An Introduction to GNUmed

Written by the GNUmed Development Team

March, 2005

This document is intended to provide a brief introduction to the GNUmed project. It begins by reviewing some important characteristics of electronic health care records, and then examines how the GNUmed project seeks to address these issues.

TOC

Electronic Health Care Records

A <u>health care record</u> (HCR) comprises the total amount of data pertaining to a single patient. Traditionally the HCR is recorded on paper in different files stored at various locations.

The <u>electronic health care record</u> (EHCR), also known as the <u>electronic medical record</u> (EMR) or <u>electronic health record</u> (EHR), can be defined as data pertaining to one single patient stored in digital format on electronic media. Ideally, the EHCR should contain all of the information which is traditionally stored in the HCR (hence being comprehensive), but with additional characteristics which derive from its digital nature and the specific functional requirements it was designed to address (such as a particular structural approach, compliance to certain standards and use-case needs).

Datain the EHCR should preserve the nexus between context and content, while still allowing data to be easily exchanged between different applications (as stated in CEN ENV 12265).

Data Content Principles for Electronic Health Care Records

1. The record should give a faithful account of the clinician's understanding:

a) Data should be captured in terms that are found natural.

b) Conflicting statements must be allowed and uncertain and negative statements must be accepted.

c) Descriptions should be able to be given at any arbitrary level of detail and at the clinician's natural level of abstraction.

d) Once entered, data should be stored permanently. Data may be edited, but an audit trail of those edits must be kept so that the state of the data at any point in time can be reconstructed.

e) Mechanisms must be available to change the status of data (e.g. problem's activity status).

f) It should be possible to enter and view data according to various paradigms: longitudinal (e.g. progress notes), problem-oriented (POMR), SOAP-based, episode centered or report based.

2. Electronic Health Care Record Systems (EHCRS) must provide real benefits:

a) Time saving: mainly in the retrieval of clinical data, the process of issuing repeat prescriptions, referral letter writing, etc.

b) Quality of treatment: fewer gaps in available information, minimizing the decisions made with data that is either missing, or available but overlooked; automatic drug checking for interactions, contraindications, correct dosage, etc.

c) Better and more complete communication with patients and health professionals.

d) Automatic generation of reports, requests (e.g. tests) and immediate access to referrals and tests results.

e) Support for public/population health functions, including organized disease control and prevention programmes and research functions.

The EHCR may make intensive use of clinical encoding systems. The main rationale for clinical encoding is to associate concepts with precise meanings to patient data.

The architecture of the EHCR refers to the principles used to organize the logical (i.e. conceptual) structure of the health care record. As an architectural principle, it is desirable a strict separation between record and system requirements. That is, there should be a minimal coupling between the EHCR data and the information systems responsible for providing access to that data – in other words, it should be possible to easily move data between clinical information systems. Needles to say, the system architecture must be thoroughly described in readily available documentation.

Communicationof data stored in the EHCR (Electronic Data Interchange) is a major requirement. The system must allow:
a) The automatic integration of reports or other data in the record.
b) Access and support for multiple data sources (e.g. external information repositories) and media types.
c) Importation/exportation of data to/from other systems - both similar systems using the same software, and quite different ones.

Conformanceto standards is very important because it promotes such integration and interaction with third party and external systems.

A good EHCRS must be designed to provide a minimum acceptable level of reliability, security and confidentiality, providing access control services, authentication, failure tolerance, data integrity and availability.

The Problem Oriented Medical Record Paradigm

The problem-oriented medical record (POMR) was introduced by Dr. Lawrence Weed in 1968. On 1969, Dr. Weed proposed the use of "SOAP" notes for the initial recording and follow up of clinical problems as they evolve over time. SOAP stands for Subjective, Objective, Assessment and Plan.

POMR reorganizes the medical data in order to make the use of the data more systematic. It is made up of four main parts: problem list, defined data base, initial plan and progress notes. All the plan and progress notes are indexed by one or more problems in the problem list.

The problem list is the core of the POMR and acts as a table of contents for a patient's medical record. It is dynamic in nature and contains a list of clinically relevant events in a patient's life and highlights the factors which might affect clinical management. Problems which require action are called 'active problems'. Other which require no immediate action are 'inactive' problems. An active problem may become inactive and vice versa.

The data base consists of problem specific and general information. The problem specific information relates to the complaints or problems with which the patient presented, or which are elicited during history taking and physical examination. General information is common to all patients, which may include the results of routine laboratory tests.

The initial plan provides a systematic approach to the management of a patient's problem to avoid serious omissions and to capture the logic behind the doctor's actions. Each plan may include further diagnostic information, collection of information for monitoring patient progress, therapy (e.g., medications, operation, occupational therapy) and patient education.

The progress note records the patient's progress, and consists of the four parts of a SOAP note (subjective information, objective information, assessment and plan).

Free, open source software

Open source software is software that is licensed in a way which ensures that the source code is available, and that users have the right to modify that source code for their own needs and re-distribute that code to others under the same terms as the original license.

Open source software is typically created by users of the software. This is different to traditional software development methods which are usually are based on a central authority that prioritizes features and allocates limited resources to create a product. In the traditional method, technical and feature decisions are centralized.

The open source model for developing software uses the Internet to assemble and coordinate teams of developers and test users.

Feature decisions are proposed, discussed and agreed by collaborative and decentralized means. Features are made real by writing code and evaluating the results. If a new feature is needed by someone, they are free to expand the development team, which is continuously enriched by multiple kind of contributions (ideas, designs, users feedback, test data, documentation, translations...).

Because open source software can be redistributed, it is usually available at no cost and without license fees or other charges, other than the resources needed to set up and maintain an installation. However, it is entirely at the discretion of the user whether (and how much) to pay someone else to do this if the user is not able to do it him or herself.

Users of open source software fund its development directly by either working on the software themselves or contracting someone to do it. This is the key to its success and why it is revolutionizing the software industry.

Benefits:

-Lower software costs.

-More flexibility.

-More reliable products.

-Better standardization and long term stability.

-Not reliant on a single vendor.

-Faster pace of innovation.

-New projects can build on the existing base of open source code.

-Data is not "hidden away" in proprietary formats.

-Peer review increases security for systems exposed to public networks.

Risks:

a) Projects failure: Open source methods, like all software development methods, do not guarantee success. The key technical factors for success are the skills and dedication of the core developers and interface designers on the project. Open source projects can also fail for market reasons if they do not produce results faster than competitive projects.

Although no statistics are available, the failure rate for new open source projects is probably similar to the failure rate for new proprietary development projects.

b) Not deadline driven: With an open ended development team, it is impossible to reliably predict release dates. This is not a problem when deploying finished works, but can be a problem if customers become dependent on anticipated enhancements. Customers can manage this risk by active participation in the open source project concerned.

Open source and health care

Around the world, health care software is moving from hospital-centered departmental systems, to patient-centered medical records that are distributed across networks. These changes mirror and support organizational changes in the health care industry. Health care providers and their software needs are in transition.

Electronic medical records and networks are the solutions to the technical issues around coordinating the work of diverse health care professionals caring for a single person across multiple sites.

Open source software has the potential to solve some of the obstacles now being encountered in this transition:

a) Open source reference implementations of medical record standard could speed their adoption and increase interoperability in practice. For example, the Open Electronic Health Record (openEHR) project is an international attempt to develop open standards for the interchange of records between different systems. The National Library of Medicine's UMLS (Unified Medical Language System) is publicly available, and importantly, is increasingly widely used by ordinary doctors to encode and categorize medical data.

b) Open source software could reduce the issue of "Who pays?" in community health networks by eliminating per user and per site license costs and unbounded implementation and support charges. For example, one open source in health protagonist, Ignacio Valdes, co-ordinated a project to install recycled computers running Linux in a mission hospital in Guatemala, which he administers from his Houston base using secure remote access. The machines run FreeMed software which is licensed under the General Public License (GPL).

c) Practices enjoy complete control over both their software environment and their data using open source paradigms. Every application is open from top to bottom, making extensive customization and maintenance possible. Because thousands of users have submitted bug reports and fixes for the basic system software over the years, open source operating systems are famed for their reliability. Doctors, with valued IT collaborators, can build a better software system.

d) In niche markets such as specialized health care applications, software houses regularly fail and leave their customers in the lurch: and not everyone will have third-party escrow agreements written into the contract. The open source model is obviously attractive for this reason, as everyone has guaranteed access to the source code and can go on with their maintenance even if the original project development team evaporates.

Of course, realizing this potential will take time, and requires building a history of successful pilot projects and leading edge implementations.

GNUmed

GNUmed is a comprehensive, scalable software solution for paperless medical practice with emphasis on privacy protection, secure patient-centric record sharing, decision support and ease of use.

The development team is a group of practicing doctors, programmers and free software enthusiasts from around the world, committed to provide a superior, free software solution for community practice and primary health care. Using tried-and-tested technology, GNUmed software will start out by providing basic record-keeping, but will eventually cover all aspects of medical practice, and will interface well with third-party software.

With the intention of GNUmed to become a comprehensive and robust open source software package for paperless medical practice in mind, it was implemented from the outset with a client-server design. It his highly modular, which makes it very flexible and adaptable to several fields of use. This flexibility also lets doctors in several countries with different health care systems use it.

GNUmed uses PostgreSQL as a database back-end server. PostgreSQL is known to be a rock-solid open source database server and thus it is the right tool for this kind of application. Currently the user front-end is written in Python programming language and the wxWindows GUI (graphical user interface) library but in principle it is possible to create alternative front-ends - indeed, a Web-based GNUmed client application is currently being developed.

Who is GNUmed for?

GNUmed will be most suited to doctors in community practice, especially general practice, but may suit others who provide a degree of comprehensive care (general internists, pediatricians, others). GNUmed will operate on networks of a few to many users, and will support secure, remote access. GNUmed will also operate on a single computer, which makes it possible to initially examine the software, and may suit doctors or nurse clinicians serving rural or disadvantaged areas with limited infrastructure.

What is GNUmed "NOT"?

GNUmed is not currently intended for use in hospitals. It is, however, intended to interface well with hospital information systems. It is possible that some departments (such as hospital associated general ambulatory care) may be suitable venues for the use of GNUmed.

What computer system(s) will it run on?

GNUmed has been designed to operate on Unix, GNU/Linux, Mac OS X and Microsoft Windows systems. The forms in which GNUmed can currently be downloaded(Concurrent Version System repository, snapshots) are suitable for developers and others with some more advanced computing and/or programming experience. In the future, easy-to-install packages will be created which will be suitable for everyday users.

GNUmed License

One of the most widely known open source licenses license is the GNU General Public

License (GPL). Under the GNU GPL you can use, copy, modify and redistribute (or even sell) free software but the software must come with either the source files or access to the source files must be provided. If you were to modify GNUmed and then redistribute that modified version to others, the GPL requires that the modified software also be covered by the GPL. You are required to let the buyer know that they can have the source code and they have the right to use it, modify it or re-distribute it if they wish. The effect of the GPL and similar licenses is that OSS (open source software) is rarely sold and that most OSS vendors make their money instead by installing OSS and providing support to end users.

What might it cost to run?

The costs of self-sufficiency must also be kept in mind. The ability to install, configure and troubleshoot (to the point of debugging) packages on your operating system(s) is needed, as well training and support for your own office staff. Even if you are *able* to do this you may find this erodes your total time available, and causes the disruption to your medical practice activities, and your enjoyment may come more from helping to improve GNUmed than in doing all of your own support.

Most doctors will want or need skilled computer support people to do some or all of their computing support. Especially for a first implementation you would want or need to secure ample help with the hardware and network design plus software installation and configuration and training. Once your system is functioning smoothly you will likely want to structure an arrangement in which these people provide a base amount of ongoing support, with additional service on a "pay as you go" basis. Because you are unlikely to need a full-time person (at least not on an ongoing basis) it will make tremendous sense to co-ordinate your needs with those of one or more other GNUmed-based medical practices in order to make feasible a critical mass of sustainable local support.

Architecture overview

GNUmed is implemented using a *client-server architecture* with *optional middleware* components. The database back-end is provided by PostgreSQL, a largely SQL-92 compliant open source object-relational database server.

GNUmed allows distribution of *database services*. Closely related information like a person's name and address are regarded as a "service", which may or may not be hosted on the same physical server as other services. On the client side, distribution is made transparent to the developer and end user through a *"database service broker"* object which hides information about service distribution.

Tables are *normalized* to optimize storage and avoid data duplication as long as this does not deteriorate query performance too much. In order to simplify client software development, *pseudo-denormalization* is achieved through updatable views. Foreign key constraints ensure *referential integrity* of highly normalized data.

Database trigger functions ensure that business logic is enforced and that an *audit trail of all modifications to the database is kept - a matter of great importance from a clinico-legal point of view.* Deleting and updating of data is caught by triggers, and copies of the originals are kept in audit trail tables at all times, together with the author and time of the change.

The PostgreSQL server supports *secure authentication protocols* such as Kerberos as well as *secure communication protocols* between client and server(s) such as SSL. In addition, strictly confidential data items are stored in encrypted form using well-known algorithms; although the cryptographic process is performed on the client side, this is supported by the GNUmed server through a sophisticated key management. A user hierarchy can be implemented, and access to tables and procedures can be regulated through this hierarchy on the server side.

GNUmed EHCR

Strongly influenced by the Weed POMR paradigm.

Health Issue:
A health issue is a longer ranging medical condition associated with a patient. There may be several episodes relating to one health issue and there must be at least one. At times a health issue will first be appreciated by considering underlying similarities between several medically related episodes. A health issue may have a finite duration if the underlying medical condition is eventually fully resolved. It may also span the entire life of the patient such as in chronic or genetic ailments. Start and end are not directly recorded but must be derived from the start of the earliest and the end of the most recent associated episode.

Clinical Episode:

A clinical episode denotes a period of time during which the patient was under care for one particular Health Issue (medical condition). Several episodes (eg. progression, bouts, etc.) may be associated with a particular health issue. There can be several encounters per episode, and there must be at least one.

Clinical Encounter:
A one-off contact between patient and health care system (or on behalf of the patient) is called a "Clinical Encounter". The encounter table in the GNUmed database records the location, provider and subject of care as well as the type of encounter. It really is more of an administrative than a purely clinical concept. Note that it also does not fit hospital stays particularly well where it is harder to define clean boundaries between encounters. One encounter will often relate to more than one episode/health issue.

Clinical Item:
In GNUmed a medically meaningful piece of information is called "Clinical Item". Such items are typically elements of the SOAP structure such as a diagnosis, a history item, an assessment, a plan, etc. Other examples include allergies, vaccinations and test results. Several tables, both specific and generic store such items. All clinical items have a generic "narrative" field for capturing comments and clinical narrative.

```
health issue
-> several episodes
   -> several encounters
      -> several partial contacts
```

-> several items

What can GNUmed do for me today?

Firstly it must be said that these use cases doesn't reflect the current status of development of GNUmed. They should ber considered a proof of concept. Indeed GNUmed can do much more. The database schema is almost complete, rebust, flexible and extremelly well designed, containing embedded a lot of logic (as for data integrity and audit trailing). The middleware software components are in a quite mature state, allowing a reliable interaction with the backend. At this moment, the GUI is being actively developped to sew and make work together the three layers of the client (GUI, middleware and backend).

Login:
- By selecting one of the available user defined profiles
  - 'gnumed at localhost'
    - .host: localhost
    - .port: 5432
    - .data base: gnumed
- User/password authentication
  - user: any-doc
  - password: any-doc

Workspace:
- Plugins are auto-discovered and loaded during startup.
- Main window title displays the enviroment information (* patient not updated on selection)
- Plugins are activated from menu bar and notebook tabs:
  - .allergies, request, vaccinations, BMI, BDC, XDT, STIKO, demographics editor.

Patient management:
- Patient search and selection (by last names: Kirk, Chapel, Spock)
- Activate a particular plug-in for that patient (allergies)
- Import patient from German-style XDT file (BDT/GDT)

Vaccination status:
- display indications, active regimes and missing vaccinations
- display administered vaccinations for a selected indication
- display details for an administered or missing vaccination
- *enter administered vaccination
- generate vaccination status table (by using ascii exporter tool)

EHCR:
- load EMR tree and browse its elements:
  - health issue -> episode -> encounter
- display the information associated with the selected EMR element: descriptive name, initial and end dates, narratives, progress notes and clinical items: vaccinations, allergies, test results
- display episode contextual menu with planned actions (implement in a close

future)
-export to text file (using exporter)

Progress notes:
- -load multisash based notes input plugin
- -display patient's problems (issues and episodes)
- -create progress note for an episode
- -create unassociated progress note
- -create progress note from issue
  - -episode selection
  - *episode creation (problem list not refreshing)

Medical documents :
- -scan from paper
- -import from disk
- -associate with a patient
- -display on screen

Laboratory tests:
- -create new lab request (German style)
- -display lab requests history
- -import lab data result files (German style)
- -display import errors
- -sign off as-yet unreviewed (e.g. new) lab results
- -display lab results for a patient

GNUmed roadmap:

Status definitions:

-draft:

.some ideas are floating about in the list or the Wiki

.some code may reside in test-area, not part of GNUmed yet

-alpha:

.first implementation, not completely functional, almost untested

-pre-beta:

.advanced implementation, almost functional, developer-tested

-beta:

.supposed to be functional, exhaustively developer-tested.

-final (release quality:

.tested by a large number of users, (supposed) to contain no severe bugs

Description	Maintainer	Hackers	Status
Version 0.1 is to have			pre-alpha
patient input		Ian Haywood	pre-beta
patient modification		Ian Haywood	pre-beta
patient search	Karsten Hilbert	Karsten Hilbert	beta
progress notes input	Ian Haywood , Carlos Moro		
progress notes viewer	Carlos Moro	Carlos Moro, Karsten Hilbert	beta
progress notes ASCII export	Carlos Moro	Carlos Moro, Karsten Hilbert	pre-beta
third party application connectivity	Karsten Hilbert		beta
simple ConfigRegistry plus complete default config data	Hilmar Berger		beta
Version 0.1 is also likely to have			pre-alpha
path results		Karsten Hilbert	alpha
vaccinations	Karsten Hilbert		beta
referrals handling		Ian Haywood, Karsten Hilbert	alpha
requests handling			
date & time input widget	Karsten Hilbert		beta

post-0.1			
document archive		Karsten Hilbert	beta
allergies			
drug information browser		Hilmar Berger	
prescription		Horst Herb (please confirm)	
past history			
contacts		Ian Haywood	
future			
recalls			

Resources:

-GNUmed:
- -Home, http://www.GNUmed.org
- -German community, http://www.GNUmed.de

-Wiki, http://salaam.homeunix.com/twiki/bin/view/Gnumed/WebHome
-Debian-Med:
-http://www.debian.org/devel/debian-med/
-http://people.debian.org/~tille/talks
-http://www.oshca.org/

-http://www.openhealth.com/en/healthlinks.html

-http://www.minoru-development.com/

-http://www.openhealth.com/en/healthcare.html

-http://www.informatics-review.com/open.html

-http://www.linuxmednews.com

-Lawrence L. Weed, M.D.; Medical records that guide and teach; (orig.) N Engl J Med 1968; 593-600; (reedited) M.D. Computing 1993.
-CEN WGII Prague; 'POMR and SOAP'.
-College of Veterinary Medicine (Washington State University); 'Ancillary Notes on POMR'.
-Westerhof Henk Dr.; 'Episodes of Care in the New Dutch GP Systems'.
-Ceusters W. Dr. (President of PROREC-BE); 'Good Characteristics for describing Electronic health care Records and Systems'.
-Ho L.M.; 'The application of a computerized POMR and its impact on patient care'; International Journal of Medical Informatics 55 (1999).
-Mohtadi Nikan; 'Le POMR'.
-Vergil Skee, MD; 'The Endangered Medical Record'.