Volume 5, Issue 4, 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Steps to implement Universal Data Warehouse

Neelam Ahlawat
Computer Science Engineering Department
SGT Institute of Engineering & Technology,
Gurgaon, Haryana, India

Dr. Ritu Sindhu
Associate Professor, CSE Department
SGT Institute of Engineering & Technology,
Gurgaon, Haryana, India

Abstract— In this paper we describe a methodology that emerged during a healthcare project, which consisted among others in grouping information from heterogeneous and distributed information sources. We developed this methodology build a UDW data repository, containing information about different health care companies term care system (LTC). We argue that the use of this methodology could save time in similar undertakings or in other fields than LTC or even healthcare.

Keywords— >DOTNET; MYSQL; Date Warehouse; OLAP , LTC

I. INTRODUCTION

In this paper we describe both a medical software engineering (MSE) methodology that emerged as a by-product of an incremental implementation of a healthcare-oriented data repository for LTC (long-term care system) and also the important aspects of the usage of this repository. Built as a central database, it is grouping information from heterogeneous and distributed information sources, containing information. We argue that the use of this methodology could save time in similar undertakings or in other fields than healthcare. The methodology is an extension of known iterative frameworks as RUP (Rational Unified Process). Gathering the data for our project has been a difficult undertaking, leading to "re-starts" a couple of times. Basically, our data gathering and data organization was repeated three times, each time trying to avoid the mistakes previously. Given now the benefit of hindsight, we empirically estimate that the knowledge about the best ways to design and implement a medical data warehouse 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 medical areas with identical features concerning the data collection. Applying the proposed "six-steps" will speed data gathering and data organization, and also help to avoid some already known pitfalls and bad practices.
The paper is organized as follows: -

- Deals with the general description of our specific project,
- Presents the methodological framework we have build and propose for further use,
- Elaborates on two of the most interesting lines of research that have been open by our work, and finally
- Discusses the limitations of frameworks and their applicability, and draws the main conclusions.

Six methodological steps

II. THE DESCRIPTION OF THE LTC RELATED PROJECT

The aim is improving the quality of health and social care services through better information management .For this purpose, in addition to define a standard set of data to be collected , analysis techniques had 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. Separately from this nationwide project, smaller academic projects , went towards establishing models for data for analysis, and also provide tools for practitioners in the business processes related to the LTC. We can classify our repository as a database for decision making in healthcare.

2.1. Data gathering from heterogeneous sources

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 . These tendencies can be depicted in terms of survival behavior (modeling) , 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 [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 .

In many complex social organizations, 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 are usually a result of their associated local 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.

2.2. The use of the centralized data-repository

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 centralization 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 (the general architecture of the 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.

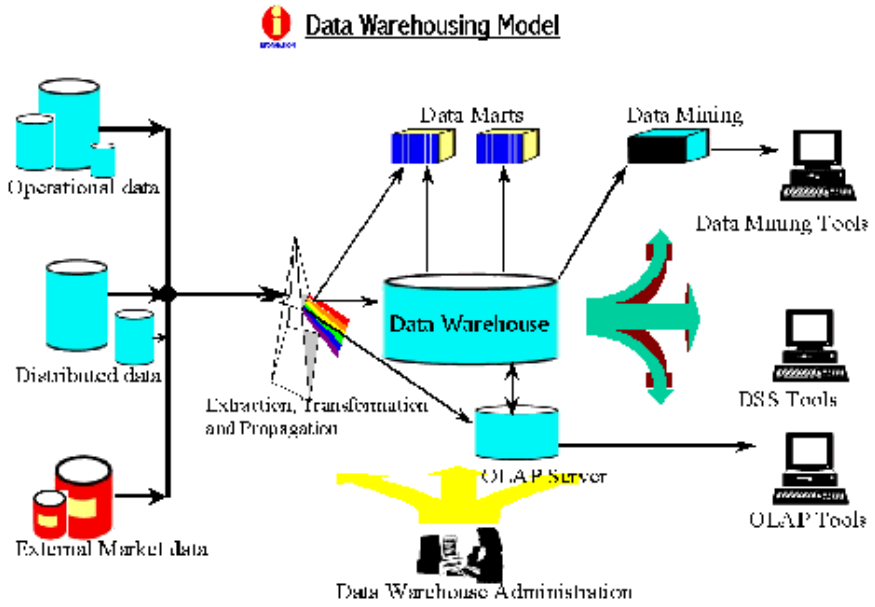

Figure 1

The results of the analysis 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 renewal (with a rate of update of 3-6 months) of the historical data concerning the LTC patients; all these across the various organizations that own and use these data.

- Central repository
- Local Data
- Local Data
- Filter – up-date
- Survival & Cost Analysis
- Interfaces
- Processed Data

2.3. The importance of a SE methodology

Information technology projects in general, not only the ones in the medical domain (for a thorough discussion about MSE issues) have a high degree of complexity, risk and uncertainty. The rate of failure and delays (i.e. running over the time and budget) is still considered unacceptable in the software industry. The main factor for success is considered today the adoption of best-practices, either by using a framework like RUP, or at a smaller scale, domain specific reusable methods (so-called “project skeletons”). These can be seen as an empirical foundation for new business-demand driven knowledge, opposed to the technology driven approaches, which wrongly adopt a stance that new (technical) innovations like for example object-orientation, web-services, or component technologies offer an immediate “silver bullet”, and a guarantee for success. Without adopting novel principles like iterative development, user participation (both strongly emphasized in RUP), and domain specific knowledge reuse, IT projects in the medical domain will not deliver the expected benefits.

III. THE METHODOLOGY

The methodology as established by us consists of six steps that can be eventually aligned with some of the established milestones in the RUP framework. One can argue that a six-step methodology is just another version of the previously used waterfall model . This has been criticized for the lack of flexibility that can be achieved otherwise via iterative development. Nevertheless, the iterative style of development may and should be applied over the steps of our proposed methodology. This can be done by viewing the iterations as small sections of the steps, with each iteration implementing a small part of the required functionality. In this view one can see the methodology as an extension of any iterative methodology, as RUP, but also 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 – of course, in strong relation with time/budget milestones. Inside each iteration there is a small waterfall project (with sequenced analysis, design, implementation and test tasks), and there is a common practice across any iterative framework to have a meeting between the software team and the project stakeholders, to analyse the partial results, and eventually change requirements for the next iterations.

3.1. Step one – Prove that the hardest requirement can be solved

The first step (that can be aligned with the "Inception" phase in RUP) ends when the development team has identified and (empirically at least) 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 large amount of data about potential terrorist communication, fast search for certain communication 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 requirements. In our particular project, the strongest requirement about the data in an LTC warehouse (at borough, county or national level) is that the data should be anonymous . 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. It is possible that for other projects of data gathering the main requirement is different. Our point is that this critical 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 fulfill 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 convince all the stakeholders who had their data added to our central (academically kept) repository, that all the data we keep and use for testing and validation of our analysis models are anonymous. He have developed a schema involving 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. Only after explaining this schema to the data owners we have been able to start our project. For the project management, the reaching of this milestone (i.e. the solving of the critical aspect(s)), can be also viewed as the "go/no-go" decision point in the project (identical with the ending of the "inception" phase in the RUP framework). A proposed alignment with RUP RUP's "Inception phase" and our "Step one" may be aligned In data-gathering undertaking, the facilities offered by the Web are more and more popular. Databases offer today web-enabled interfaces. However, this can be very costly in order to be achieved in a secure way that ensures data privacy. Healthcare networked data repositories are using typically a public key infrastructure (PKI), but experience shows that the implementation of this infrastructure could be costlier and politically sensitive than the gathering of anonymous data (for a study of the trade-offs with a PKI-based system). Another approach is to insure anonymity by ambiguity algorithms, where the records are altered in a way that makes discovery of patient identity very difficult, but keeps the records usable for various kinds of analysis. Thus, we advise that any project should identify the criticalrequirements first. Typical requirements are high levels of data privacy, data accuracy, analysis speed, ability of filtering large amounts of data, good connectedness, short-term data availability, etc.

3.2. The second step – Quality and Scope of the Sources

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. Basically, one has to understand and to model all the source databases. Sometimes these models are available from the design documents that have been used to implement them, but sometimes the models have to be reverse-engineered. Time granularity is important for each database. In our case, some databases kept financial information per year, some kept per month, and some kept per week. One has to keep in mind that in the central repository these data have to be finally aligned. It is possible that large amounts of data from the sources will never be used. Nevertheless, the builders of the central repository should be aware of the entire potential. Many times, stakeholders are asking late in the project for new data, which is existent in the sources, but the designers are not aware or prepared to tackle these new requirements. If they know the complete models of the sources, it is much easier to change the central data repository model and adapt the gathering procedures, to include the newly asked data requirements. Another issue relevant 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 they appeared with a different gender in different states. Most of these cases are due to data-input mistakes and normally they should be fixed. These 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. One final advice in this step is to validate the collection of data by running one (small-scope) collection exercise. Of course, in this step, it is not yet clear what data will be finally used, but the designers could select those records and fields that are considered relevant by the data owners. With respect to RUP, this step can start during the inception phase (in the last iterations there), or at least in the very first iteration of the elaboration phase. It will end after the first iterations of elaboration phase .

3.3. Step three – identify what data is needed by the stakeholders

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 current analysis methods and certain data that are currently not used can be valuable in the future. For us, another aspect in this step was to determine what factors 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 –she 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). With respect to RUP and project management, this step should be finished before the end of the elaboration phase. In an RUP-based project end of elaboration means that the requirements are "frozen". Any change in the software requirements after this milestone will incur contractually established delays and/or budget modifications. It is evident that "what data the stakeholders want" is naturally aligned with established requirements. The alignment of the RUP project framework and the third step .

3.4. Step four – Build an Ontology

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). Previous research emphasized the importance of semantic alignment in data sharing and data exchange in medical IT systems . Currently, there are various proposals for ontologies for the Semantic Web, specialized for various fields of biomedicine and health care) 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 put 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 ontology could be the most important outcome of such a project. This step is more difficult to link with the RUP framework. Common shared ontology building starts early in any heterogeneous project. However, when we talk about the formal construction of an ontology, we consider that it is necessary to be done somewhere at the end of the elaboration phase and the first iterations of the construction phase. We recommend an alignment .

3.5. Step five – How to update the central repository

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 transforms the collected information into a format with unique (tuned 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 of implemented software should happen anyway after all iterations and this eliminates the need for an explicit testing phase (or step) 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. The fifth step should be finished before the construction phase of RUP ends. This milestone is for software completion, and that means that the update mechanisms should be fully functional at this moment. Hence, step five is part of the construction .

3.6. Step six – Enact exception handling protocols

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. Step six is clearly a part of the transition phase. It can eventually start earlier, as a small pilot during the construction phase, but this is recommended only if the environment of the application is prone to a high number of exception handling. Linking the last two steps with the RUP framework

IV. THE SIX-STEP METHODOLOGY AND PROJECT MANAGEMENT

We consider that two project management aspects are very important. Our proposed methodology shows only "what" should be done and in which order these issues should be solved.But it has no clear mapping onto a project management structure. We have made a link with iterations and phases in RUP, but we have not offered yet clear guidelines of how to plan the project, how to assign the plan to a team and how to link these to a budget and control mechanism. First of all, the methodology is best suited the use of a predictive model for project management. In software development the predictive model is still used today because of the desire for predictability. Project managers and strategy planners want to know how much it will cost to integrate data and how long it will take to make it usable. Also, stakeholders want to reach early the 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, i.e. the inception phase in RUP) ends with strict project planning as a final product. The main shortcoming is that this first stage is difficult to predict, and usually takes much more time than necessary. However, only few people are involved in this stage, and the costs can be still low. The second stage is much more predictable because the plans are already in place. There is a debate about whether software projects can be really predictable . The core of this problem is requirements elicitation. The main source 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 requirements' change 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. We learned from our experience, but also from the experience of other scientists working in data gathering projects that requirements' change is clearly unavoidable, hence it is impossible to stabilize requirements sufficiently in order to use a predictive plan. This leads to the use of 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 the content of iterations. 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 contract is not 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 kind of 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 schema 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.

V. DISCUSSION AND CONCLUSIONS

One of the main problems with any framework (for software development or other domains) is that it typically covers only partial aspects and there are too many gaps left, especially at conceptual and also at implementation levels. Though, by trying to fill these gaps, the understandability and usability of the framework is always reduced because the potential user is lost in too many details. To learn how to use our framework, a user needs to try it out and fill her/himself the gaps.

All ideas presented in this paper are based on the assumption that it is undesirable to start a data integration project without a methodological framework, which has more specificity than the generic RUP framework. We believe that we have discovered a pattern for a specific problem and a specific approach and we consider also that the pattern is generic enough to be applicable in other contexts. Our intuition is that the methodology could be very useful, especially for environments with lots of heterogeneity, in terms of conceptual domains, people's skills and backgrounds, goals; and also containing heterogeneous 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 methodological 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:

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

We also claim that this (smallest) framework is sufficient to be used for simple practical implementations. 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 most important finding here is that in distributed 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.

REFERENCES

[1] [1] DOH, Making it happen - The key areas for action, www.publications.doh.gov.uk/ipus/socialcare/happen/, accessed the 27 th November 2004.
[2] NHS, Single Assessment Process (SAP) Dataset, www.nhsia.nhs.uk/datasets/pages/ds_older.asp, accessed the 27 th November 2004.
[3] C. Pelletier, T. Chaussalet, N. Szirbik and C. Vasilakis, Intergration of Data on Long-
[4] Term Care from Heterogeneous Sources for Research Purposes, MEDICON'04, Ischia, Italy, 2004.
[5] H. Xie, T.Chaussalet and P. Millard (2005) Continuous-time Markov models for the flow of elderly people in institutional long-term care, Journal of the Royal Statistical Society, serie .
[6] C. Pelletier, T. Chaussalet and H. Xie (2005) A framework for predicting gross institutional long-term care cost arising from known commitments at local authority level, Operational Research Society 56, 144--152
[7] H. Xie, T. Chaussalet, W. Thompson and P. Millard (2002), Modelling Decisions of a Multidisciplinary Panel for Admission toLong-term Care, Health Care Management Science.

