Medical Informatics Europe 2002
G Surján et al. (eds)
IOS Press 2002

A one-step approach to data retrieval, analysis and documentation

Johan KARLSSON, Patrik EKLUND

Umeå University, Department of Computing Science, SE-901 87 Umeå, Sweden
{johank, peklund}@cs.umu.se

Abstract. In this paper we propose a system architecture to support data retrieval from patient records, together with data analysis, and report generation. Moving from retrieval all the way to documentation should be done as in a pipeline, and computer support must minimize manual steps in this knowledge refinement scenario. A case study is drawn from urology for surgery on patients with lower urinary tract symptoms.

1. Introduction

Data mining and warehousing including knowledge elicitation also in form of automatic statistical reporting and document generation is a broadly accepted vision in management and refinement of health information. Several partial approaches are presented in the literature [1,2,3]. We argue that software engineering approaches involving a careful analysis of software modularity and module interaction are prerequisites for successful broader developments. Further, if such systems are to have real impact, several interrelated clinics must be involved with contributions to patient information handling. Further, information must be understood and utilized in a workflow context [4], so that data warehousing projects can support both quality assurance as well as guideline development needs. Thus information utility is prescribed by requirements and needs defined by respective health care unit. Our information management experience stems mainly from workflow and guideline development within family medicine [5,6], geriatrics [7], oncology/radiation therapy [8] and urology [2]. Various approaches to data analysis, in particular for laboratory medicine, can be found in [9]. Software aspects of guideline implementation and corresponding mobile extensions are found in [10].

In order to meet requirements for a broad approach to information management and knowledge elicitation, a careful analysis of generic software architectures is needed especially as software components must adapt to changes e.g. in database structures and information formats. Database structures are complex and usually updated along with changes and reorganisations made with respect to working practise, i.e. modifications are seen in workflows. It is also important to observe the fruitful interaction between information refinement needs and database structure update requirements within clinical practise. From software development point of view, generic queries and terminology based information must be carefully handled to adapt to such needs and requirements. Software engineering techniques must also handle client/server solutions as well as middleware techniques.

Transparency of information is of utmost importance. Database structures should be visible and explained to the user. An information retrieval system should include detailed advice, support and suggestions to enable extensive and elaborate use of information and information structures available.

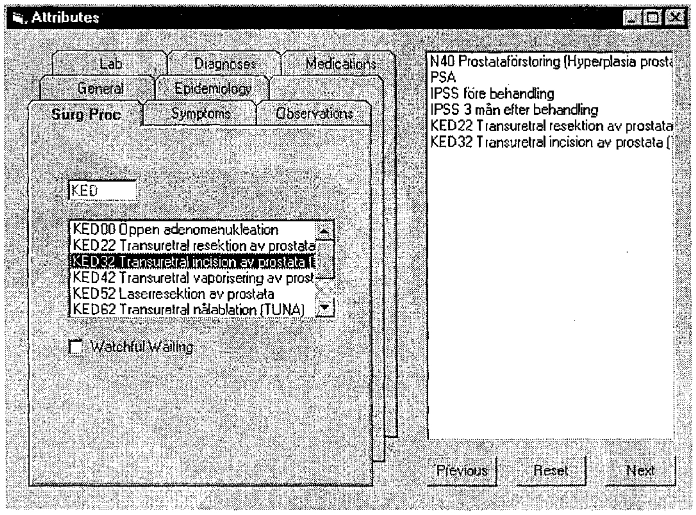
In this paper we present a proposal for a software architecture as a basis for data warehousing and knowledge elicitation as required by information management in workflow contexts. We build upon previous developments [2,11,12] on data warehousing for a regional patient record system. Companion papers on guideline implementation and workflow [7,10] are related to these information management developments.

2. A typical usage scenario

As an illuminating case study we will use follow-up concerning quality of life after surgery on patients with enlarged prostates [2]. LUTS (Lower Urinary Tract Symptoms) patients have symptoms such as a weak or interrupted flow of urine, pain or a burning sensation during urination, and urinary leakage. Patients may have conditions such as prostate cancer, enlarged prostate (Benign Prostatic Hyperplasia, BPH), neurological diseases and diabetes. Treatment and decisions concerning surgery depend on the diagnosis and degree of symptoms. Information can be retrieved within an external retrieval system connected to the patient record system.

Data included in this case study, for surgery on patients with enlarged prostate, covered attributes like PSA (Prostate Specific Antigen), IPSS (International Prostate Symptom Scoring) given by the patient condition before and after the operation, flow voided volume (urine flow), amount of urine that remains in the bladder after urination, blood in urine as a yes or no, and the IPSS difference as measured after the operation. Surgery methods included are transurethral resection (TURP) and the less invasive transurethral incision (TUIP) of the prostate. A typical question is the following: How has IPSS values improved after operation when comparing different surgery methods?

In the retrieval phase, search criteria could include diagnosis (BPH), IPSS values before and three months after surgery, together with surgical methods being either TURP or TUIP.



KED

KED00 Öppen adenomenukleation
KED22 Transuretral resektion av prostata
KED32 Transuretral incision av prostata (
KED42 Transuretral vaporisering av prost
KED52 Laserresektion av prostata
KED62 Transuretral nålablation (TUNA)

N40 Prostataförstoring (Hyperplasia prost
PSA
IPSS före behandling
IPSS 3 mån efter behandling
KED22 Transuretral resektion av prostata
KED32 Transuretral incision av prostata (

Figure 1

Search criteria can be converted to database queries and the server returns a data set for these attributes. The conversion is not implemented in our prototype system. However, queries to a regional patient record system have been tested and found feasible in a warehousing context. A data set was retrieved by 'hand-picking' records from the database. Statistical summaries and data analysis was performed on this data set in the conventional way. Computation e.g. of the p-value in comparison of IPSS improvement mean values gives in this case 0.024. Other similar investigations could involve additional diagnosis information such as presence of dementia, or involving various symptom classes. Presentation of rudimentary statistical results and the computations of these can then rather easily be included for automatic generation in the pipeline.

3. Architecture

One important goal in software development is to make the developed applications easy to maintain. This is not a trivial task, especially in larger, more complex software systems. In fact, the cost of software maintenance has been steadily increasing. Figures vary, but during the 1980's up to 60 percent of the software budget in many companies was spent on maintenance [13]. This motivates special attention to building easily maintainable systems, in general software development as well as in medical informatics.

One reason for this lack of maintainability is that many earlier systems were developed using procedural constructs (typical languages include C, Visual Basic). These languages tend to have no or weak constructs for information hiding. By hiding implementation details in a module, other modules are safe from any internal changes in the module.

Object oriented systems tend to have a high degree of modularisation. The software systems are defined in terms of objects and their capabilities (what the object can do) by the means of object oriented analysis and modelling. The question of how the object actually achieves the task is irrelevant to outside objects/modules. This isolates parts that are likely to change. Typical languages include C++ and Java.

Applications that use relational databases are vulnerable to changes in the structure of the database or in the format of individual fields. This is certainly true for applications that use electronic patient records (who typically are based on some form of relational database). The structure of the relational databases storing the (in itself hard to interpret) patient data for the regional electronic patient record is quite complex (a clinic level database consists of almost 300 tables).

If the EPR vendor is changed (or if a restructuring of the databases is needed), most (if not all) applications that use data from the electronic patient record would have to be updated. A similar situation occurs if the operating system is changed, rendering most existing applications incompatible or in need of maintenance.

When establishing the data warehouse, major table restructuring might be needed. By reducing the number of tables, we could see a decrease in the necessary table conjoinings while searching the databases, thereby increasing the performance. Naturally, the database structure of the patient record system is designed to enable patient specific access to data. Also, data imported into the data warehouse might need some scrubbing (for example by coding the existing data even further using standards).

It is clear that applications that make use of data found in the patient record would benefit from a higher degree of adaptability to changes in the database (or indeed of platform). The platform issue can be partially solved by using a platform-independent language such as Java. A smooth migration to another platform is in practice unlikely for a more complex application because of minor inconsistencies in the Java virtual machine and standard utility libraries (sometimes with native system calls) implementations.

One way to remove knowledge of the patient record's internal database structure from the client applications would be to use a middle layer and call its API's for the patient information. The patient information could then be returned as classes representing the concepts (for example Patient, Appointment etc).

While this certainly removes knowledge of the database structure from the clients, they are limited to objects/structures supplied by the API. This approach is suitable to specific higher-level applications such as time-booking software. However, if the objects are not to be used as mere holders of data but actually contain capabilities, it is hard to make them general. This is because the potentially different problem domains (i.e. uses of the object) would create a bloated and unmanageable object.

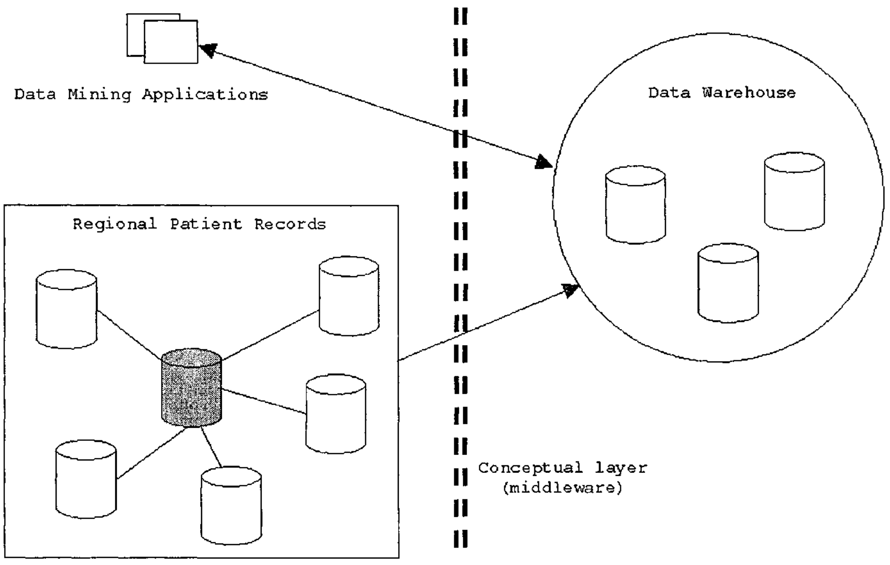


Figure 2

Data mining medical knowledge in the form of patient records does require ontologies and semantic information. Therefore we suggest for the retrieval layer in [11] a solution where knowledge of the database structure and its content is stored in a separate middle layer. To enable a general gateway to the data warehouse, we need to have knowledge of the contents without constraining ourselves to a fixed number of data structures (or objects).

While on a higher level, it would certainly be motivated with API-access to the databases (methods that return objects containing data as well as capabilities), at a more basic level a dynamic description of the contents and the meaning/context of the data is preferable. The string "N40" does not in it self say anything. Additional information give its meaning and context. By observing that "N40" really is an ICD-10 code, we can conclude that it means hyperplasia of prostate. Additional information, such as the time of entry in the patient record, source (the doctor or expert system), which steps proceeded the entry (i.e. the guidelines used), patient history (earlier examinations and diagnoses), what range of values are normal and indeed possible? Indeed, the description of the data could also contain links to the workflow used at the clinic at the time. One study of workflow in radiation theraphy was reported in [8, 12]. Since it is hard to add further information/descriptions to the existing clinical databases, it should be stored independently.

The regional patient record unfortunately does not store all this information and when it does, information is not easily accessible. Reorganizing and adding additional knowledge is left to the client application programmers. If this could be done on a lower level than client-side programming, data lookup would be simplified. The need to know the specifics of the database structure of the EPR would be less necessary.

Obviously, great care must be taken choosing the standards to represent the various elements in the patient journal (for example, prescribed drugs could be coded using ATC, diagnoses with ICD-10 etc.). The representation should also be easily accessible and possible to parse by most clients. For data exchange between separate systems, the standard format XML should be suitable.

With a graphical presentation of the required content in the database, the user of a data mining application could piece together a query. Together with the representation of the database content and structure, the middleware layer then constructs a SQL query that is given to the database. The results are then in turn encoded with the same concepts to allow easy parsing when returned to the client application, where further data processing takes place. This data processing could be done by using other tools such as spreadsheet software or by internally doing statistics and/or machine learning. The end result, as suggested in [11], would be a produced expert system complete with a user interface.

References

[1] F. Banhart and H. Klaeren, A Graphical Query Generator for Clinical Research Databases. Methods Inf Med **34** (1995) 328-339.

[2] J. Karlsson, P. Eklund, C.-G. Hallgren and J. Sjödin. Data Warehousing as a Basis for Web-based Documentation of Data Mining and Analysis. Medical Informatics Europe, '99 (Eds. P. Kokol, B. Zupan, J. Stare, M. Premik and R. Engelbrecht), IOS Press, 423-427, 1999.

[3] R. K. Taira, D. B. Johnson, V. Bhushan, M. Rivera, C. Wong, L. Huang, D. R. Aberle, M. Greaves and J. G. Goldin, A Concept-based Retrieval System for Thoracic Radiology. J Digit Imaging **9** (1996) 25-36.

[4] B. Heller, M. Löffler, M. Musen and M. Stefanelli (eds.), Computer-Base Support for Clinical Guidelines and Protocols, Proceedings of EWGLP 2000. IOS Press, 2001.

[5] National Institutes of Health. The Sixth Report of the Joint National Committee on Prevention, Detection, Evaluation, and Treatment of High Blood Pressure. National Institutes of Health, NIH publication no. 98-4080 edition, 1997.

[6] M. Persson, J. Bohlin and P. Eklund, Development and Maintenance of Guideline Based Decision Support for Pharmacological Treatment of Hypertension. Computer Methods and Programs in Medicine, **61** (2000) 209-219.

[7] H. Lindgren, P. Eklund and Sture Eriksson, Clinical Decision Support System in Dementia Care. MIE2002, in this volume.

[8] L. Franzén, M. Karlsson, G. Duvefjäll, P. Eklund, I. Lax, K. Nordlinder, I. Näslund, C. Svedberg and B. Zackrisson, Information Technology in Radiation Therapy (abstract). ESTRO Meeting in Physics for Clinical Radiotherapy, Germany, 1999.

[9] P. Eklund and J. Forsström, Computational Intelligence for Laboratory Information Systems. Scand J Clin Lab Invest **55** Suppl. 222 (1995), 21-30.

[10] P. Eklund, S. Eriksson, J. Karlsson, H. Lindgren and A. Näslund, Software development and maintenance strategies for guideline implementation. Proc. EUNITE-Workshop "Intelligent Systems in Patient Care" (Ed. K.-P. Adlassnig), Austrian Computer Society, 2001, 26-34.

[11] J. Karlsson and P. Eklund, Data Mining and Structuring of Executable Data Analysis Reports: Guideline Development in a Narrow Sense. Proc. MIE2000 and GMDS2000, Medical Infobahn for Europe (Eds. A. Hasman, B. Blobel, J. Dudeck, R. Engelbrecht, G. Gell, H.-U. Prokosch), IOS Press, 790-794, 2000.

[12] J. Karlsson and P. Eklund, Workflow Design as a Basis for Component Interaction. Proc. Medinfo 2001, London, UK, 2-5 September, 2001, 1158-1160.

[13] R. S. Pressman, Software engineering – a practitioners approach. McGraw-Hill, 1992.