Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 100 (2016) 498 – 506

Procedia Computer Science

Conference on ENTERprise Information Systems / International Conference on Project MANagement / Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2016, October 5-7, 2016

# Towards a more effective hospital: helping health professionals to learn from their own practice by developing an easy to use clinical processes querying language

Juris Barzdins[a], Mikus Grasmanis[b], Edgars Rencis[b,*], Agris Sostaks[b], Aslak Steinsbekk[c]

[a]*Centre of Health Management and Informatics, Faculty of Medicine, University of Latvia, 19 Raina blvd., Riga, LV-1586, Latvia*
[b]*Institute of Mathematics and Computer Science, University of Latvia, 29 Raina blvd., Riga, LV-1459, Latvia*
[c]*The Department of Public Health and General Practice, Faculty of Medicine, Norwegian University of Science and Technology, Håkon Jarls gate 11, 7030 Trondheim, Norway*

**Abstract**

Application of complex socio-technical systems theory to optimization of clinical processes in hospitals highlights the importance of the acceptance and promotion of responsible autonomy among health professionals. Therefore the independent ability for clinicians to search for answers to questions which are outside the scope of pre-made reports is important. However, the ad-hoc data querying process is slow and error prone due to inability of health professionals to access data directly without involving IT experts. The problem lies in the complexity of means used to query data. We propose a new natural language- and star ontology-based ad-hoc data querying approach which reduces the steep learning curve required to be able to query data. The proposed approach would significantly decrease the time needed to master the ad-hoc data querying and to obtain direct access to data by health professionals.



Peer-review under responsibility of the organizing committee of CENTERIS 2016

*Keywords:* Socio-technical system; hospital management; ad-hoc querying; star ontologies; controlled natural language; hierarchical data.

* Corresponding author. Tel.: +371 26462224.
*E-mail address:* edgars.rencis@lumii.lv


Peer-review under responsibility of the organizing committee of CENTERIS 2016
doi:10.1016/j.procs.2016.09.188

## 1. Responsible autonomy for healthcare professionals

Successful management of large multispecialty hospitals requires managerial, organizational, technological and clinical approaches to be linked together. Failure to have a joined optimization of the technical system and social system could lead to paradoxical fall in the productivity and an increased absenteeism despite the significant improvements of technology which have been described in a widely used socio-technical systems theory[1].

One of the major technological improvements in hospitals – the change from paper based recording of clinical processes to electronic records with the possibility for relatively easy traceable data on the patient flow - can also create a certain tension in the social system of a hospital. Hospital operations thus become more transparent for managers with the possibility of enhanced managerial control on clinical process. On the other hand, the management efforts to make clinical process more effective can be perceived by physicians as a threat to their professional autonomy[2–8]. It is widely recognized that clinicians performance is the best when "being invited, empowered and nurtured rather than directed, micromanaged and controlled through a hierarchy"[9]. It is the physician, not the manager that remains responsible for the results of the treatment of the concrete patient, including decisions regarding the use of medical technologies, treatment modalities, localization and duration which also drive the distribution of limited resources of the whole organization[10]. The fact that so far there has been little evidence that implementation of the health information technologies (HIT) has led to healthcare cost savings[11] partially could be attributed to violation of some principles essential for the optimization of complex socio-technical systems.

No single model exists for optimizations of socio-technical systems. However, some universal principles could be underlined: acceptance and promotion of responsible autonomy so that individuals take responsibility for the performance and outcomes of concrete decisions; development of the ability to adapt at the individual level in order to distribute optimization; performers themselves set performance indicators; improved performers awareness of the ultimate systemic goal of the task[1,12–14].

Thus, applying these principles to the introduction of HIT in hospitals at a clinical level "dominated by individuals with professional backgrounds, valuing self-governance and autonomy"[15] could lead to a situation that health professionals feel more empowered in their daily work. Some authors have argued that empowered self-regulated work groups can be a complement, or even a substitute of traditional management in those situations when there is a lack of efficiency[16–18]. However, the application of the mentioned principles for optimizations of complex socio-technical systems have been limited in healthcare, and "a bottom-up strategy led by clinicians is badly needed to balance the predominantly top-down approaches which frequently result in only modest improvements which are difficult to sustain[9].

To support the bottom-up strategy, a particular interest is to use HIT in order to provide health professionals with systemic knowledge about their work with concrete patients. This includes focused and process oriented analysis, including a possibility to clarify particularities and patterns by querying data accumulated in hospital informational systems. Consequently, an increased transparency of clinical processes, which is not limited to managers, increases the autonomy and responsibility of clinicians.

Compared to managers, health professionals until now have had considerably less possibilities to explore the information stored in databases and search for answers to questions which are outside the scope of pre-made reports. The complexity of hospital databases and data heterogeneity demands either advanced programming and data processing skills, or available, directly subordinated data professionals to query data. Currently these two options refer to managers rather than physicians.

The proposed solution which would allow medical professionals to widen their responsible autonomy also would enhance the effectiveness of hospital as a complex socio-technical system.

## 2. State of the art in ad-hoc data querying for non-programmers

Direct access to data by medical professionals would be a solution, but why there are no ad-hoc data querying tools for them (non-programmers)? The problem is that non-programmers do not possess the required skills to formulate queries by themselves, because of the complexity of query languages used to retrieve answers from data stores. There are three main problems: how to describe data to be easily perceived by non-programmers; how to

query data simply enough for non-programmers; how to perform query efficiently enough in order to get an answer in a reasonable time.

The SQL (SEQUEL) language is the *de facto* standard of querying relational databases where most of data are stored at the moment. However, it turned out that the way data are stored and retrieved in the relational databases was too complicated for non-programmers. There are similar languages to SQL, e.g. SPARQL for RDF ontologies. They require a very precise formulation of the textual query (both syntax and semantics) and deep knowledge of the underlying technology, thus making them too sophisticated to learn and use. Therefore there have been attempts to make wrappers for these languages, e.g. graphical query builders like Graphical Query Designer for SQL Server, ViziQuer[19] for SPARQL and RDF databases, or form-based tools using wizards and standard GUI elements (e.g. tables and lists) like SAP Quick Viewer SQVI. There are also other proposals which provide the means for direct data access. One of the most well-known approaches is Self-Service Business Intelligence (SSBI) which was proposed by Microsoft[20]. It provides a rich set of tools (Power BI) allowing the end-user to build sophisticated data visualizations and make data analysis mainly through spreadsheet applications. IBM Watson Analytics is another giant which attacks this problem. Yet, there is a significant drawback of the mentioned approaches, namely, a steep learning curve shall be overcome in order to learn a new query language and to understand the way data are stored.

A viable option to query data for medical professionals is a natural language, more precisely, natural language interfaces to databases (NLIDB-s). A lot of work has been done in this area[21-25]. However, formulation of the precise query itself is a hard problem for users without mathematical background, and there are lots of problems in the understanding of complex queries. In order to make an NLIDB system usable by non-programmers it is necessary to solve the problem of linguistic coverage. It is very important to explain to users what the database "knows" and what can be asked. Database schemas used by IT experts (like ER models) are too complex and contain too many technical details to be useful for the explanation of the underlying data. Computers, on the other hand, cannot properly understand what users mean by their queries because of richness and ambiguity of the natural language. In order to achieve a consensus between the user and the computer an intermediate representation of data schema is needed. Ontologies define concepts, their properties and relationships which can be used by user. Although an ontology conceals some technical details, it requires the understanding of basic entity-relationship model principles. Thus, the traditional NLIDB approaches have not reached wide usage, at least not for deep querying with nontrivial calculations. Therefore other approaches have to be studied.

## 3. Constructing easily understandable data ontologies

When thinking about the platform on which data storage is to be based, IT specialists often propose a relational database as a way of storing the data and the SQL as a query language. This is, indeed, a very natural solution if we keep in mind that the ontology will have to be implemented later. Therefore, the ontology is represented in the form of ER model. However, the ER model is hardly ever granular, and, therefore, it is not the best way to depict the data in the manner that is understandable for end-users who are not IT specialists.

In the case of hospital data the problem with more understandable data storage format may be simpler, because the data are naturally stored in the so-called patient cards which are filled in by hand. We can say that all hospital data are *sliced* in slices, where each slice represents a particular patient together with his/her related data (his/her episodes in hospital, movements, diagnoses, operations, etc.) that are laid out in a hierarchical structure. Such a division is clearly understandable by domain experts. Therefore it is the most natural way of representing data sets which can be naturally divided in hierarchical slices this way. We call such ontologies, whose data can be divided in hierarchical slices, *star ontologies*. The concept of the star ontology is already introduced by J. Barzdins et al.[26-28] where it is shown that a star ontology can serve as a very good format for storing data in a user-friendly manner. Concepts of the star ontology (classes and attributes) in case of hospital data are directly linked to the concepts of a hospital patient card. Thus it would be much more understandable to clinicians and managers of hospital than the traditionally-used ER model where these concepts are mixed together. A version of the star ontology corresponding to the hospital patient card can be seen in Fig. 1.

Of course, some doubt could arise whether this is a case of good coincidence that happened only in this particular example, or the same pattern could be found in other similar cases too. Our experience shows that the composition-like structure is an intrinsic feature of such type of documents themselves, and therefore the star data ontologies must

be one of the most natural ways of depicting such documents in general, not just in this particular case. Star ontologies can be useful also in those cases when there are no paper-format input forms in a company (all information is entered electronically via input dialogs). Quite a wide class of ontologies can be reduced to star ontologies, which are much more understandable for end-users than the real underlying ER model.

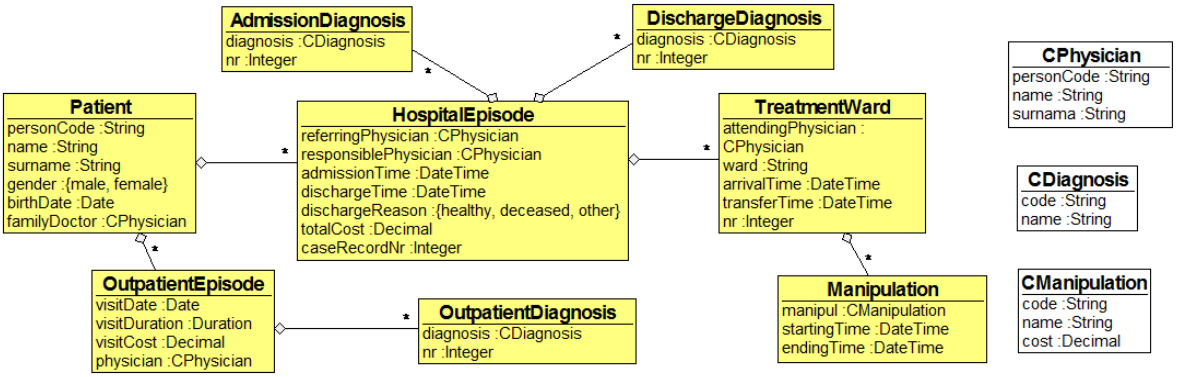


Fig. 1. A simplified star ontology used in Riga Children's Clinical University Hospital (RCCUH).

Since star ontologies are granular (meaning – they can be naturally divided in slices[27,28]), they provide a possibility for us to develop a new kind of querying language that is very easily understandable by domain experts who want to formulate their ad-hoc queries. We demonstrate this language in Section 4. Although it is not as powerful as SQL, it possesses three very important features, namely, simplicity, coverage of a sufficiently wide class of queries and a very efficient implementation.

## 4. Towards a natural language-based query language

One of the benefits of storing data in a star ontology is the ability to depict the data scheme in such a way that can be understood by domain experts. Another benefit is a possibility of developing such a language for querying data that can also be understood by the same domain experts. If data were stored in a relational database, they would typically be queried by the SQL language that is very technical. In case of star ontologies we have developed a natural language-based query language, thus it is more convenient for everyday use by non-IT specialists. We have based the language on several typical constructs of a natural language, and we have then supplemented those constructs with some *foreign* parts like expressions and variables in some cases. Our main goal here is to minimize the steep learning curve that is typically needed when learning a new query language. We assume that a natural language would be the best solution here, because users are already familiar with it, and we try to come as close to the natural language as possible. Our query language is still quite controlled, of course, but we believe it is nevertheless much more suitable for clinicians and managers of hospital than SQL.

Now let us introduce an example query – *count Patients, who have at least one HospitalEpisode, which has Manipulation with manipul.code=02078.* This natural language sentence is understandable by domain expert. Let us now inspect a bit more complicated query: *count Patients, who have at least one HospitalEpisode, which has at least one TreatmentWard, which has at least one Manipulation with manipul.code=02078.* This sentence may cause a certain ambiguity as it is not clear whether the asked Manipulation refers to HospitalEpisode, or to TreatmentWard. It could be used in both meanings. In other words, relative pronouns such as "who" and "which" not always give us accurate understanding of what we relate to. To cope with such situations we introduce a concept of so-called short name in our controlled natural language. Formally, the short name is a variable over instances of the given class – *count Patients p, where exists p.HospitalEpisode e, where exists e.TreatmentWard t, where exists e.Manipulation m, where m.manipul.code=02078.* We have also unified other components of the natural language,

e.g. we use the keyword "where" instead of "who", "which" and "with", and the keyword "exists" instead of "have/has at least one". The dot notation after the short name must be perceived as the "of" relation – *count Patient p, where exists HospitalEpisode e of Patient p, where…*

Let us now introduce some basic notations that we will use to describe the query language. We will use the terms *parent class* and *child class* to refer to classes that are higher or lower in the "have" hierarchy. For example, the class "TreatmentWard" has two parent classes – "HospitalEpisode" (direct parent) and "Patient" (further ancestor) and one child class "Manipulation". If *x* is an instance of the class "TreatmentWard", then its parent instances will be denoted as *x.HospitalEpisode* and *x.Patient*. In both cases they denote exactly one instance, i.e. that of the class "HospitalEpisode" and of the class "Patient", respectively. We use the same dot notation also for accessing instances of child classes, but in this case we obtain a set of instances. For example, *x.Manipulation* would be a set of manipulations reachable from the given treatment ward *x*.

If *AClass* is an arbitrary class of the ontology, we will use the term *AClass attribute expression* to denote attribute expressions of both *AClass* itself and all of its parent classes (we assume here that parents and children share no common attribute names). Since there can be many children's instances for the given *AClass* instance, we will be able to access these instances by introducing quantors *exists*/*notexists* later.

Queries are to be written in a controlled natural language and are based on seven sentence constructs which we call templates. The main part of the templates is the so-called selection condition which is a selection condition over instances of the given class. We assume that selection conditions are written in a natural language. The sentence templates described in this section can be understood without knowing the precise syntax of selection conditions.

T1. COUNT AClass [x] WHERE <selection condition>

Semantics: counts instances of AClass, which satisfy the selection condition. Example:

- COUNT Patients, WHERE EXISTS HospitalEpisode, WHERE referringPhysician=familyDoctor (count of patients who have been referred to hospital by their family doctors);

T2. {SUM/MAX/MIN/AVG/MOST} <attribute expression> FROM AClass [x] WHERE <selection condition>

Semantics: selects instances of AClass, which satisfy the selection condition, calculates the attribute expression for each of these instances obtaining a list to which the specified aggregate function is then applied. Example:

- SUM totalCost FROM HospitalEpisodes, WHERE dischargeReason=healthy AND birthDate.year()=2012 (how much successful treatments of patients born in 2012 have cost);

T3. SELECT FROM AClass [x] WHERE <selection condition> ATTRIBUTE <attribute expression> ALL DISTINCT VALUES

The semantics is obvious. Example:

- SELECT FROM HospitalEpisodes, WHERE dischargeReason=deceased, ATTRIBUTE responsiblePhysician.surname ALL DISTINCT VALUES;

T4. SHOW [n/ALL] AClass WHERE <selection condition>

The semantics: shows n or all instances of AClass which satisfy the selection condition.

T5. FULLSHOW [n/all] AClass WHERE <selection condition>

The semantics: the same as "show", but shows also the child class instances attached to the selected AClass instances.

T6. SELECT AClass x WHERE <selection condition>, DEFINE TABLE <x-expr'1> [(COLUMN C1], …, <x-expr'n> [(COLUMN Cn)] [, KEEP ROWS WHERE <Ci selection condition>] [, SORT [ASCENDING/DESCENDING] BY COLUMN Ci] [, LEAVE [FIRST/LAST] n ROWS]

The semantics: selects all instances of AClass, which satisfy the selection condition, then makes a table with columns C1 to Cn, which for every selected AClass instance x contains an individual row, which in column C1 contains the value of the <x-expr'1>, …, in column Cn contains the value of the <x-expr'n>. Then it is possible to perform some basic operations with the table like filtering out unnecessary rows, sorting the rows by values of some column and then taking just the first or the last n rows from the table. Example:

- SELECT HospitalEpisodes x, WHERE dischargeReason=deceased, DEFINE TABLE x.surname (COLUMN Surname), x.dischargeTime.date() (COLUMN Death_date), (COUNT x.Manipulation, WHERE manipul.code=02078) (COLUMN Count_02078), (SUM manipul.cost FROM x.Manipulation, WHERE manipul.code=02078) (COLUMN cost_02078);

T7. There are two more cases in the definition of the table, where table rows come from some other source, not being instances of some class. Being very similar, these two cases form two subtemplates of the last template:

a) SELECT FROM AClass [a] WHERE <selection condition> ATTRIBUTE <attribute expression> ALL DISTINCT VALUES x, DEFINE TABLE…

b) SELECT FROM INTERVAL (start-end) ALL VALUES x, DEFINE TABLE…

The semantics of both cases is obvious. Examples:

- SELECT FROM TreatmentWards ATTRIBUTE ward ALL DISTINCT VALUES x, DEFINE TABLE x (COLUMN Ward), (SUM manipul.cost FROM Manipulations, WHERE ward=x) (COLUMN Cost);
- SELECT FROM INTERVAL (1-12) ALL DISTINCT VALUES x, DEFINE TABLE x (COLUMN Month), (COUNT HospitalEpisodes, WHERE admissionTime.month()=x) (COLUMN Episode_count) (MOST diagnosis.code FROM AdmissionDiagnoses, WHERE nr=1 AND admissionTime.month()=x) (COLUMN Most_frequent_main_diagnosis).

## 5. Proof of concepts

We showed in Section 3 that star ontologies cover quite a wide spectrum of practically important data ontologies including hospital data schemas from the point of view of patients and physicians. In this section we will inspect other important aspects of the proposed query language, namely, whether it is expressive enough for practical usage and simple enough for understanding by domain experts and whether it has sufficiently efficient implementation.

The first aspect consists of two parts. The expressiveness of the query language was demonstrated by turning it into a working language for RCCUH when annual reports had to be generated. It turned out to be expressive enough for this task. During the two year period, when it was used for report generation, the language underwent a continuous improvement process. It was important to achieve such a level that managers of wards are able to formulate themselves all the necessary queries without referring to a programmer with every 5th or 10th query to write the desired query in SQL. Results of such queries were either single numbers or data fields, or tables of data fields. In case of tables, our aim was to generate a table containing all the necessary data that can then be exported to a spreadsheet or an R tool (a tool for statistical analysis).

The second part of the first abovementioned aspect regards the possibility for domain experts to learn the language. To test this aspect we performed both individual experiments with potential end-users and group tests. General situation from the language teaching point of view was best demonstrated in an experiment with experienced nurses who study to obtain Master's Degree at the Faculty of Medicine, University of Latvia. We presented a two hour long lecture about the language and the tool for querying the data. One third of that time was devoted to explanation of the underlying data ontology (what is a class, an attribute, etc.). Afterwards the language was explained on examples, and homework was given to test the level of understanding. The homework consisted of two parts. Firstly, students had to understand sentences written in our controlled natural language and to write them in a good really natural language. Secondly, they had to do the task in the opposite direction – changing natural language sentences into our formal language. The results obtained from this experiment can be seen in Table 2. Main conclusion is that another two hour long lecture after the completion of homework would be beneficial for a better understanding of the proposed query language.

Table 1. The results of the experiment.

| | Number of students succeeded (n=15) | | | | |
|---|---|---|---|---|---|
| Task execution level (%) | ≥90 | ≥75<90 | ≥50<75 | ≥25<50 | <25 |
| Understanding of queries | 9 | 2 | 2 | 1 | 1 |
| Writing of queries | 3 | 5 | 2 | 3 | 2 |

A very important factor related to the teaching process is the fact that the underlying data ontology was anonymized (from the point of view of patients, physicians and wards), but real. Our experience shows that students being domain experts of this ontology rapidly got very interested in the querying process and started to perceive this as a game. This fact had a beneficial impact on the learning process.

Finally, let us talk a bit about the ability to implement the language efficiently. To test the performance of our implementation we gathered 120 typical query examples from real annual report analysis of intensive care ward and from discussions with managers of other wards. The complexity of these queries is similar to those demonstrated in Section 4. The volume of data over the period of year 2015 was the following – there were about 35'000 hospital episodes and 70'000 outpatient episodes in RCCUH (in total the less than 2GB RAM). The performance on such queries and data volume is seen in Fig. 2, where queries are sorted in an increasing order by their execution time.

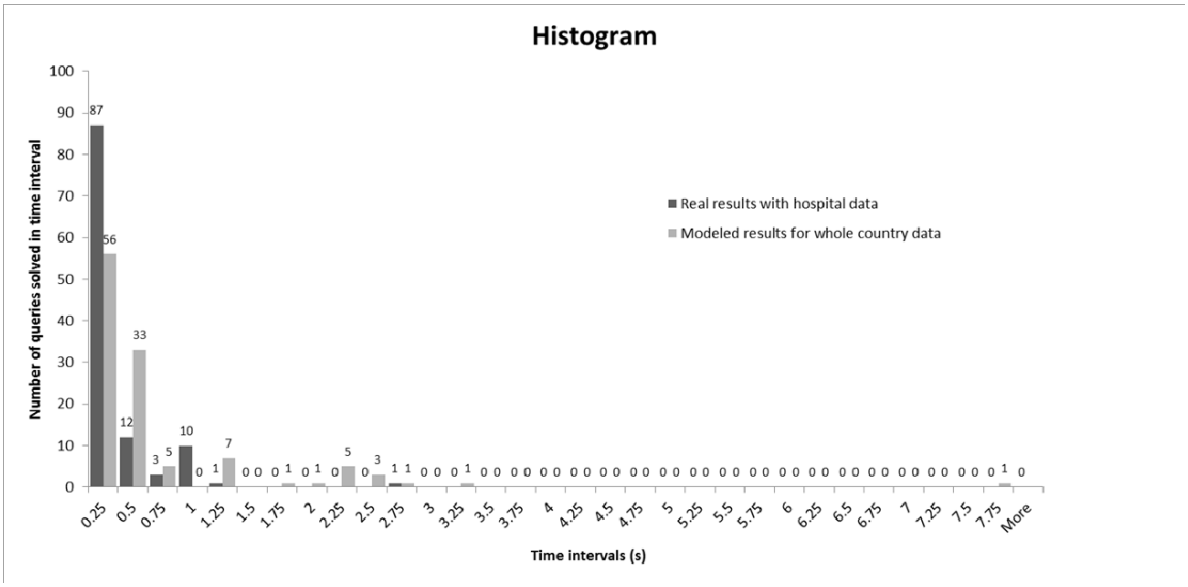


Fig. 2. Performance of the query execution.

We can see that the vast majority of these 120 queries executes in less than 0.3 seconds. According to statistics there are about 350'000 hospital episodes altogether in all hospitals in Latvia per year (about ten times more than in RCCUH). It means that all these data would take up less than 20 GB RAM. Since the star data ontology is granular[27,28], the query execution can be done in parallel on all four cores of a quad-core computer thus improving the execution time four times (our experiment seen in Fig. 2 was performed on only one core). We can conclude that the performance of the query execution over data of all the hospitals in Latvia would only be 2.5-3 times slower than it is now providing the ability to answer a vast majority of queries in less than one second.

We are, of course, not limited by only one computer. We can also use several computers connected via high throughput Ethernet thus reducing the waiting time even more (e.g. one second on ten year data of all Latvian hospitals). Acceptable performance on very large data volumes is another research topic requiring more studies.

By working on the proof of concepts we can conclude that practical testing of our approach has demonstrated that the proposed language can be successfully used at least for the scope of the health system in Latvia.

## 6. Conclusions and future work

Health professionals consider that clinical governance shall be effective only if financial control, service performance and clinical quality is integrated so that clinicians are engaged and service improvements are generated[29]. An integral part of this engagement is also the ability to understand how the data representing clinical process are collected and integrated, and what the numerous data elements mean. Learning of the proposed query tool naturally fits in a larger set of activities aimed to foster self-regulation ability of health professionals necessary for joint optimization of both technical and social aspects towards the goal – a more effective hospital.

In this paper we have proposed a query language that is based on the natural language principles. Thus it is easier to use than traditional query languages for querying data from databases (like SQL). Practical experiments with our query language have shown that there are yet at least two important features that must be added to increase its usability – the subset definition feature (DEFINE InfectiousDisease = SELECT …) and the attribute definition feature (DEFINE HospitalEpisode.duration = dischargeTime-admissionTime). These and similar features are currently under development and require some technical work to be implemented. Another useful feature would be to obtain the event distribution in time which could further be analyzed in MS Excel using its time axis component.

The formal natural language sketched in this paper is still quite a bit controlled. Our future goals include reducing the level of control, so that the language would become even more usable for domain experts being non-programmers. Queries would be formulated very inaccurately (perhaps providing only some basic keywords), and the system could then try to understand the query the user has wanted to formulate and offer the resulting query (or more than one potential queries) back to the user for affirmation. This would be the next step towards a really user-friendly query language.

## Acknowledgements

This work is supported by the Latvian National research program SOPHIS under agreement Nr.10-4/VPP-4/11. Authors are also very thankful to Lolita Zeltkalne for language consulting.

## References

1. Fox WM. Sociotechnical System Principles and Guidelines: Past and Present. J Appl Behav Sci 1995;31(1):91–105.
2. Davies HTO, Harrison S. Trends in doctor-manager relationships. BMJ 2003;326(7390):646–9.
3. Degeling P, Maxwell S, Kennedy J, Coyle B. Medicine, management, and modernisation: a “danse macabre”? BMJ 2003;326(7390):649-52. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1125544&tool=pmcentrez&rendertype=abstract
4. Edwards N, Kornacki MJ, Silversin J. Unhappy doctors: what are the causes and what can be done? BMJ 2002;324(7341):835–8.
5. Ham C, Alberti KGMM. The medical profession, the public, and the government. BMJ [Internet] 2002;324(7341):838–42. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1122770&tool=pmcentrez&rendertype=abstract
6. Hellström A, Lifvergren S, Quist J. Process management in healthcare: investigating why it’s easier said than done. J Manuf Technol Manag [Internet] 2010 [cited 2014 Nov 2];21(4):499–511. Available from: http://www.emeraldinsight.com/10.1108/17410381011046607
7. Klopper-Kes a. HJ, Meerdink N, Harten WH Van, Wilderom CPM. Stereotypical images between physicians and managers in hospitals. J Heal Organ Manag [Internet] 2009 [cited 2011 Dec 5];23(2):216–24. Available from: http://www.emeraldinsight.com/10.1108/14777260910960948
8. von Knorring M, de Rijk A, Alexanderson K. Managers’ perceptions of the manager role in relation to physicians: a qualitative interview study of the top managers in Swedish healthcare. BMC Health Serv Res 2010;10:271.
9. Braithwaite J, Runciman WB, Merry a F. Towards safer, better healthcare: harnessing the natural properties of complex sociotechnical systems. Qual Saf Health Care [Internet] 2009;18(1):37–41. Available from: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2629006&tool=pmcentrez&rendertype=abstract
10. Barzdins J. Developing health care management skills in times of crisis: A review from Baltic region. Int. J. Healthc. Manag.2012;5:129–40.
11. Goldzweig CL, Towfigh A, Maglione M, Shekelle PG. Costs and benefits of health information technology: new trends from the literature. Health Aff (Millwood) 2009;28(2):w282–93.
12. Broekstra G. Building High-Performance, High-Trust Organizations [Internet]. Palgrave Macmillan; 2014. Available from: http://www.palgraveconnect.com/doifinder/10.1057/9781137414724
13. Di Maio P. Towards a Metamodel to Support the Joint Optimization of Socio Technical Systems. Systems [Internet] 2014;2(3):273–96. Available from: http://www.mdpi.com/2079-8954/2/3/273/
14. Mumford E. The story of socio-technical design: Reflections on its successes, failures and potential. Inf Syst J 2006;16(4):317–42.
15. Spehar I, Kjekshus LE. Medical Management in Norwegian Hospitals. Prof Prof [Internet] 2012;2(1):42–59. Available from: http://journals.hioa.no/index.php/pp/article/view/178
16. Semler R. Managing Without Managers. Harv Bus Rev 1989;67(5):76–84.
17. Lubans J. The Invisible Leader. Lessons for Leaders Orpheus Chamber Orchestra. Organ Dev Pract 2006;38(2):5–9.
18. Karakas F. New Paradigms in Organization Development: Positivity, Spirituality, and Complexity. Organ Dev J 2009;27(1):11–25.
19. Zviedris M., Barzdins G. ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In The Semantic Web: Research and Applications, LNCS, Vol. 6644, 2011, 441 – 445.
20. Aspin, A.: Self-Service Business Intelligence. High Impact Data Visualization with Power View, Power Map and Power BI, pp. 1-18 (2014)
21. F. Li and H. V. Jagadish. Constructing an interactive natural language interface for relational databases. PVLDB, 8(1):73–84, 2014.
22. M. Llopis, A. Ferrández. How to make a natural language interface to query databases accessible to everyone: An example. Computer Standards & Interfaces, 35 (5) (2013), pp. 470–481

23. N. Papadakis, P. Kefalas, M. Stilianakakis. A tool for access to relational databases in natural language.Expert Systems with Applications, 38 (2011), pp. 7894–7900
24. I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases – an introduction. Natural Language Engineering, 1(1):29–81, 1995.
25. A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In COLING, 2004.
26. Barzdins J., Rencis E., Sostaks A.: Data Ontologies and Ad Hoc Queries: a Case Study. In: H.M. Haav, A. Kalja, T. Robal (Eds.) Proc. of the 11th International Baltic Conference, Baltic DB&IS, TUT Press, pp. 55-66 (2014)
27. Barzdins, J, Rencis, E., Sostaks, A.: Granular Ontologies and Graphical In-Place Querying. In: Short Paper Proceedings of the PoEM 2013, CEUR-WS, vol 1023, pp. 136-145 (2013)
28. Barzdins, J., Rencis, E., Sostaks, A.: Fast Ad Hoc Queries Based on Data Ontologies. In: H.M. Haav, A. Kalja, T. Robal (Eds.), Frontiers of AI and Applications, Vol. 270, Databases and Information Systems VIII, pp. 43-56, IOS Press (2014)
29. Smith LFP, Shepperd J. Making Clinical Governance Work for You. BMJ Br Med J 2001;322(7302):1608.