Crossing heterogeneous information sources for better analysis in Long Term Care for elderly

Szirbik N. B. a,1, Pelletier C. a,1 and Chaussalet T. b

a Information System Cluster, Faculty of Management and Organisation, Rijksuniversiteit Groningen, NL
b Health and Social Care Modelling Group, Centre for Health Informatics, Cavendish school of Computer Sciences, University of Westminster, London, UK

Abstract. In this paper we describe a methodology that emerged during an implementation of a healthcare-oriented data repository for LTC, which consists in grouping information from heterogeneous and distributed information sources. We developed this methodology by first constructing a concrete data repository, containing information about elderly patients flows in the UK’s long-term care system (LTC). In our specific case, the role of the data repository is to allow knowledge extraction about consumption and behavioural tendencies in the elderly people population within the LTC system. These tendencies can be depicted in terms of survival behaviour (modelling), cost evolution, and bed use. Other types of knowledge that can be extracted are typical patient profiles, placement policy in term of rules and criteria effectively applied and, specific features of the business process behind the long-term care provision. A well-constructed data repository can support the discovery (analysis) of hidden aspects about the way the patients are placed and accepted in the LTC, and also how the allocated resources are consumed. We argue that the use of this methodology could save time in similar undertakings or in other fields than LTC or even healthcare. We present some examples where the methodology could be applied with good results.

Keywords. Camera ready manuscript, IOS Press, LATEX, book, style

1. Introduction

In this paper we describe a methodology that emerged as a byproduct of an implementation of a healthcare-oriented data repository for LTC (long-term care system). This repository is grouping information from heterogeneous and distributed information sources, containing information about elderly patients flows in the UK’s LTC. We argue that the use of this methodology could save time in similar undertakings or in other fields than LTC or even healthcare. Given now the benefit of hindsight, we empirically estimate that the possession of knowledge about the best ways to design and implement the data ware-
house could have saved more than 70% of the time and effort to build the repository. This is why we consider important to document and publish our development method. This could help healthcare practitioners, and practitioners from other areas with identical features concerning the data, to speed their data gathering, organization and avoid pitfalls and bad practices.

In 2001, the government of UK started a three-year project aiming at improving the quality of health and social care services through better information management [1]. For this purpose, in addition to defining a standard set of data to be collected [2], analysis techniques have to be developed and implemented. These techniques should support better understanding of resource use and population needs, particularly the health and social care services for the elderly. In such contexts, the knowledge of tested methods and established frameworks is very useful to improve data gathering about the flows of the patients through the entire system and also studying what are the different tendencies in the provision of patient cares, and especially of the patient trajectories within the care system.

The paper is organized as follows: section 2 deals with the general description of our specific project, section 3 presents the methodological framework we have build and propose for further use, section 4 elaborates on two of the most interesting lines of research that have been open by our work, and finally section 5 discusses the limitations of frameworks and their applicability, and draws the main conclusions.

2. Problem

In the context of the studies of the long-term care for elderly people, the role of the data repository is to allow knowledge extraction about consumption and behavioural tendencies in the elderly people population within the LTC system [3]. These tendencies can be depicted in terms of survival behaviour (modelling) [4], cost evolution [5], and bed use. Other types of knowledge that can be extracted are typical patient profiles, placement policy in term of rules and criteria effectively applied [6] and, specific features of the business process behind the long-term care provision. A well-constructed data repository can support the discovery (analysis) of hidden aspects about the way the patients are placed and accepted in the LTC, and also how the allocated resources are consumed (see [7]).

In many complex social organisations, like healthcare, the decision making process is distributed. There are a number of different stakeholders, with rather different goals, who have access to different, and disconnected sources of data. This leads to local ‘views’, usually narrow and biased. Local decisions can be supported by their views. Nevertheless, the impact of these decisions on a larger scale cannot be estimated. For global decisions (e.g. a consolidated budget) the stakeholders have to meet and negotiate. Due to the fact they hold views that have been grown from partial and sometimes conflicting perspectives, the decision taken can be just biased in the favour of the strongest negotiator.

A data repository that gathers information from the different areas of the social health care stakeholders leads to the possibility to have a holistic perspective of the investigated social healthcare system. Also, this centralisation allows various analysis, data flow and process mining that search for global estimations and global understanding of otherwise
hidden phenomena. A global-view data analysis tool of crossed historical data about LTC patients is the main instrument to achieve better coordination between stakeholders. Such a tool allows:

- To analyse the use of the resources for LTC in the past,
- To identify possibilities for improvement,
- To discover the path pattern (trajectories) of the patients in the system (eventual bottlenecks),
- To establish patient’s profiles based on their experience in the system.

The results of the analysis will enable tactical and strategic managers to have a better understanding of the system and leads to a better utilization of the existing and planned resources. Our approach was to allow a continuous re-grouping (with a rate of update of 3-6 months) of the historical data concerning the LTC patient, across the various organizations that own and use these data.

3. The methodology

The methodology as established by us, consists of six steps or phases. Compared with the software and database developments models known in software engineering, this methodology is close to the well-known waterfall model [8]. This model has been criticized for its lack of flexibility via iterative development, but in our case, the iterative style of development can be anyway applied over the steps of the proposed methodology. This can be done by viewing the steps as milestone-ending phases, containing many short term iterations that can implement small parts of functionality and data schema of the overall project. In this view, one can see the methodology as an extension of the UP (the Unified Process), or any other iterative development methodology as Agile Development or XP (extreme programming), where short iterations that are strongly time-boxed (i.e. these has to start and end at a very precise time) and are part of more "elastic" phases. These phases are defined by methodological constraints (and not project dates), and the end of these can be seen as conceptual milestones for the project manager - opposed to time/budget milestones.

The first step (phase) ends when the development team has identified and solved the most critical requirement for the building of the data warehouse. Also, the developers should be able to convince the stakeholders that they have a sound solution for this critical requirement. For example, in a typical warehouse that collects weather data, the elimination of "noise" (unwanted data) is crucial. For a warehouse that stores data about potential terrorist communication, fast search for certain text patterns is the most important. In genome research, structuring algorithms have to be discovered before the warehouse is built. Every application tends to have its own core set of critical requirement.

The strongest requirement about the data in an LTC warehouse (at borough, county or national level) is that the data should be anonymous according to the UK privacy act. Our methodology considers that in the implementation of such a repository, the first step is to establish clear ways to ensure that the privacy requirement (via anonymity of the records in the repository) is achievable. For other projects of data gathering in other fields, it is possible that the main requirement is different. Our point is that this requirement should be identified from the start and tackled first. The implementers should show
to the data owners and the stakeholders that there is a way to align to this requirement. In our case, we have used mapping files, owned by the owners of the data sources. In the mapping file, the unique identity of the patient (given by his social security number) was associated with a unique identifier generated by our system. To disclose the identity of the patient, access to this mapping file is necessary, and the access is limited only to the updating sessions of the central repositories.

We had to demonstrate to all the stakeholders that contributed with their data to our central (academically kept) repository that all the data we keep and use for testing and validation of our analysis models are anonymous. By developing a "cookie-based", stakeholder owned key-mapping mechanism, that allowed the cyclic update of the central repository without using the names, social security numbers or other non-anonymous information from the local records, we have basically been able to start our project. For project management, the reaching of this milestone (i.e. the solving of the critical aspect(s)), can be viewed as the "go/no-go" decision point in the project.

The second step in the methodology is to identify the quality and scope of each data source and also the rate of updating (depending on the dynamics of the entities to which the data refers). We are calling this step "scope and granularity" analysis. Another aspect identified in this step is about the complementarities of the sources of data, these raising usually all kinds of potential inconsistencies. For example, after crossing two sources, we have found patients who were still in the system as alive after their registered death or appeared with a different gender in different states. Most of these situations are due to data inputting mistakes and should be eliminated or fixed. This aspects are critical to any data integration effort and various software filters and inconsistency detectors should be developed before other software applications that are necessary for the repository.

Sometimes, for data gathering projects where the data volumes are low, but the structure of the data is very complex (contains lots of unstructured or semi-structured text), some filtering and consistency procedures should be designed to be done by hand by human operators. These procedures have to be established and validated early in the project. We are applying here the same development principle like in the UP (unified process - see [8]) that states that the critical aspects of a projects should be tackled in the early stages of the project. One good final advice in this phase is to validate the collection of data by running one (small-scope) collection exercise.

The third step is to match the potentially collectible data with the results that are desired by the stakeholders and the decision makers. This will reduce the scope of the data gathered and will simplify the schemas of the data repository. In our case, the scope reduction was also a consequence of the analysis tools that have been already developed. However, such an intension of the scope should be made with care, and the implementers should consider that it is possible that new tools and models can be added to the analysis methods and certain data that are currently not used can be valuable in the future. Another aspect in this phase is to determine what are the factors that can increase the speed of the analysis (for example, it is no use to develop a tool that makes an analyst to wait minutes or hours to have a result for a trial with some test parameters -s/he should be able to "play" with the system in an almost interactive way). The speed factor is dependent on the availability of data - or search time. In any data repository, there are data that are used very often is buffered in small "cache"s, or the navigability of the database schemas is designed in a way that shortens the path to those records that are queried often. Data that are considered immediately useful for the analysis tools, should be very
easy reachable, and data that are not, could be placed "at distance" from the main query starting points. If necessary, the new queries can be written for new tools, or the old queries can be rewritten. In extreme cases, the whole schema of the database could be changed, and simple porting software can be used to migrate the whole information in the repository towards a database with a different operational structure (but containing the same information).

The fourth step consists in building an ontology that maps the relevant terms in the scoped universe of discourse. This is necessary because in distributed environments, the denominators for data attributes and values can be different (depending on local technical jargon and/or background of the local data collectors). The central data repository schema and architecture should be based on this common ontology, which can be also shared via mapping the terms and concepts to the local database schemas. What we have observed in our work is that different people, from different environments and with different backgrounds use local specialized languages that are incomprehensible for the outsiders. This makes enormously difficult to pot the stakeholders to discuss together about their local decisions. A very important byproduct of the development of the data integration is that the stakeholders that are also decision makers learn to communicate better and subsequently to understand better the problems of the others, enabling in this way the possibility to achieve win-win decisions through collaborative negotiation. We observed that the ontology makers are also perceived as third-trusted parties (not only the developers of the integrated repository). Looking retrospectively, we consider now that the achievement of this milestone could be the most important outcome of such a project.

The next two phases can be grouped under the name maintenance. However, the scope and goals of the two are different. The fifth step is to establish the update policy for each local source and estimate the costs involved. Besides the speed and intensity of local change, granularity is an important aspect, because different databases can store the same kind of information with a different temporal or spatial scope. For example, costs can be recorded as per week, per month or per year; or in a hospital, per clinic, department or bed (or individual case). This leads to the transformation into a unique (adapted for analysis) granularity. This can involve the writing of software that adjust granularity to the desired level, but also can involve manual procedures where the granularity conversion cannot be achieved by simple routines. At this level, a manager can consider that the project is terminated. However, we include as a last step the post-validation and training phase. Validation should happen anyway in each iteration (functionality and data models created in that iteration are thoroughly tested) and this eliminates the need for an explicit testing as in the waterfall model of development. Therefore, we consider the first operational use of the system (first effective collection of data) as a last maintenance and bug-fixing stage.

Hence, the sixth step is the first real-life gathering of the data, and by this testing if the filtering software deposits correctly the right information into the repository. The sources should be analyzed with the simplest methods and after the data collected should be analyzed immediately with the same methods to detect anomalies that are induced by the data gathering process. In our case, we discovered in this phase that it is necessary to enact some methods to "clean" automatically the gathered data from noise. The development of this "combing" software is tedious and the implementers should plan their project in a way that allows slack in this step. However, if a few records generate the
noise it is better to handcraft ad-hoc some simple queries that eliminate these. In other projects - with different nature of information gathered, we believe that other problems can appear in this phase. Our advice for this stage is to include some slack iterations (two, but this depends on the scale of the project), to iron out the discovered problems without the pressure to launch the repository for decision making analysis. This does not imply that the users should not work already with the system, offering them a good opportunity to obtain final training for system use.

Throughout the whole project, the stakeholders should be heavily involved. Each iteration (we recommend 3-4 weeks for an iteration, with 2-4 iterations per methodology step) should end with a presentation of the existing functionality to the stakeholders. These should provide feedback and approve the course of action. Changes of the functional requirements should lead to the changes in the operational plans (number of iterations and their content), plus the time/budget re-arrangements that raise from this desired changes.

4. Future work

We consider that two aspects are very important to be still investigated. Our proposed methodology for development is actually a "methodological framework" that shows "what" has to be done and in which order these issues have to be solved. But it has no clear mapping onto a project management structure. We have made a link with iterations, but we have not offered yet clear guidelines of how to plan the project, how to link the plan to a team and how to link these to a budget and control mechanism. The other interesting avenue for future research is related to ontological engineering.

The methodology suggest the use of the waterfall model. This model in software development is still used today because of the desire for predictability. Project managers and strategy planners cannot work by not having a clear idea how much it will cost to integrate data and how long it will take to make it usable. Predictive approaches in project management look to do work early in the project in order to yield a greater understanding of what has to be done later. This way, stakeholders can reach a point where the latter part of the project can be estimated with a reasonable degree of accuracy. With predictive planning, a project has two stages. The first stage (usually up to the "go-no/go" decision) comes up with plans as a final product and is difficult to predict, but the second stage is much more predictable because the plans are in place.

However, there is a debate about whether software projects can be really predictable [9]. The core of this problem is requirements elicitation. The main sources of complexity in software or database centric projects is the difficulty in understanding the requirements of the stakeholders. These tend to change even later in the development process. The requirement changes are usually making a predictive plan impossible. Freezing the requirements early on and not permitting changes is possible, but this leads typically in the outcome of delivering a system that does not fulfill the real requirements of its users.

Requirements change is clearly unavoidable, and it is impossible to stabilize requirements sufficiently in order to use a predictive plan. This leads to the need advocates adaptive planning. The difference between a predictive project and an adaptive project is that planning is done either in the beginning either continuously. Changing requirements will just change the iteration structure and content. Of course, this will lead to changes
in the initial contract. With a predictive plan, one can develop a fixed-price/fixed-scope contract. Such a contract says exactly what should be built, how much it will cost, and when it will be delivered. Such fixing isn’t possible with an adaptive plan. However, it is possible to fix from the start a budget and a time for delivery, but this will not fix exactly what functionality will be delivered. An adaptive contract assumes that the users will collaborate with the development team to regularly reassess what functionality needs to be built and in extreme situations will cancel the project if progress ends up being too slow.

For stakeholders, the adaptive approach is less desirable, as they would prefer greater predictability in the project. However, predictability depends on a precise, accurate, and stable set of requirements. If the development team can demonstrate to the stakeholders that they cannot stabilize their requirements early, a flexible, iterative project planning and management should be forced upon them. However, we have not yet developed a clear framework for project planning and control and how this is mapped on the methodological steps proposed in the previous section. This is subject for further research and will need input from other cases of software development involving database integration. Also, we should extend the framework by including team organization tips, and also a mapping to a model that includes the time/budget constraints of the project.

Another line of research we consider valuable to continue is the development of the activities in step 4 (ontology construction). For environments where the interaction of the stakeholders is not really important, this step can be even ignored. However, in our case, as mentioned, the building of the ontology triggered a conscious effort of the stakeholders to better the understanding of each other. We would like to investigate if this has happen in other projects and also to extend the ontology building process from a mere step in the framework to a full process that takes place in parallel with the development process (and supporting it), from the beginning. Also, it is very interesting to study the novel technologies offered by the ontology building research in the field of the Semantic Web (especially those investigations in the Genome Ontology Project - involving huge data gathering, see [10]). These can bring new insights in how the parallel process of ontological construction in the project can be supported, automated and integrated with the development process and stakeholder feedbacks.

5. Discussion and conclusions

One of the biggest problems with any framework (for software development or other domains) is that it covers partial aspects and there are always too many gaps, at conceptual and also at implementation levels. Though, by trying to fill these gaps, the understandability of the framework is always reduced and the potential user is lost in too many details. To learn how to use our framework, a user need to try it out. We would appreciate any feedback from such potential users (our contact details are indicated on the first page of this paper). The content of the paper reflects the incomplete state of our knowledge and also on the gaps that need to be filled and extensions to be developed. Anyway, all of the above proposed ideas are in fact based on the idea that it is a bad thing to start a data integration project without a methodological framework. We have discovered a pattern for a specific approach and we consider that the pattern is generic enough to be applicable in other contexts. What was not a goal of this paper is to demonstrate the applicability of
the methodological framework in other domains with similarities with our project. Never-
theless, this lack of proof is not a negation of the applicability either, on the opposite,
our intuition is that the methodology could be very useful, especially for environments
with lots of heterogeneity (in terms of domains, people’s skills and backgrounds, goals,
and especially information systems).

In conclusion, we claim that a lot of effort can be saved in a data integration undertaking if
the development team identifies the correct methodological pattern. Our methodo-
logical framework can at least play the role of a guideline or of an inspiration point. We
have presented the outlines of our project and the derived framework steps:

1. identification and solving of the most critical issue(s)
2. identification of the scope and granularity of the data sources, including the data
   overlapping identification
3. identification of the most needed data in the repository
4. ontological alignment of the data and also of the people
5. identification of the update policies
6. validation and fine-tuning via the first real data-gathering

We also claim that this (smallest) framework can be used for practical implementa-
tions. It has been argued in the paper that the development team (by realizing step 4)
is playing the role of third trusted party that teaches the stakeholders a common shared
language, enhancing mutual understanding. The main conclusion here is that in distrib-
uted decision making the process core boils down to mere negotiations. Insight into the
problems faced by other (by understanding its language, data, and the analysis results of
this data), can lead the negotiators to win-win situations. This should be the social result
of any project that collects data for better decision making leading to enhanced global
outcomes.

Acknowledgements

We thank Teresa Temple and Peter Crowther from the London Borough of Merton So-
cial Services Department for providing data and feedback during model development,
and Peter Millard, visiting professor at the University of Westminster, for expert advice.
This work was supported by the Engineering and Physical Sciences Research Council
(GR/R86430/01).

References


