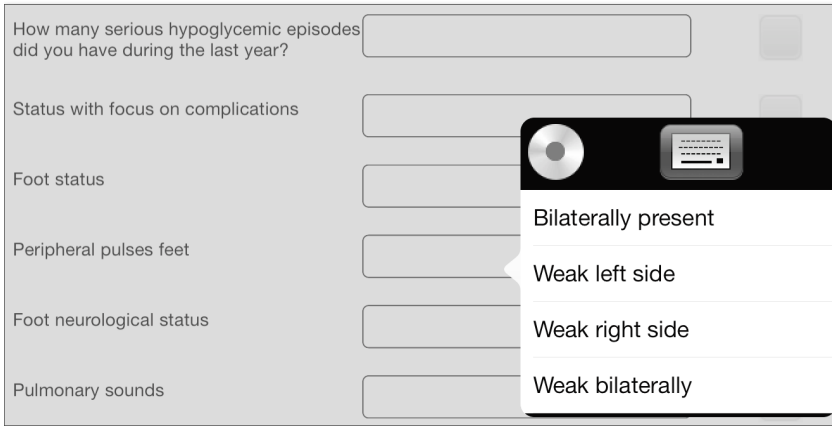


Figure 16.10: Part of a block showing six items.

Each block consists of a number of elements that I choose to call "issue items", or "items" for short. In figure 16.10 we see part of the block "diabetes, yearly followup". Within the image, there are six "items", each consisting of a "prompt", such as "foot status", "peripheral pulses feet", and so on. Next to the prompt is a field for the entry of the value. Some of these fields allow for a popup of predefined values, too.

In the figure, we see that the item with the prompt "peripheral pulses feet" has it's value popup open with four predefined values: "bilaterally present", "weak left side", "weak right side", and "weak bilaterally".

The prompt, the field, and the predefined popup values (if present), together form an "item". The item as a whole is defined in a "data dictionary" and can be reused in any number of different blocks, even across issues. This means that if you need to ask the same question about peripheral pulsations in another template block, you don't need to redefine the question and the possible answers, you can simply refer to it from two or more places at the

same time. The only thing needed in the block proper is that reference, nothing more.

The actual code that defines the shown part of the yearly followup block looks like this:

```
<item name='dd:Hypos'/>
<item name='dd:Status'/>
<item name='dd:FootStatus'/>
<item name='dd:PeripheralPulses'/>
<item name='dd:FootNeurology'/>
<item name='dd:PulmSounds'/>
```

Data definitions

As already described, the items in a template block refer to data definitions that are separately defined. This allows us to reuse data types in multiple item definitions. So, not only can one item definition be used in several templates, but data definitions can also be used in several item definitions.

The data definition for "peripheral pulses" in the above example looks like this in the data dictionary:

```
<obsdef name='PeripheralPulses' type='select' default='bilok'>
  <prompt lang='en'>Peripheral pulses feet</prompt>
  <prompt lang='sv'>Perifera pulsar</prompt>
  <select value='bilok'>
    <prompt lang='en'>Bilaterally present</prompt>
    <prompt lang='sv'>Bilat närvarande</prompt>
  </select>
  <select value='weakleft'>
    <prompt lang='en'>Weak left side</prompt>
    <prompt lang='sv'>Försvagad vänster</prompt>
  </select>
  <select value='weakright'>
    <prompt lang='en'>Weak right side</prompt>
    <prompt lang='sv'>Försvagad höger</prompt>
  </select>
  <select value='bilatweak'>
    <prompt lang='en'>Weak bilaterally</prompt>
    <prompt lang='sv'>Bilat svag</prompt>
  </select>
</obsdef>
```

As you can see, all the textual information that can be shown to the user, either as a prompt string in the left part of figure 16.10, or as one of the

predefined choices in the popup to the right in the same figure, can be defined in different languages. The actual string displayed to the user is taken from the language that closest matches the language of the logged in user. As you can also see in the code snippet, the data definition as such, and each predefined value, also has a "name" or "value" attribute that uniquely identifies it in a user language neutral fashion, allowing automatic translations of at least the predefined values, since what is saved in a database is the language-neutral value or name only.

If the user enters free form text, however, that will be saved as is in the database, and can only be translated automatically if some other mechanism is used to achieve that.

Both the code for the data definition (the "name" attribute) and the "value" attribute for the predefined values can be replaced or coupled with standard terminology references, such as for example SNOMED CT. This is, as far as I'm aware, the first actually useful application of this term system in medicine[6]. It's unfortunate that the SNOMED CT set of terms appears too incomplete to be reliably implemented in iotaMed, though[7], which explains why my examples don't show it.

16.4 The data pool

The data pool is the collection of all item data that is entered into all the used issue template blocks for one patient. Each value in the data pool is indexed on a combination of "encounter" and "data definition". The "encounter" is a certain point in time and provider, for instance "Dr Sistronberg" at "Jan 14th, 2014, 15:30".

Since the data value has no relationship to any particular item or template block, *any* reference to one and the same data definition will show the same value for the same encounter. In other words, if you have a patient with a template for diabetes and a template for hypertension open at the same time, you only need to enter a blood pressure value into one of them to see it in both. This neatly eliminates duplicate work for patients with multiple pathologies.

This separation of data values in the pool from the presentation in a context, as in template blocks, also exactly matches what we in software design describe as the "model" on one hand, and "view" on the other. To put it another way: it doesn't matter if you measure the blood pressure in the context of managing diabetes, or if you measure the blood pressure in

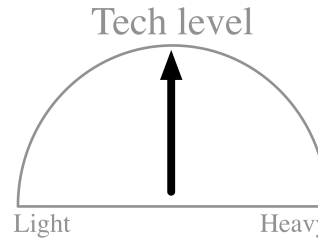
[6] Hehe...

[7] The catalogue contains around 400,000 terms, and I *still* can find only about half of what I need for an everyday trivial clinical exam. Amazing, but not in a good way.

the context of managing hypertension, the blood pressure still is one and the same value.

If you have entered the blood pressure into a template block for diabetes, and you then add in another template such as for instance for the issue "cardiac failure", then the blood pressure will automatically be pre-filled into the new template.

When you delete a template block, that is a "view", this does *not* erase any data values that were already entered, so replacing a template block with another does not mean you have to re-enter any values. This allows us to change our mind about diagnoses, deleting and replacing templates, and not have to repeat already performed clinical examinations.



Chapter **17**

Matching findings to templates

Getting from headache to migraine.

How do we arrange to have the system help us find the right set of templates from a few symptoms? For instance, if the patient has a headache, can't we already present the user with the options "migraine", "hypertension", and "meningitis", just to pick a few?

In previous chapter, I described issue templates containing a series of clinical findings, each with a set of alternative values. The template as such works as a check list, ensuring that the most important aspects of a particular issue are examined and considered by the physician. But these clinical findings have another important function, namely to indicate which *other* issues need considering, leading to a mutual dependency between item values and templates.

For example, if the physician (or the patient) starts out filling in a very general template, such as one for "fatigue", the values entered should be used to propose other, more specific, templates, such as those for depression, Addison's disease, hypothyroidism, diabetes, etc. Each of those templates should be already filled in with the values entered in preceding templates, and further entry into these more specific templates should lead to further refinement and suggestions for other templates.

The following discussion is heavily influenced by the writings of Lawrence L. Weed in the book "Medicine in Denial" [6]. Weed et.al describes this matching of clinical findings to issues (or as he calls them, "problems") as the most important advance in medical management we can achieve with computers. It would eliminate the almost random way physicians currently

make diagnostic hypotheses and perform clinical examinations and diagnostic tests. Far too many diagnoses are missed because the physician is incapable of considering the entire gamut of possible routes to a diagnosis, something a computer would have no problem with at all.

17.1 The initial findings

The very first issue template should be fairly simple and serves to select the major area the problem concerns. This template could consist of a single item with the values "fever", "fatigue", "abdominal pain", "thoracic pain", "headache", "throat ache", "skin problems", "muscle and joint problems", "trauma", "weight problems", and maybe a few more. Depending on the answer to this first question, more specific issue templates would be activated for further work.

As soon as a clinical finding is entered, the system can present a number of candidate issue templates. For instance, if "fatigue" is selected in the first issue template, the system should already list a large number of possible other issues that involve fatigue, such as hypothyroid disease, diabetes, rheumatic fever, cancer, Addison's disease, and many, many more. Each of these issues have their own set of relevant clinical findings, and a set of these would automatically be added to the active template as the user fills in values.

We shouldn't pick every possible clinical finding from all candidate issues to add to the current template, since that would quickly overwhelm the user. Instead, the most discerning clinical findings should be selected. The selected clinical findings are those that have the highest potential to reduce the number of candidate issues[1].

But that is not enough. We have to have a feedback mechanism such that the system can improve its accuracy as it learns. We also need a mechanism so that "standard problem cases" can be run against the system to verify that the system does not miss these diagnoses.

17.2 Combinatorial matching

The term "combinatorial matching" is used extensively in [6] to mean the process of finding diagnoses or treatment from the answers to a large set of questions about symptoms, clinical tests, laboratory tests, and so on.

[1]The mechanism used is akin to how an SQL query optimizer selects which index to process first, but instead of having the goal of preserving computing resources, our goal with issue oriented systems is to as quickly and efficiently as possible reduce the number of candidate diagnoses down to just one or a few.

Each combination of signs and symptoms results in a small set of possible diagnoses to work with.

Weed describes the gathering of signs and symptoms by the physician, or by the patient, with the aid of a computer, as a separate step before the actual combinatorial matching step, but it could be made more interactive and more effective, if the two steps are merged into one.

We need to define some terms before we proceed. These definitions serve the purpose for this discussion, without any claim that these terms are widely accepted. That in itself is not important, but it is important that we agree on the meaning in this context.

Issue items are defined separately and can be used in any number of issue templates. Medically speaking, each issue template contains issue items that either confirm or exclude the diagnosis the issue template defines. If the same item, say "blood pressure", is used in two different issue templates, both templates refer to the exact same issue item, *and* to the same value. This eliminates multiple entry if two or more issue templates make use of the same clinical finding.

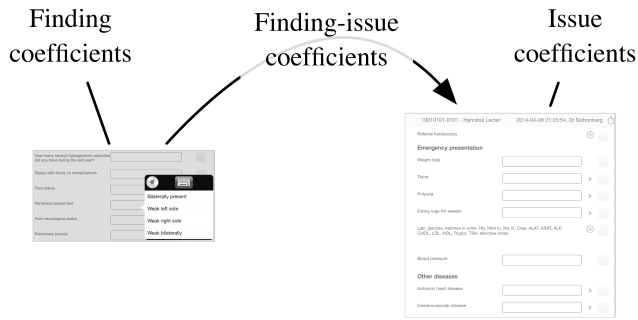


Figure 17.1: The three groups of coefficients relating to clinical findings and issue blocks.

There are three groups of coefficients determining the selection and implications of clinical findings. See figure 17.1 and table 17.2. These coefficients together determine which issue templates to consider given how the user has entered clinical findings into items, and simultaneously determine which new items to add to a list of items for consideration by the user. These added items are presented as items in the current issue template view the user is working with.

All these coefficients are determined as part of the process of building the issue template, so they are all under control of the

Table 17.1: Meaning of terms in this discussion

Term	Meaning
Healthcare issue	A disease, or a major symptom such as “fatigue” or “headache”, which can have its own clinical findings, and recommendations.
Issue	Short for healthcare issue.
Finding	Short for clinical finding.
Lab finding	The result of a lab test or group of lab tests.
X-ray finding	One or more results from a defined suite of X-ray studies.
Referral finding	One or more results from a referral.
Sign	One or more results from the clinical examination of the patient.
Symptom	Something the patient reports during history taking.
Clinical finding	Any of the above findings or symptoms.
Issue item	The technical implementation allowing the user to enter a clinical finding into an issue block.
Item action	The implementation of the mechanism to create prescriptions, referrals, lab orders, X-ray orders, letters, forms, other forms of documentation.
Issue block	The collection of entry fields, widgets allowing creation of item actions, and links to background information, related to a particular part or phase in a healthcare issue.
Issue template	The collection of issue blocks for a particular healthcare issue.

Table 17.2: Groups of coefficients

Group	Function of the group
Finding-issue coefficients	This group determines the degree of inclusion or exclusion of a particular healthcare issue in the list of likely issues.
Finding coefficients	This group determines how far up the list of questions this particular item should be placed, other factors being equal.
Issue coefficients	This group determines the sorting of issues that are similarly ranked by other calculations.

one mind with the most expertise on the subject, with the best shot at achieving "conceptual integrity" [2, p. 42].

Finding-issue coefficients

When a clinical finding is positive, it will imply that some diagnoses, i.e. issues, are more likely, and some diagnoses are less likely. For instance, an increase in body weight makes the issue "congestive heart failure"[2] more likely, while at the same time making the issue "hyperthyroidism" less likely.

When the finding is negative, it will also imply that some diagnoses are less likely. For instance, a normal hemoglobin will make both polycythemia vera and iron deficiency less likely. A negative finding may, in combination with other findings, increase the likelihood of issues. For instance, if hemoglobin is low (a positive finding), but the average volume of red blood cells is normal, it elevates the likelihood of bone marrow problems[3].

Even though clinical findings interact this way, we don't need to invent some complicated algorithm to calculate these effects. Instead, we'll rely on pattern matching the way Weed et.al describes [6]. It turns out that the issue "bone marrow problems" has a pattern of "normal cell volume" combined with "low hemoglobin", and therefore is the best match for this combination of clinical findings.

We can construct a set of coefficients reflecting this relationship between the two clinical findings and the three issues as in the tables 17.3 and 17.4.

[2]This is just one of many diagnoses made more likely by increased body weight, of course.

[3]Don't take this text too literally. I'm oversimplifying for the sake of argument, and I really haven't taken all the factors into consideration.

In the example, I'm using the issues "congestive heart failure" (CHF) and an overactive thyroid gland (hyperthyroidism).

Table 17.3: Positive finding-to-issue example

	CHF	Hyperthyroidism
Weight gain	+0.5	–0.9
Fatigue	+0.7	+0.7
Tremors	0	+0.7

Table 17.4: Negative finding-to-issue example

	CHF	Hyperthyroidism
Weight gain	–0.1	0
Fatigue	–0.3	–0.3
Tremors	0	–0.7

As you can see in the table, values go from −1 through 0, and up to +1. −1 indicates that the diagnosis is excluded, and other, smaller, negative values indicate varying degrees of the clinical finding being an argument against the diagnosis. Positive values indicate arguments *for* the diagnosis in the same way, where +1 would indicate that the diagnosis is a certainty with this positive clinical finding. A value of 0 indicates that the clinical finding gives no indication whatsoever about the diagnosis, either confirmatory or exclusionary.

For instance, from the first table (17.3), we can see that weight gain is an argument for CHF to a medium degree (+0.5), but strongly against hyperthyroidism (−0.9). Fatigue is a good argument for both diagnoses (+0.7 in both cases).

From the second table (17.4), we see that the absence of weight gain says nothing about there being hyperthyroidism or not (0), but is a weak argument against CHF (−0.1). Clearly, even if the presence of a finding can be an argument for a diagnosis, its absence does not always form an argument against it. It may, but this varies from finding to finding, and from diagnosis to diagnosis.

One can argue that a clinical finding with the value 0 shouldn't even be in the issue template, since it has no relevance to the issue. However, it may be important for pointing to other possible issues, that is, as part of a differential diagnosis[4]. If the user is working through a "congestive heart

[4]A "differential diagnosis" is the process whereby we look for alternative diagnosis, that is, other explanations. Most findings can have a multitude of explanations.

failure" issue template and answers the question about "tremors" with a "yes", the system could include the "hyperthyroidism" issue template automatically, and thereby also include other fields for clinical findings relevant to the "hyperthyroidism" issue. So a 0 value *does* have a function, albeit indirectly.

If we introduce "general" templates that are used during initial workup to cover all the symptoms and history elements that should be worked through for every patient, these templates don't correspond to any particular diagnosis, so *all* the finding-issue coefficients will be equal to zero. This is the way we implement the initial database that Weed stipulates. Depending on the entered value in this initial template, other candidate issues will be pulled in and listed.

The conclusion is that we need to view the influence of findings on the selection of diagnoses (issues) as consisting of two independent parts: the influence of a positive clinical finding on the selection or deselection of candidate issues, and the influence of a negative clinical finding on the selection or deselection of candidate issues. These two influences are generally independent of each other.

Issue-item coefficients

Each issue item (clinical finding) comes with its own set of coefficients that are independent of any issue templates that may use the item.

Table 17.5: Issue-item coefficients

Coefficient	Meaning
Cost	An indication of the actual monetary cost of answering the item question.
Delay	Indicates how long it will take to produce an answer.
Discomfort	An indication of pain, discomfort, and general scariness involved in answering the question.
Applicability	A set indicating applicability to genders, races, ages, and so on.

The set of issue items to present to the user is determined by the selection of candidate issues (which in turn are selected by other items), but the order these issue items are presented to the user is largely determined by the coefficients in table 17.5. The cheapest and quickest clinical questions should indeed be answered first.

Another factor also has an influence on the order of the items, namely the selective ability of the item. The more difference an item can make to the set of candidate issues, the higher it should come in the item list.

Some tests are interdependent or exclusionary. For instance, you should not do contrast imaging of the urinary tract within a couple of days of a barium contrast study of the colon, since the barium will cloud the images. One could imagine any number of expressions or coefficients to take care of all such interdependencies, but that would be overkill. In most cases, it would suffice with a short warning text as part of the template from which you order these test. After all, we're plucking the low hanging fruit here, and we're not trying to automate *everything* away.

Issue coefficients

Each issue template comes with its own set of coefficients, reflecting attributes of the issue itself.

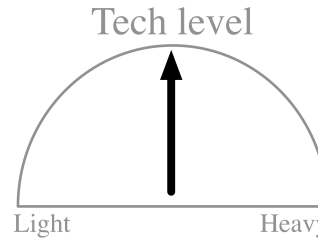
Table 17.6: Issue coefficients

Coefficient	Meaning
Prevalence	An indication of how common the issue is.
Urgency	Indicates how urgent it is to diagnose or exclude this issue.
Importance	Indicates how important it is not to miss this issue.

The "prevalence" coefficient should not be used to include or exclude issue templates from the list of candidate issues, but can be used to order candidate issues of equal likelihood. If, for instance, a certain set of clinical findings point to both hypothyroidism and sleeping sickness with equal likelihood, hypothyroidism should come first in the list, if we're in Europe, simply because hypothyroidism is more frequent.

The "urgency" coefficient elevates the consideration of certain issues that must be quickly found if present, even if they are not the top contenders in the list of issues. For instance, chest pain usually does not indicate a myocardial infarction, but because it must be detected early if it is happening, it is the first thing we should exclude.

The "importance" coefficient is similar to the "urgency" coefficient, but indicates that the issue definitely should be considered, even if it doesn't have to be immediately.



Chapter **18**

Document tree

There's logic in how we reason. This is how you persist that logic in the system.

In the section on receiving results (section 12.8), I described how each received result must include references to all other documents and results its conclusions are built upon. In the document structure we have in current systems and that I described on page 73, that is quite impossible to achieve. The relationships simply aren't in place to do that.

What we need instead is a document structure in the application that mirrors how decisions and documents in clinical practice depend on each other. As an example, when I write a referral to a specialist for a patient with some as yet undefined problem of the liver, I'll probably send along some lab results and the protocol from an ultrasound examination of the liver. This is shown in diagram form in figure 18.1

In this figure, the top-most element has a double border to designate that it is a "root", in other words, an element that is not owned or included by any other element. The significance of this will become clear later.

In my example, the specialist performs a liver biopsy and sends it off, then later gets a report from the pathologist about the tissue sample, then finally uses that tissue report and the ultrasound protocol I sent him, together with his best judgement, and writes me a reply to my referral.

In my referral, I include the ultrasound protocol and a lab report. His reply includes the pathology report and the ultrasound protocol I sent him. In his report, he refers to the ultrasound and the pathology report, but

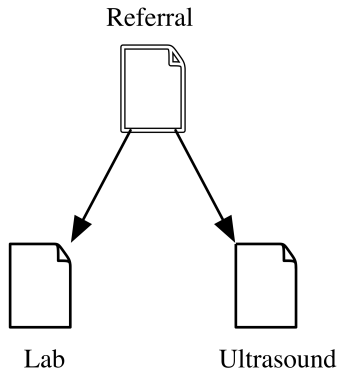


Figure 18.1: Referral with two included documents

he never needs to refer to the lab results, so only the ultrasound and the pathology report will be referenced in his reply to me.

His reply will also refer to and include my initial request. So what we'll see in the record system after receiving the report will look like figure 18.2. Also note that the report from the specialist is now a "root" element with a double border. The request I sent to the specialist has lost its "root" double border since it is now referred to by another element, the report.

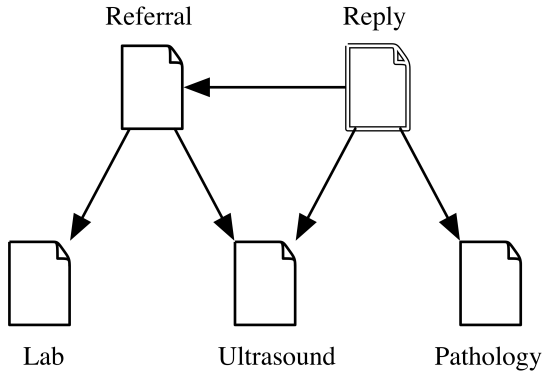


Figure 18.2: My system after receiving a reply to my referral

If the system I'm using had a list of "root" elements, that list would initially hold just one entry, the request in figure 18.1. Once I got a report back from the specialist, that entry would disappear, since it's not a root any longer, and be replaced by an entry for the report back from the specialist, as in figure 18.2.

So, what does this mean in clinical terms? Each root is an independent problem, an issue, something we should keep an eye on or react to. After writing a referral, the referral becomes an item to watch, to keep an eye on, until it results in a report back. As soon as the report comes in, the referral can be removed (which it automatically is, since it becomes referred to by the report) and does not need any attention anymore, but the report itself now becomes an item of interest. That report *remains* an item of interest, a "root" element, an item in the list, until some other item is created that makes use of it, and it becomes part of a larger, higher-level reasoning.

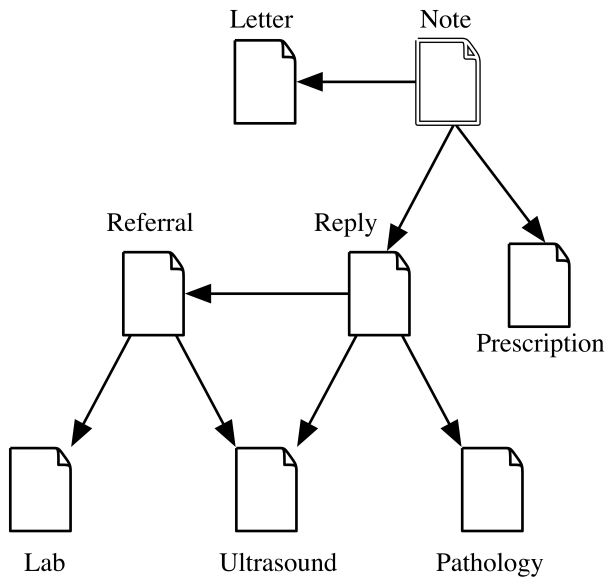


Figure 18.3: The document tree after writing the note

If I read the report from the specialist and make a note of that in the records, write a letter to the patient, and create a prescription, while referring to all three in my note, I will have replaced the root, that is the

element to keep an eye on, with that note. The new relationship will look like in figure 18.3. In this tree, the note itself becomes an "item of interest". Clinically, this makes sense, since we need to act on it, make it part of something else, classify it as part of a healthcare issue. Maybe that issue could be "liver problem". If we add that high-level root to the records, it will look like figure 18.4.

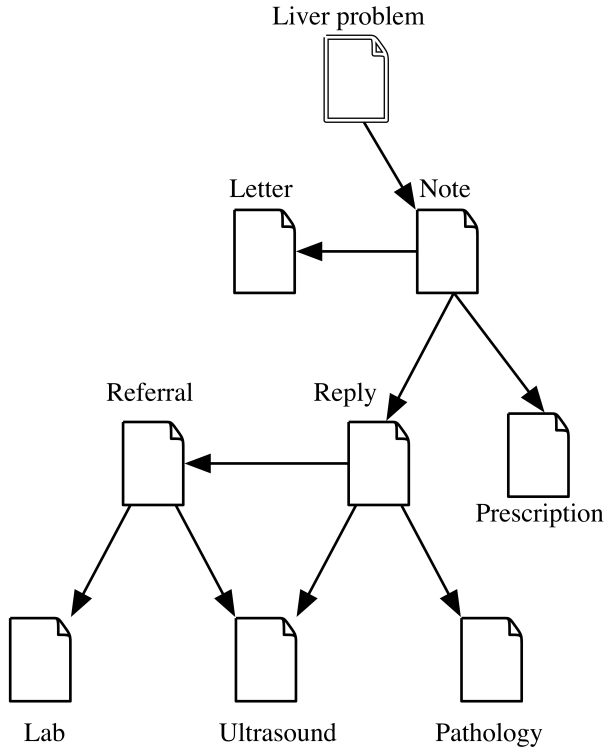


Figure 18.4: After creating the issue "Liver problem" at the top level

This high-level healthcare issue could also be diabetes, hypertension, or schizophrenia, for instance. It could also be a symptom which hasn't been clearly assigned to an issue yet, such as "headaches" or "exhaustion". It could also be an incoming request that hasn't been seen and handled yet. The list of root documents forms an *excellent* overview of the patient's current issues and any elements that are not yet referred to by any other elements. New incoming results will therefore automatically show up in this list and stay there until someone acts on them. As results are acted

on, and therefore linked into other actions as subdocuments, they disappear from this list. The only way to remove an element from the list of things "to keep an eye on" is to act on it in a way that makes that element part of a larger whole. The very action on the element removes it from the list. This list fills many of the functions we see in the list of "unsigned" items in current systems, but in a much more sensible way.

Another way of looking at the inclusion of documents into other documents, is that they are interdependent. The higher level document *depends* on the lower level documents. If any documents that contain judgement calls or measurement values turn out to be incorrect, then any other documents that refer to them also become suspect. This is inherent in medical reasoning and should be reflected in the document architecture of the EHR system, as it is in this design.

If we start from the top, instead, then we first read my note with my conclusions about the liver problems, and from there we can find and view underlying documents, the one from the specialist, and in turn the pathology report. Clearly, we can reach *all* details that were used in any conclusions this way. Any conclusions that are *not* adequately based on other findings will stand out as a sore thumb.

Interestingly, we can *only* reach the details that have a direct or indirect relationship to the top level issue we start with, which eliminates a lot of irrelevant information from the context of a particular issue or result. The details that we don't see as part of an issue tree will be shown as part of another issue tree, or if they are entirely free-standing, they will be shown as roots in the attention list.

Elements can have several parents, i.e. be part of several trees at the same time. This is logical, since the same lab result or specialist report can have a bearing on more than one issue. Everything that this shared element depends on will automatically also become shared between both issues.

18.1 The attention list

I've mentioned the "attention list" several times now, but it's necessary to expand on what this means. To do that, I'll go through the same example I used in the preceding to illustrate how the document tree is built, but this time accompanied by a description on how the "attention list" evolves.

The "attention list" fills the function of overview over the patient's issues, while at the same time filling the function of a list of "unsigned items" as it is implemented in current systems. In fact, it turns out that there is no definable difference between "issues" (or "diseases", or "problems") on the one hand, and "unsigned" incoming results on the other. Both

concepts are primary elements of attention and have many similarities from an information conceptual standpoint.

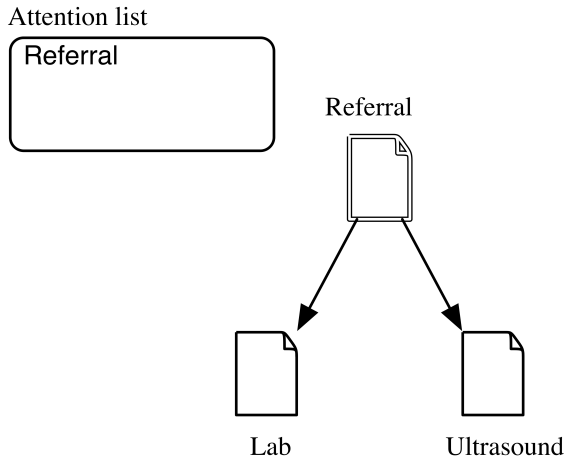


Figure 18.5: Having only written the referral

In figure 18.5 we have just created the referral and included two documents in it, a lab result and an ultrasound report. The referral itself is a top level element, a root of the tree, and will therefore also appear in the attention list at the top left. When the doctor opens the record, her attention will be drawn to this referral, indicating that it hasn't been replied to, or made part of another attention item in any way. It's freestanding and calling for attention.

In figure 18.6 the issue "Liver problem" has been created by the doctor, and the referral she just wrote has been made a part of that issue. Since the root of the tree is now and issue element called "Liver problem", that is the only attention item shown in the list in the upper left.

The specialist receiving the referral, the "referee"[1], receives one document with two subdocuments as shown in figure 18.7. The referee then creates a reply document which contains links to the referral document, the pathology report, and the ultrasound report that was sent in by the

[1]"Referee" is ambiguous, I know. It could mean the patient being referred, or the doctor receiving the referral, but in this text I will use it to mean the latter. I have no other word for that doctor, while the patient can always be called a "patient" instead. So that's what I'll do.

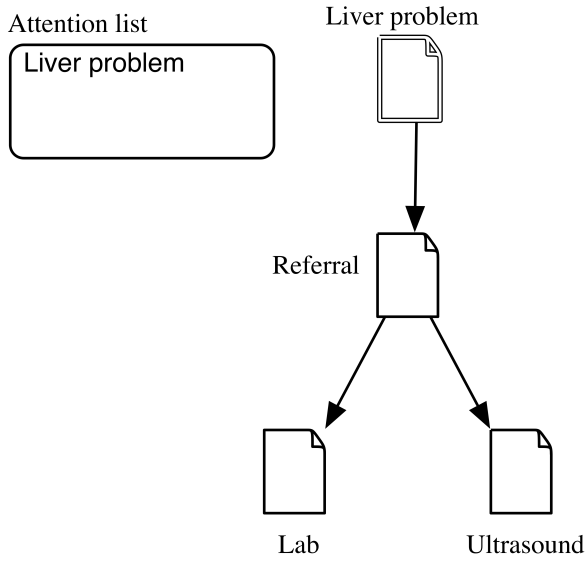


Figure 18.6: The referral has been made part of an issue

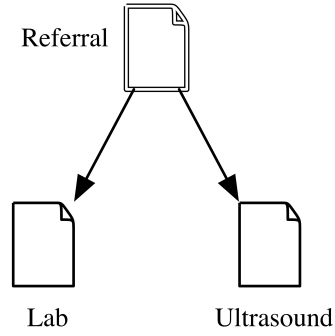


Figure 18.7: The document as received by the referee

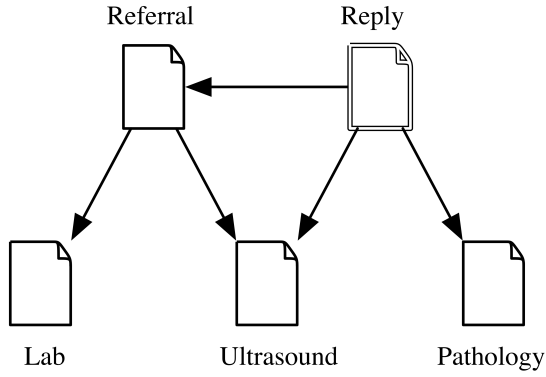


Figure 18.8: The document as returned by the referee

referring doctor. The document set that the referee finalizes and send back looks like in the figure 18.8.

When the document is received by the referring doctor, it is linked into her system as shown in figure 18.9. Both the referral and the ultrasound report were already present in the receiving system, and are not duplicated. The reply is simply linked to those preexisting documents. The pathology report and the reply itself are added to the document tree. Nothing in the existing tree refers to the reply element, so it becomes a second root of the tree (the first one is "Liver problem"), and is added to the attention list. The fact that the reply refers to elements that are already a part of the tree (the referral and the ultrasound report) does not in itself make the reply element part of the tree, allowing us to have it shown in the attention list.

When the receiving doctor looks at the attention list, she can select the reply from that list, write up a note with her conclusions, base it on the reply, and then write a letter to the patient, and a prescription for a suitable medication. The result will look as in figure 18.10. The very act of basing her note on the reply to the referral makes the reply a child node and removes it from the attention list automatically. The note itself becomes a new root and will show up in the attention list in figure 18.10.

The note is in the attention list since it hasn't been yet made a part of a greater whole; it hasn't been properly put into a context. The doctor now selects the note, then links it to the proper issue, in this case the issue "Liver problem", and so makes it a child of "Liver issue" and automatically removes it from the attention list. The result is an attention list containing a single item, the issue "Liver problem", with no other outstanding items

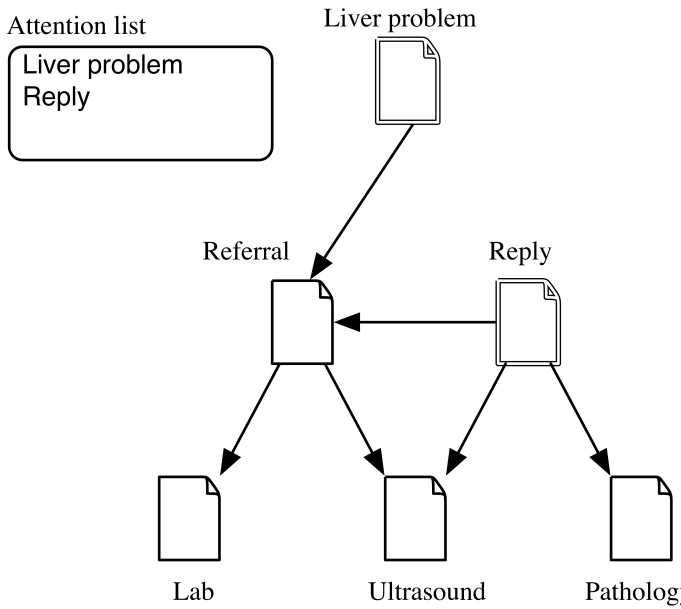


Figure 18.9: The reply has arrived

that need attention. Depending on the implementation, this step could be made part of the actual writing of the note in the first place, making the linking to an issue automatic.

One question arises, namely how to view the letter to the patient and the prescription. Is the letter based on the note, or is the note based on the letter? The same question arises when we think of prescriptions. It is clear that if the letter has no relationships to anything, it should be in the attention list so that the user is prompted to place it into the right context. The same goes for prescriptions. But if there is a link, which should be child and which should be a parent?

It turns out that elements such as letters or prescriptions are a bit peculiar, they're not below or above other elements such as notes, but more to the side of them. Since it would be clinically absurd to put the letter as such as an attention item, or the prescription as an attention item, I've chosen to always make them children of any relationships they are in.

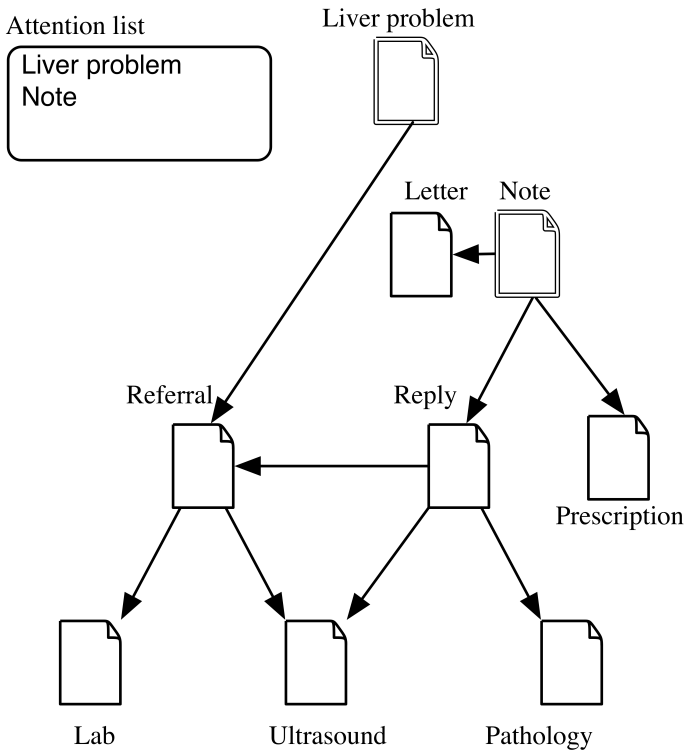


Figure 18.10: The doctor creates a note, a letter, and a prescription

18.2 Encryption

We can add an encryption twist to this tree. Assume that each included document at any level consists of a reference to the included document *and* a decryption key so that the referred document can be read. This makes it trivial to read any included documents if you have access to the document that includes them, since that is where the decryption key resides. This also makes it impossible to read a document if you haven't retrieved it by way of another document that includes it. The only way to enter the tree is by way of a top level document which can be found in the list described in the previous section. From there, you can descend the tree (yes, it's upside down with the root at the top), accessing underlying documents one level at a time.

It turns out that this is a very desirable property. If a document has been

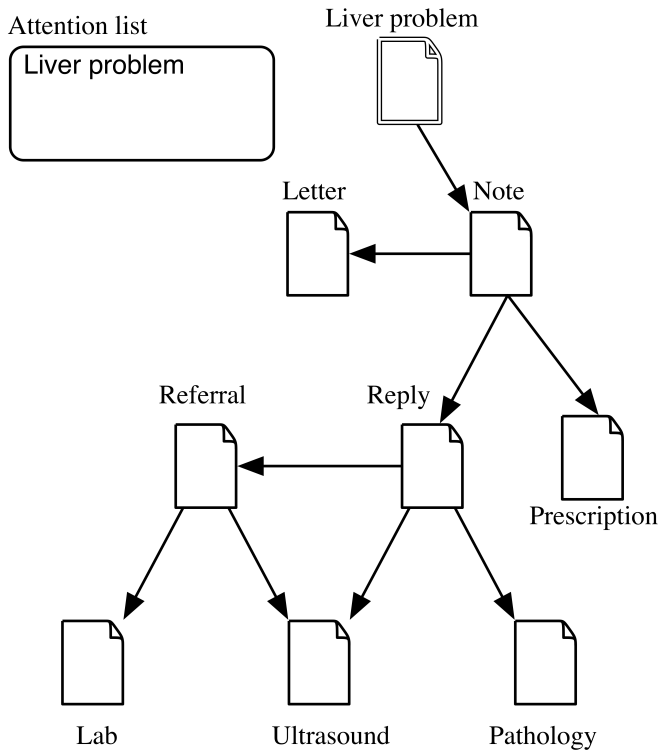


Figure 18.11: After the note is linked to an issue

used as basis for another document, that base document will always remain accessible to that derived document, but not to any readers who have no access to a derived document. In more clinical terms, you can say that if you have access to an issue, such as diabetes, you will automatically have access to everything that is relevant to that issue. If you have access to another issue, such as schizophrenia, but not diabetes, you will not have access to any documents that are part of the diabetes tree, *unless* those documents are *also* part of the schizophrenia tree. You will not automatically know that a particular document is part of a tree you have no access to, but that's how it should be. Parents know about their children, but the children do not know about their parents.

The document-tree design also neatly implements the abstraction and encapsulation I discussed in the section on encapsulation (chapter 3) earlier. It turns out that this abstraction and division into levels that I described

in some depth in that section happens continuously with every new referral or response. The document-tree design very closely mirrors how we think about referrals as doctors, which is very different from how it is implemented in current systems.

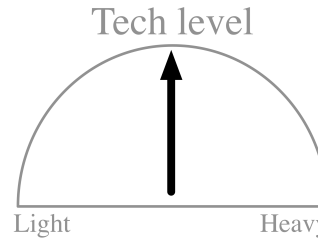
In a paper-based medical practice, I would write a referral or response and physically include copies of documents I refer to, such as lab reports and X-ray protocols. Implicitly, if you have access to the referral (by opening the envelope), you have access to the copies of documents I included in the same envelope. Since the receiver will base his or her conclusions in part on those included documents, the receiver needs permanent access to those documents. In other words, once you've sent them, you shouldn't be able to take them back.

In current computerized medical-record systems with electronic transmission of referrals and responses, this generally turns out to be impossible. There's no provision for attachments in that sense. The solution for the problem seems to be to create huge stovepipe systems, such that the receiver of a referral or result can go scrummage for himself in the record system of the sender for those documents that couldn't be attached in the first place. Worse, this seems to be the *major* reason for large unified systems.

In other words, one basic design flaw is incompletely compensated for by another, even bigger, mistake.

This poorly conceived arrangement leads to absurdities. In order to give the receiver access to material I'm referring to, I have to allow the receiver *full access* to a large part of the sender's system. If I ever revoke that access, the access to the referred documents is also revoked, removing any possibility of later, on the receiver's side, verifying that the conclusions were correctly based on included documentation.

I describe the implementation in detail in appendix A, so the interested reader can see how the whole thing really works.



CHAPTER 19

Transitioning and deployment

How do we get from here to there without climbing mountains or upsetting the apple cart?

If we compare the systems of today with the system we ought to arrive at in the future, they're quite different. Everything needs to change, including the role and type of work physicians and nurses do. We can't expect this change to be achieved in a single step, nor can we expect the change phase to be limited to a certain period in time. Instead, we must set up a system in such a way that it can be gradually introduced with gradual improvements in processes and outcomes. The system must also inherently lead to further improvements. Instead of climbing a mountain to get to the valley beyond, we need to find a road around the mountain that is always sloping down. Every change we need to introduce must produce immediate payback to all involved parties, or at the very least not introduce any net negatives. So, let's go over the phases of this change.

19.1 Phase 1, guidelines

A large number of guidelines for most common complaints and diseases already exist in one form or another. One particular guideline is discussed in section 8.4. From that discussion, we can conclude that the contents of the guideline is not the problem, as far as applicability is concerned, but the medium and presentation are. It's too hard to locate and use a guideline that is presented in a passive form, even on the web. Also, as

already discussed, any use we make of it leaves no footprint in the medical record, making it hard to verify the use after the fact.

Another advantage to starting with the automation of guidelines is that the implemented guidelines will be useful from the start, even if the available number is limited. Whatever else is covered by a "general template" as found in most current EHR systems, can be just as easily covered using a general "issue template", with the added and considerable advantage of having the clinical findings and history elements correctly coded from the start. In other words, even if the initial issue template does not have the capability to automatically analyze and suggest differential diagnoses, the data entered in such an early implementation can still be useful in the future, as more advanced analytical tools come online.

Replacing old fashioned passive keyword based templates in current EHR systems with the structured issue templates of an issue oriented medical record makes data entry easier and quicker, while at the same time structuring the data in a way that makes it useful for future tools. It also enables full assistance in the creation of correct referrals and orders.

Having implemented at least a few guidelines as issue templates, already allows for much more efficient and safe prescriptions of medications, creation of referrals, creation of X-ray orders, and lab tests. These features, and the concomitant reduction in mistakes and double entry, will result in huge improvements in efficiency of delivered care, reduction in errors, and higher consistency and quality of care.

These issue template based guidelines also provide the ability to warn for contra-indication, *and* they also enable much more useful and accurate management information, by being easily counted and tracked. Each issue template, by its very activation for a particular patient, implies that the issue the template describes can be regarded as "reported" without further intervention by the user.

19.2 Phase 2, combinatorial matching

Once there is a substantial amount of issue templates in use, it makes sense to automatically select matching issues from the set of initial findings. This process would be very similar to the "combinatorial matching" algorithm Weed describes [6], but the source of the items is essentially different.

Weed builds a single engine that holds all the items needed for an initial workup, then lets that single engine select the issues that match those initial findings. This requires one single code base to be up to date on *all* possible initial findings, and *all* possible issues. In theory this should work, but in practice, it will not be maintainable.

A more maintainable and scalable system would delegate the definition of items, and the definition of issues, to distributed specialists. The execution engine would dynamically build up the set of items to present to the user for the initial workup after extracting this set from all available issue template definitions. Similarly, the selection of issues from the set of responses to the items should be dependent on the definitions of all included issue templates. This allows the totality of the system to dynamically update itself depending on all the knowledge inherent in the contributed issue templates.

19.3 Phase 3, analysis and feedback

Any computerized decision making system needs to have a feedback loop built-in, so that the accuracy and precision can be improved continuously and automatically. If a number of physicians use the same system, the system will, as it were, make *all* the users learn from each other's experience and mistakes.

At the same time, the system needs to implement checks and balances, so that the automatic adjustments don't make the system worse or introduce errors, as can easily happen if the feedback system hunts for sub optimizations to the detriment of global optimizations. This could happen if a particular system very rarely, or never, sees particular issues, and becomes trained not to see that issue at all, once it actually presents itself.

One way to detect and correct such elimination by optimization is to introduce validation cases, consisting of unusual patient case reports where the ultimate and correct diagnosis is known, to train the system and verify that they aren't missed. A great way of picking up such validation cases is to save those cases that *were* in fact missed by the system. Using these validation cases across a number of systems will make *all* the systems, and thus *all* the users, learn from everybody's experience and mistakes.

Another important element is a review by expert humans of the adjustments made to the parameters in the combinatorial database. This has a twofold function:

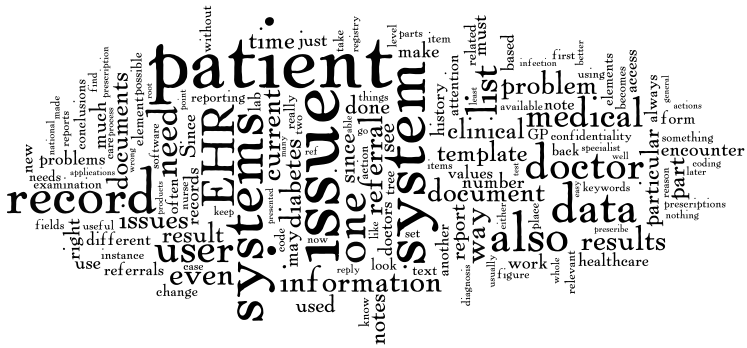
1. To detect and correct feedback adjustments gone wrong. Every algorithm has its weaknesses, and an expert human mind can detect even unforeseen errors this way.

2. To learn from the adjustments. Each adjustment reflects a mismatch between the expert knowledge of the authors of the guidelines and combinatorial parameters, on the one hand, and the reality of clinical findings in disease, on the other. Some instances of adjustments will

be signs of hitherto unknown clinical correlations. Some new syndromes in the form of new groupings of clinical findings can even be flagged entirely automatically by the use of Kohonen maps [3]. The appearance of new groups in such a map could point to a need for new issue templates for that particular group.

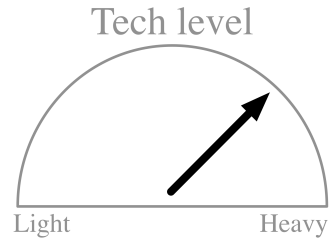
Part IV

Appendices



Summary

These appendices are completely optional reading. (As if the entire book wasn't completely optional reading, but you know what I mean.)



App. A: Document-tree design

A more detailed look on how medical documents relate to each other.

Whenever we as doctors write a summary or a reply to a referral, we always base our conclusions on other documents. It could be lab reports, radiology reports, patient history, clinical examinations, or other replies to referrals. The conclusions we reach are only as valid as those source documents are, with the added ingredient of our own competence and cognition. When we sign off on a conclusion, we do that on the implicit condition that the underlying documents were produced in a similarly considered and reliable way. If any of the underlying documents turn out to be false, erroneous, or ill considered, our conclusion also becomes, at least potentially, invalid. All this needs to be reflected in the way we manage documentation, not just having all the documents in an EHR system thrown in a heap, but by having explicit dependencies of some documents on others.

If we look at a single note in the medical record that contains a conclusion made by a doctor, based on a lab report and an ultrasound protocol, this relationship can be viewed as in figure A.1. The "conclusion" note itself contains a text where the doctor describes what the lab report and the ultrasound protocol mean, which diagnosis is made, and recommendations for further workup or treatment. The note with that text also contains links to the mentioned documents, represented as the downwards pointing arrows in the diagram. These links make the dependency of the conclusions in the note explicitly conditional on the content of the two underlying documents (lab and ultrasound).

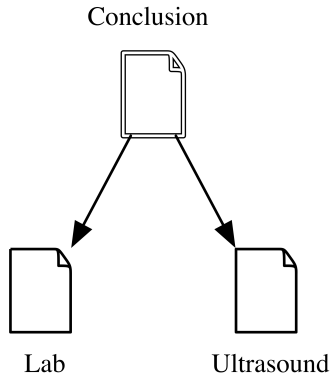


Figure A.1: A single note, dependent on two underlying documents.

As each level of medical specialist refers to a number of base documents, the specialist provides a summary at a higher level than the parts that form the basis for the new summary. The higher level summary of the lower level documents form the same kind of abstraction I described in the chapter on object oriented knowledge management (chapter 3).

This increasing level of abstraction building on lower levels of abstraction extend all the way out to the patient, who gets the ultimate summary of earlier and more detailed summaries. In figure A.2 I've shown that if the patient has access to the root element, the issue itself, he automatically has access to all underlying documents tied to this issue.

If a particular element in the document tree is still on the attention list and not linked to the issue (see page 151 for the discussion on the attention list), and the patient has direct access only to the issue itself ("liver problem" in figure A.3), the patient will not see the not yet handled element since it is not part of the tree the patient has access to. If your policy says that patients should only see parts of the record that have been seen and handled by a doctor, this is the perfect solution.

If, on the other hand, you want to give the patient explicit access to a not yet handled element in the record, you can do so, which automatically also gives the patient access to the underlying documents that element depends on. See figure A.4.

Finally, if and when the new note is handled by a doctor and becomes part of the tree rooted in the issue, the patient gets access to that part of the tree as well, automatically. See figure A.5.

Interestingly, elements at any level which were never part of summaries, or at least not part of any summary the patient or doctor at a higher level

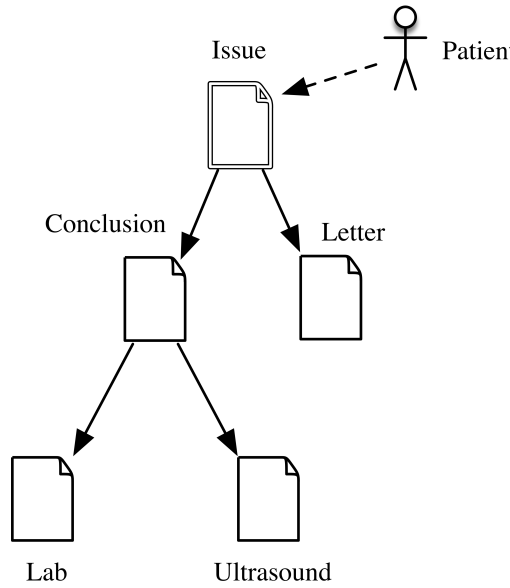


Figure A.2: Patient can have access to the root document, the issue.

ever received, become invisible to this process, which solves another problem, namely how to handle false starts or abandoned hypotheses. Unless these abandoned trails are referred to in any summary that is in the tree we're unravelling, they will become invisible, as they should be.

This solves several problems, not least of which is how to hide guesswork and scary little side trails from the patient, if the patient has access to the records. If the scary little side trail had no consequences, it won't become part of a trail of reasoning, and thus not a part of the issue tree, so it won't be seen. It's as simple as that.

Another problem we're solving with this pattern is the separation into distinct record systems for different specialities. One way of doing that is hiding all the internal details in a system and only issuing summaries from one speciality to the next. With the document-tree principle, we're doing the same thing but on a continuous basis, document by document, summary by summary, making the process much more pervasive, and at the same time with no particular border drawn around specialities.

But it gets better still. If each summary contains not only the references to the underlying documents, but also the decryption keys (symmetrical) for those documents, the underlying documents could be fully protected

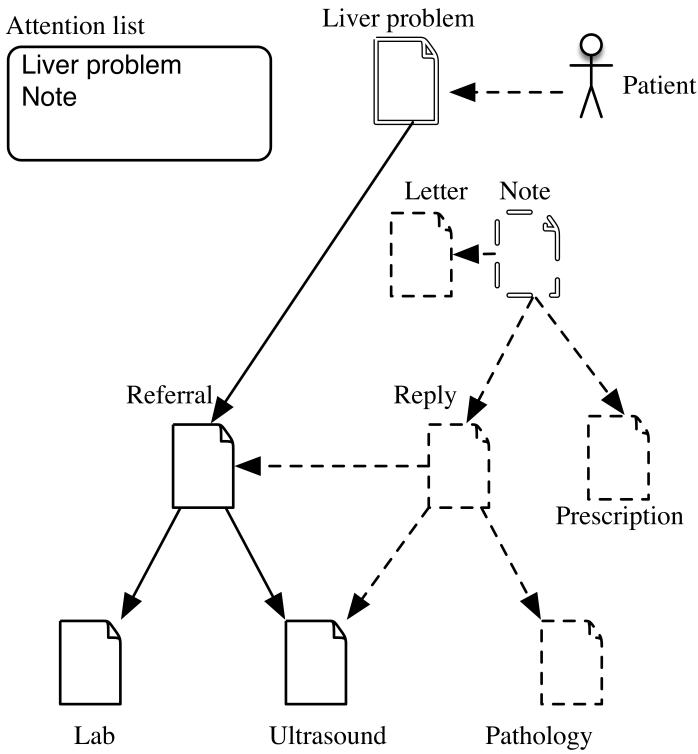


Figure A.3: The patient sees only handled elements.

from view from anyone who has no reason to see the document. And as "reason to see" the document counts being in the possession of another document that refers to it. That document in turn can only be decrypted if the key is retrieved from yet another document higher in the tree referring to it.

If a particular document is also used in a scientific study of some sort, it will become part of a collective summary for that study, and thus reachable from the study side. The same documents may, but don't have to, be part of some clinical summaries for other purposes. The meaning and function of "primary data source" used in the performance of clinical studies becomes clear and explicit. High five!

It's turtles all the way up, but sooner or later you do reach the last turtle, which could be the patient, the GP, or some kind of database, or a clinical study setup. Each patient could have more than one root, depending on

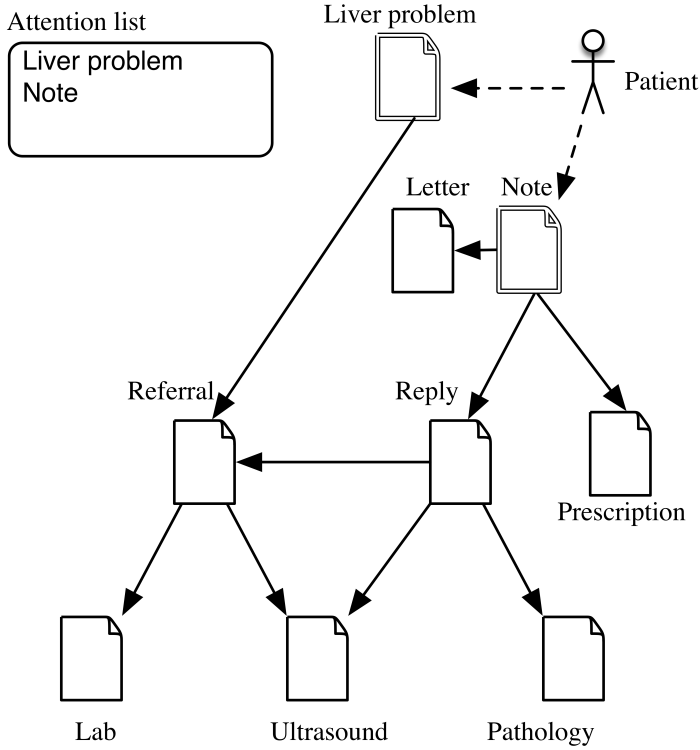


Figure A.4: The patient can be given direct access to unhanded elements.

problem areas.

If documents are arranged this way, any constituent documents can have their decryption keys "forgotten" as long as they are part of a higher level summary. Orphaned document keys could be saved in a special database for useless facts, or simply aged out and discarded.

Interestingly, this implies that *all* documents, except the very top level root documents in the tree, can be stored anywhere, even right under the nose of the NSA. As long as the root documents are safe, the other documents can't be decrypted. Root documents themselves can also be set free, as long as the key to each root document is safe. This makes the management of the (momentarily) ultimate keys quite tractable.

The act of "signing off" on received documents, which today is a pretty meaningless hassle, also achieves meaning. As it is today, we look through new results, sometimes signing off on them, but we're never really sure

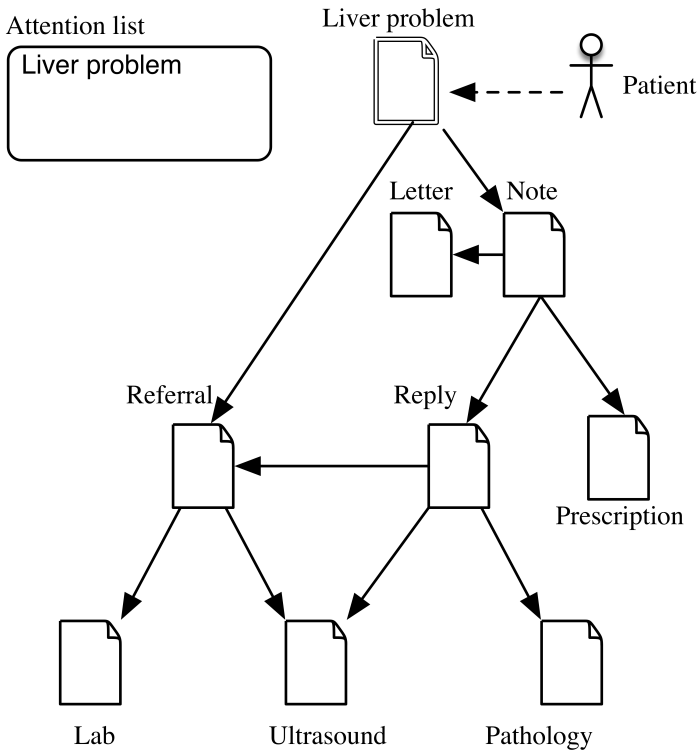


Figure A.5: When the element becomes handled and linked, the patient also gets access.

if there is an action presumed on our part, or if the simple viewing of a result has meaning. This confusion is reflected in the implementation of current systems, such that some systems flag results and reports we haven't seen (literally, just like "unread mail" in an email program), while other systems expect us to click a "sign" button, and still others implement both methods in a crazy mix of contradictory messages. If new results instead are viewed as document references that are not made a part of another higher level summary of some kind, then the action expected of the doctor to remove a result or report from the "new" list, is to incorporate that reference in a summary, which could be a note, an outgoing referral, or anything similar. With this system, the actual *action* that makes *actual use* of the result or report, *forms* the signature. There's no ambiguity left. Instead of "unread" versus "read", we have "not handled" versus "handled",

or "unused" versus "used", which makes a lot more sense. Every report requires some kind of disposition, or it will stay in the list of the "things that need attention". Neat. (There's a more in depth discussion on these lists of "unused" elements, which I call "attention lists" on page 151.)

The documents as such, in their encrypted form, can be stored in a distributed-hash-table form or similar, for maximum accessibility and redundancy, with minimal requirements on local database resilience.

Document checksum

What I'm describing here is the technical design to persist these dependencies, while it's up to the designers to make this process transparent and painless to the user. It could be as simple as a clickable symbol inside a document to open any included references to other documents.

First, we need a unique and reproducible reference to documents in general, and the most obvious choice is a checksum on the canonicalized contents of that document. Yes, that was a mouthful, but it can be explained.

A checksum is simply a sum made from the actual text of the document using a pretty complicated arithmetic process.

A "canonical" version of a document, is a document where we've normalized spaces and other invisible characters, so that differences between systems should not influence the value of the checksum. We can, for instance, stipulate that all trailing spaces in lines should be removed, and we could specify a particular character encoding in the file, and how the end of lines should be marked[1].

Document signature

A document signature is nothing but an encrypted form of the document checksum, where the encryption is done using a public-key system, with the private key of the person signing the document. First we create the checksum (hash) of the document:

$$SHA(M) = M_H$$

Assuming our signer is good ol' Alice, we encrypt this number M_H with Alice's private key $PK_{\text{priv}}(A)$ to get the encrypted version:

[1] As an example: some Unix-like platforms like OSX use a single LF character at the end of each line, while Windows and DOS use a pair of characters: CR and LF. Before calculating the checksum we must convert to one of these versions and see to it we always do that, regardless of which machine we're doing the calculation on. If we didn't, a document looking exactly the same to the user would result in two entirely different checksums on OSX and on Windows.

$$M_H \overset{PK_{\text{priv}}(A)}{\longrightarrow} M_{S(A)}$$

The resulting signature M_S can be stored as a pair with the original checksum M_H in a database, allowing us to look up who signed off on a referenced document even without knowing what that document contains. This can be very useful at times. Note that the digital signature M_S contains a reference to the owner of the key, so the identity of the signer can be recovered from the signature.

References summary

We can refer to any element from any other by using the document checksum as an identifier. The document can be stored anywhere convenient, and actually locating the document by this identifier can be done in any number of ways, for instance by using distributed hash tables.

The documents should always be stored in an encrypted form, so to access the contents once the document has been located, you also need a decryption key. Each reference to a document must therefore include two elements: the document reference (which is equal to its checksum), and the decryption key. I'll use symbols as in the following table. The letter "M" denotes "message" in crypto lingo, but in this case the "message" the same thing as "document" or "element of the record".

Table A.1: Symbols for document references.

Expression	Meaning
M	The clear-text document.
M_E	The encrypted document.
M_H	Document checksum (hash), which is also its identifier.
M_K	The symmetric key used to encrypt and decrypt the document.
M_S	The digital signature on the document, created by the originator of the document.
M_R	The reference to a document from another document that is dependent on it.

With this notation, we see that a reference to a document must include two things: the document identifier (so we can locate it), and the symmetric encryption key (so we can read it):

$$M_R = \{M_H, M_K\}$$

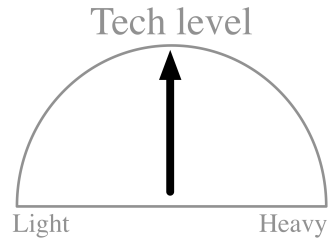
The document itself, in its encrypted form, is stored together with its identifier, so it can be located:

$$\{M_H, M_E\}$$

The document signature can be kept separately. It also needs the document identifier so we can find it and link it to the document it signs:

$$\{M_H, M_S\}$$

Since the checksum was calculated on the plain text form of the document, it cannot be verified to be correct until after the decryption of the document. The same limitation applies to the digital signature; the document must be decrypted before it can be verified.



App. B: About the author

What happened to me, to make this book happen to you?

There's no avoiding a section on who I am. This book is based almost entirely on my own experiences, and this both liberates me and forms a limitation on the applicability of my conclusions. There must be a large number of situations where my descriptions are invalid, but I have very little other literature to base any comparison on. So I'll simply have to describe my experiences and let you draw your own conclusions as to why I'm saying what I'm saying. I'll include a lot of details, but I'll try to be brief, since this isn't supposed to be a biography, but a book about the EHR.

I was born and grew up in Stockholm, Sweden, attending Stockholm University between 1969 and 1971, studying Mathematics, a sprinkling of programming (ALGOL, if anyone remembers that), and inorganic chemistry.

In 1971 I moved to Ghent, Belgium, studying medicine at the state university between 1971 and 1978, graduated with honors. During my last year of internship, I once got the task of reprogramming a small part of an analysis automation package for clinical chemistry on a PDP–8 running Focal. I managed to set fire to the power supply by issuing a bad monitor command. At least, that's what everyone around me assumed. The timing was perfect.

Between 1978 and 1983, I did a general surgery residency at the same university. This included six months of orthopedics and quite a bit of intensive care. Vascular and thoracic surgery was also part of the daily work. Train-

ing didn't include training in cardiac surgery as such, but each resident assisted hundreds of coronary bypass and valve replacement procedures, and a smaller number of pediatric cardiac operations.

Figure B.1: The main unit of the HP 2100A mini computer. (Photo courtesy of hpmemory.org, Marc Mislanghe.)

During this residency, I stumbled across an HP 2100A mini computer in a back room of the ICU. This machine had an A/D converter, a *huge* 14 inch removable disk and a fixed hard disk of the same size, each almost 2.4 megabytes in capacity. The RAM was 32k words (64k byte), and the clock frequency just a smidgen under 1 MHz. I found terminals and other

peripherals in the basement and other departments and hooked them up. I ended up with the teletype console which was already in place, a Tektronix vector graphic terminal, and a couple of HP terminals, and cheaper terminals, I've forgotten the name of, and a couple of little 8" (or so) graphic displays with 256 x 256 pixels[1].

The A/D converter had 128 channels and was still hooked up to a number of bedside monitors in the ICU. It used DMA and kernel processes to write directly to disk and could read at 50 samples per second.

The entire system was housed in two 19 inch racks bolted to the ground. When the hard disk got going real well, it would have tipped over the racks otherwise.

The way to boot this monster was interesting. Assuming it lost all track of reality, the first thing you had to do was insert a physical key and unlock the last 32 words in memory, so you could punch in the first bootstrap loader, the "loader loader", using front panel illuminated buttons, all in binary. (After about a hundred times, I knew this boot loader binary by heart.) After entering the boot loader binary, remove the key, locking down the memory[2], set the instruction counter to the start of the boot loader, insert the disk operating system boot loader paper tape in the paper tape reader, then hit "run". The paper tape ran through the reader at an amazing speed, hitting the wall almost two meters to the left. Then the disk boot got going, the RTE-III "Real Time Executive" was loaded, and after a while you got the satisfying hum and clunk from the teletype.

Residents had to stay over in the hospital two or three nights a week, while not getting any days off, so we spent 80 or 90 hours a week at work. We slept some, of course, but never enough. Once I found this HP 2100, I hardly ever slept, instead I spent my nights figuring out how to reconfigure it using "system generations", and how to program it. This is the first time I encountered the phrase "If all else fails, read the instructions", which was written on the title page of the HP 2100A manual. There's deep wisdom in that.

Using Fortran on this machine, the first thing I wrote was a system to store medical reference information. In the ICU, we had a lot of snippets of tips and tables, like how much blood a patient is allowed to lose the first hours after a coronary bypass, how long to leave a choledochus drain in place, how to calculate cardiac output using the Fick method, and so on[3]. I'd already programmed a Texas Instruments TI–59, one of the earliest

[1] I may misremember details, so please don't kill me if that was 128 x 128 or maybe even 512 x 512, but I don't think so.

[2] Theoretically, this boot loader code should never be overwritten, since there was a hardware lock on that region, but it still happened, and I never figured out why.

[3] I'm not going to explain every medical term here, since it wouldn't add to the story. If you want to know, there's always duckduckgo.com.

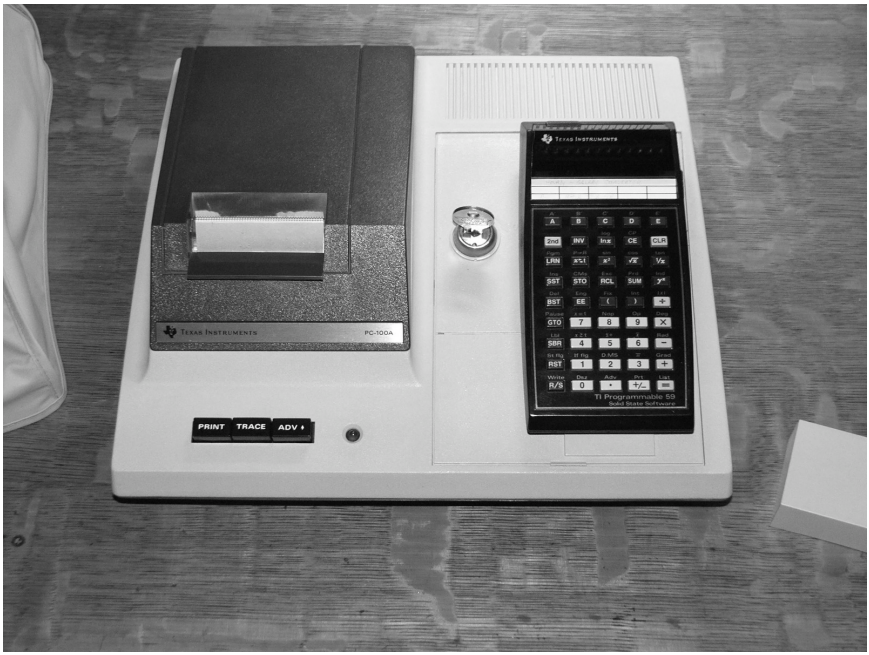


Figure B.2: A TI–59 calculator docked to its PC–100A printer. (Photo Wikimedia Commons, John Crane.)

programmable calculators, to calculate cardiac output, pulmonary shunts, and a few more little handy things. The TI–59 used little magnetic cards that were fed to the calculator and pulled through it by a little electric motor when you switched programs. For a while, I and the other residents in the ICU used my TI–59, and we passed it from person to person. One of the teaching staff had a PC–100A printer dock, which I used to program some statistics for a while.

With the HP 2100A, I moved those calculations over to the mini computer, added a lot more of our notes of collected wisdom, and set up a terminal in the ICU nursing station. To retrieve information, I figured out an indexing system so that every page of information contained a menu at the bottom leading to other pages. I also built a subscription system so that the residents that "subscribed" to my pages could get a printout of everything that had changed since the last printout they'd gotten.

After this, I wrote a system that used the A/D converter to connect to a bidirectional ultrasound vascular doppler machine, calculating and drawing a blood flow curve of the aorta. Together with a measurement of the

diameter of the aorta, this system could calculate cardiac output without any invasive procedure. Neat and almost accurate enough for clinical use, but not quite.

I also built a tool to make researching medical records somewhat easier. This is the time before we started to disparage retrospective studies, so we did a lot of those. We researched patient histories to find relationships. Doing this was a lot of work, so I came up with a system that let you set up a list of questions to work through for every medical record studied. Then the answers were fed into the system and you could go play interactively with the data to see what correlated to what. The statistics I implemented were based on the Student T-test and the chi square test, but that was pretty much enough. The user selected two different questions from the list of questions, or data points, if you will, the machine then calculated if there was any correlation between the answers for those two questions. It took about ten seconds for a verdict of significance or not to come up on the terminal. My dream was to let the machine run through *all* the possible combinations on its own, but there was no way this could be done with hardware that slow, or with my less than stellar programming skills at that time.

The first study we used this for was a comparison of post surgery treatments for esophageal cancers. We thought we had quite different results from another university, and we couldn't figure out why. I demonstrated this to Thiery Anné, the lead author of the study [1] one night we were both on call. I grabbed out of thin air what I thought was a ridiculous correlation, comparing the outcome of surgery followed by 5-FU, a chemo therapy agent which we used a lot, depending on location of the cancer (top, middle, or lower esophagus), which I was sure had nothing to do with it. This was about the third thing I showed him. The computer spit out "$p < 0.005$", and Thiery just laughed, saying "yeah, sure, as if THAT would be right... great programming, man!", and wished me luck finding the bug. I spent the next two nights trying to locate the coding error, then finally pulled out all the data, around a hundred patients, and recalculated the whole thing by hand. It all came out exactly the same as the computer had shown. That was the most important finding we had, and it was correct. It explained the difference we saw between the results in our hospital and that other hospital I mentioned, since our patients differed quite clearly in how high in the esophagus the cancer was located. It was also the last time we found anything as spectacular using that system.

During that period I also enrolled for engineering studies, and I got almost two years into it before I had to give it up. There was simply not enough hours in a day, or a night, to do all this.

We did a lot of heart surgeries, sometimes up to three or four in a day, and the main bottleneck was the ICU. Back then, a typical patient stayed

on assisted ventilation up to 24 hours, then had to stay in the ICU another two days or so. We only had a total of six ICU beds for vascular and cardiac patients, so something had to be done about this. Extending the ICU and hiring more nurses was out of the question. Other solutions needed to be found.

Dr. John Kirklin at the University of Alabama had succeeded in reducing the mean length of stay in his ICU down to eight hours by using computers to run the infusion pumps. It turned out that if you continually adjusted the flow of blood pressure stimulators and blood transfusions automatically according to measured parameters, the patient achieved hemodynamic stability much sooner, and could be taken off the ventilator and moved to medium care much quicker. This is what we wanted to do as well.

I was sent to the USA with two engineers to be trained in the programming of a later operating system version, RTE-IV on the HP 1000 system, a successor to the HP 2100A system I was used to. After that two week course I went on with my wife, but without the engineers, to Birmingham, Alabama, to watch how they did cardiac surgery there.

The idea was that the University of Ghent would also invest in more recent computers, infusion pumps, monitoring, and staff, and do what they'd done in Alabama, increasing the throughput of patients through the ICU, while at the same time reducing complications[4]. But as we got back from the trip we were told that there was no money to spend on equipment or staff. I had figured on a career as a surgeon doing some space age stuff with computers in the ICU, but it wasn't to be.

In 1983 I quit the university and took over a general practice, patients, house and all. At the same time, I started my first company which initially sold TRS–80 computer games tapes through mail order. Soon after starting the company, I wrote a very rudimentary home nursing application on a TRS–80. Then I got an order to write a system to estimate cooling systems for offices and computer rooms, which I did on the EACA Genie-III TRS–80 compatible system. That system ran both NewDOS and CP/M.

In the first few years, I sold a number of machines, including the Genie-III machines, Goldstar computers (which were later renamed to LG) and one of the first networks (based on TeleVideo mmmOST). I also wrote an accounting program, a program to store patient record diagnostic codes, and an insurance agent management program. Mostly, writing these programs served to sell turn-key systems, which was a pretty good business back then. I also programmed and installed a few industrial control computers from Merlin Systems in the UK. Not much of peripheral electronics was available at that time, so I also designed and built some minor electronics

[4]Anything that reduces time under anesthesia, or time spent in the ICU, is known to reduce infections and other complications.

like Whetstone bridges with differential amplifiers for Pt100 sensors, input protection circuits, and relay boards.

Around 1987 I arranged a connection to Medline[5] through some organisation I've forgotten the name of in Köln in Germany, utilizing packet switching to get there, and a search language I think was called "Diane". When they had a particularly difficult case at the university ICU where I used to work, they called me and I did a literature search using my Medline connection, then reported back over fax with a summary of what I found. The very expensive fax machine I sold them for this purpose must have been one of the first machines used in that hospital.

I could do a search like this in about an hour's time total. I loved doing this, but it ended a year or two later when the ICU got their own set of Silverplatter's Medline CDROMs to search through. I still think my results were a lot better than what they could find on their own, though.

In 1990 I joined "Fidonet" and later ran what was then one of the larger regional nodes[6], using three regular phone lines and an ISDN basic rate connection. Pretty bleeding edge at the time. With the computers I used for this, I also wrote a communication system to deliver lab reports and referral results to general practitioners and specialists. The system was running the "Fossil 5" communication drivers and used a system I built in the Clarion language and assembler to receive files from the labs and deliver them to users when they called in. I actually had a form of end-to-end encryption using symmetric keys that were exclusively stored in a database at each end-user's system. The client application, also written in Clarion, even contained a rudimentary medical records system with daily notes. I only charged the senders for the "stamp"; receivers got the whole thing for free.

This system was pretty darn good, but I had absolutely no time or talent for marketing, so I never got beyond about ten or 15 subscribers. In 1994 I was approached by a company called "MediBridge" who had the backing of the Belgian telecoms giant Belgacom and the University of Ghent. They had a similar product but built on Solaris systems and packet switching. The end result was way more expensive than my system with less functionality and reliability. My system's existence had hindered them in several sales where there was always at least one doctor who had seen my system and didn't want theirs. So MediBridge offered me a job and some cash if I

[5]Medline is the world's largest collection of medical publications, usually with an abstract, seldom with full text articles. It was the online incarnation of "Index Medicus", the paper-based index of published articles. Medline is part of the US National Library of Medicine.

[6]For the old-timers out there: my node address was 2:291/1906. Now, *that's* some addressing for real men, you DNS huggers!

let my system die and started working for them instead. This was my way out of my general practice, so I took it with both hands.

During the next year, I was also hired by the University of Ghent to do the Belgian version of the "Episode of Care Summary" specification for electronic messaging interchange. My boss, George De Moor, was then head of the Technical Committee 251 at CEN, the European Standards institution, which led to me being an observer there for almost a year.

In the same period, I became an advisor to the ACC, an organization of around 50 private hospitals in Belgium, to help with the development of their new medical records system, the AZIS–2000. Here I learned from the inside what it is like building a medical record system in an organization that basically had no idea what they were doing, and really couldn't care less. I walked out of there alienated, angry, disappointed, and very disillusioned with vendors of EHR systems.

After a few months of writing a boring and not very good system for care facilities for the elderly, I got a job managing the IT for the department of epidemiology at the Ministry of Health in Brussels. My job there was to migrate them from Windows for Workgroups to NT, while also making them aware of a newfangled thing called "security", and things like "firewalls". A major reason for hiring me instead of any old IT manager was that the IT people could get absolutely no respect from the doctors that made up most of the users there. The place was like a cat fight, with hissy fits all around, walls of silence where there was no screaming, and absolutely nothing being achieved when it came to computing. It was assumed that my medical background would make the users have some respect, and it worked. We did get some order in the house, better equipment, and more security. I loved that job, conflicts and all. The pay was shit, though, it being the government.

I stayed there a year, then went on to "Real Software", where I ended up in a warehouse at Zaventem airport trying to straighten out some really shitty VBA code for half a year. This is where I learned SQL Server pretty well. The next six months after that, I spent writing another medical communications package with some real asymmetric crypto for use in reporting in a clinical study of some new psychiatric pharmaceutical.

From Real Software, I went to C3, a small startup doing ICU software, where I wrote some of the server side code in `C++` for messaging and for script execution. This lasted until the summer of 2001, when I decided to move back to Sweden.

I got a job at Profdoc AB, one of the major vendors of medical records software in Sweden. My task was to design and implement a communication system allowing transfer of referrals, reports, lab results, and electronic prescriptions between any number of large and small client systems. If you're keeping count, this was the third such system I developed. This one

was also based on asymmetric crypto. While my first system was based on Clarion and assembler, and my second system on `C++`, this system was entirely written in Borland's Delphi, since that was the house language. It took me a little more than three years to build this system, and, as far as I know, it's working just fine still today.

In the years that followed, I went back to work as a GP part time in Sweden, while also developing a few minor applications. In 2008 I got hired as a contractor to develop a model application in `C#` for another division of Profdoc. The idea was to set up an architecture that they could follow when rebuilding some of the legacy applications they had in house. I did this for about one and a half year.

In april 2010, I suddenly got the idea of how a medical record should really work, an idea so different from how the EHR systems of today are designed that I simply had to develop a prototype just to see if it could be real. I made the first version of this system which I call "iotaMed" ("Issue Oriented Tiered Architecture for Medicine"), for the iPad in Objective-C. Ever since then I've been refining the idea and writing about it in different media.

I thought the advantages of my iotaMed would be obvious to everyone, but I was wrong. Doctors, in general, understand the idea and find it obviously better and entirely a new thing. They also understand that it is very different from what we have with current EHR systems. Everyone else, that is administrators and developers, seem not to see neither the difference, nor the point. I keep getting the remark that "we already have that", or "nobody ever asked for that", sometimes both at the same time from the same person. This attitude is both infuriating and enlightening. It clearly explains why the systems we keep getting from these developers are so useless to us. It's as if our (the doctors') way of thinking and working is so alien as to be invisible to them.

I then did what developers always do when met with a seemingly unsolvable problem; I introduced another level of indirection, or as other people say, took a step back. I realized I first need to explain what the problems are before I can offer the solution. You can't answer a question that hasn't yet been asked.

So this brings us to today and this book. It is intended to provide the right questions for a change, and maybe a few of the possible answers. I will only touch on the iotaMed design occasionally, since that isn't what this book is primarily about. Rest assured, however, that iotaMed resulted from the thinking that also formed the basis for the book.

Bibliography

[1] T Anné, L Berwouts, M Wehlou, G Berzsenyi, and F Derom. Surgical treatment of oesophageal carcinoma. experience between 1965 and 1980 (author's transl). *Acta chirurgica Belgica*, 82(4):359, 1982.

[2] Frederick P Brooks Jr. *The Mythical Man-Month, Anniversary Edition: Essays on Software Engineering*. Pearson Education, 1995.

[3] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[4] Christopher JL Murray and Julio Frenk. Ranking 37th—measuring the performance of the us health care system. *New England Journal of Medicine*, 362(2):98–99, 2010.

[5] Lawrence L Weed. *Medical records, medical education, and patient care: The problem-oriented record as a basic tool*. Press of Case Western Reserve University, 1970.

[6] Lawrence L Weed and Lincoln Weed. *Medicine in denial*. CreateSpace, 2011.