INTERNATIONAL JOURNAL OF MEDICAL INFORMATICS 75 (2006) 683–691

journal homepage: www.intl.elsevierhealth.com/journals/ijmi

Six methodological steps to build medical data warehouses for research

N.B. Szirbik[a,*], C. Pelletier[a], T. Chaussalet[b]

[a] Information System Cluster, Faculty of Management and Organization, Rijksuniversiteit Groningen, Landleven 5, Postbus 800, 9700 AV, Groningen, The Netherlands
[b] Health and Social Care Modeling Group, Centre for Health Informatics, Cavendish School of Computer Sciences, University of Westminster, London, UK

ARTICLE INFO

Article history:
Received 18 September 2005
Received in revised form 10 April 2006
Accepted 19 April 2006

Keywords:
Medical software engineering
Medical data repositories
Privacy
Ontology building
Requirement elicitation
Long-term care for elderly

ABSTRACT

Purpose: We propose a simple methodology for heterogeneous data collection and central repository-style database design in healthcare. Our method can be used with or without other software development frameworks, and we argue that its application can save a relevant amount of implementation effort. Also, we believe that the method can be used in other fields of research, especially those that have a strong interdisciplinary nature.
Background: The idea emerged during a healthcare research project, which consisted among others in grouping information from heterogeneous and distributed information sources. We developed this methodology by the lessons learned when we had to build a data repository, containing information about elderly patients flows in the UK's long-term care system (LTC).
Design: We explain thoroughly those aspects that influenced the methodology building. The methodology is defined by six steps, which can be aligned with various iterative development frameworks. We describe here the alignment of our methodology with the RUP (rational unified process) framework. The methodology emphasizes current trends, as early identification of critical requirements, data modelling, close and timely interaction with users and stakeholders, ontology building, quality management, and exception handling.
Results: Of a special interest is the ontological engineering aspect, which had the effects with the highest impact after the project. That is, it helped stakeholders to perform better collaborative negotiations that brought better solutions for the overall system investigated. An insight into the problems faced by others helps to lead the negotiators to win-win situations. We consider that this should be the social result of any project that collects data for better decision making that leads finally to enhanced global outcomes.

1. Introduction

In this paper we describe both a medical software engineering (MSE) methodology that emerged as a byproduct of an incremental implementation of a healthcare-oriented data repository for LTC (long-term care system) in the United Kingdom and also some aspects of the usage of this repository. Built as a central database, it is grouping information from heterogeneous and distributed information sources, containing information about elderly patient's 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 healthcare. The

* *Corresponding author*. Tel.: +31 50 363 7316; fax: +31 50 363 2275.
E-mail addresses: n.b.szirbik@rug.nl (N.B. Szirbik), c.m.p.pelletier@rug.nl (C. Pelletier).
1386-5056/$ – see front matter © 2006 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.ijmedinf.2006.04.003

methodology is an extension of known iterative frameworks as RUP (rational unified process, see Ref. [8]). 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 previous mistakes. Given 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 similar data collection features. Applying the proposed "six-steps" could accelerate data gathering and data organization, and help avoid some already known pitfalls and bad practices.

Our MSE methodology is rather domain specific, but it can be seen as a specialization of more general software engineering (SE) methodologies. A medical software project that is to be managed following a consecrated framework (like RUP, for example, see Ref. [8]) can be aligned easily with our methodology, by using the same concepts of phase, milestone, and iteration. The data gathering-oriented methodology presented here, follows six "steps", which can be naturally matched with the RUP time-phased project management schema, as we are showing later in the paper. If a heavyweight framework such as RUP is not used because of the smaller extent of the project, our proposed six steps can be applied independently and the project can be managed following our guidelines. Thus, the six-step methodology can be applied with or without using the RUP. If RUP is used, the methodology is just one of its extensions, and if it is not, the methodology becomes a very streamlined version of RUP.

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 built and proposed for further use, Section 4 elaborates on two of the most interesting directions of research that have been open by our work, and finally Section 5 discusses the limitations of these frameworks and their applicability, and draws the main conclusions.

2. The description of the LTC related project

In 2001, the UK government started a 3-year project aiming at improving the quality of health and social care services through better information management [1]. For this purpose, in addition to define a standard set of data to be collected [2], analysis techniques had to be developed and implemented. These techniques should support better understanding of resource use and population needs particularly for health and social care services for the elderly. In such context, the knowledge of tested methods and established frameworks is very useful to improve the collection of data about the flows of patients through the entire system. It also supports the study of different tendencies in the patient care provision, and the analysis of the patient trajectories within the care system. Apart from this nationwide project, smaller academic projects like ours, went towards establishing models for data analysis, and also provide tools for practitioners in the business processes related to the LTC. Relative to the healthcare database taxonomy proposed by van Bommel et al. (see Ref. [22]), 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 long-term care for elderly people, the role of the data repository is to allow knowledge extraction about consumption and behavioral tendencies in the elderly people population within the LTC system [3]. These tendencies can be depicted in terms of survival behavior (modeling) [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 (see Refs. [6,23]), and specific features of the business process behind the long-term care provision. A well-constructed data repository can support the discovery and analysis of hidden aspects about the way the patients are placed and move in the LTC, and how the allocated resources are consumed (see Ref. [7]).

In Fig. 1, we are describing the possible states of a patient in LTC (UK). The entry state is an "arrival in the system" moment, a milestone in one's life – for example retirement – after that (s)he needs care for elderly people. In each state figured, patient records (medical, financial, socially related) are registered at various locations of the LTC system. For example, it is possible that for an old patient staying at home, who needs a social worker for care, this care is provided by the local church volunteers. The record of this activity will be probably kept on a computer owned by the church community. We want to emphasize here that the overall system records are heterogeneous due to their different source, interpretation, and purpose; secure and private, hence difficult to be accessed, and most importantly, are owned by the local organizations. Sometimes, the lack of record itself can represent also something. For example, in Fig. 1, these situations could be represented by the state "no care" in the "at house" location.

2.2. *The use of the centralized data-repository*

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, and resulting in local decisions. 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 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 all areas that represent health and social care stakeholders leads to the possibility of having a holistic view of the system. This centralization allows various types of analysis, data flow, and process mining that search for global estimation and global understanding of otherwise hidden phenomena. A global-view data analysis tool of crossed historical data about LTC patients is

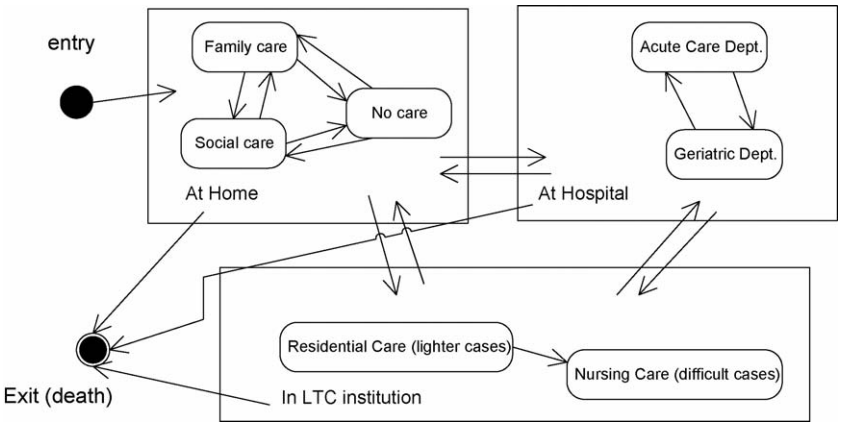

Fig. 1 – The state diagram illustrating the patient flow.

the main instrument to achieve better coordination between stakeholders. Such a tool, whose general architecture is shown in Fig. 2, allows to:

- analyze the use of the resources for LTC in the past;
- identify possibilities for improvement;
- discover the path patterns (trajectories) of patients in the system (eventual bottlenecks);
- establish the profiles of patients based on their experience in the system.

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 is to allow a continuous renewal (with a rate of update of 3–6 months) of the historical data concerning LTC patients; all these across the various organizations that own and use these data.

Currently there are more "disciplined" approaches for data gathering, based on high level policies and nationwide project requirements (see Refs. [17–19]), but for very specific data gathering exercises, were it is not possible to attain a level of wide socially driven coordination between stakeholders, it is desirable to have one partner as "data-gatherer" and the rest only as users of the interpreted data. This pattern can be identified in the majority of academic research projects that aim to provide non-academic partners (who have to supply the data) with applications and tools that are using the gathered data.

2.3. *The importance of a SE methodology*

Information technology projects in general, and in healthcare in particular [13], have a high degree of complexity, risk and uncertainty. The rate of failure and delays (i.e. running over time and budget) is still considered unacceptable in the software industry. In the software engineering domain the main factor for success is considered today to be 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 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 participa-

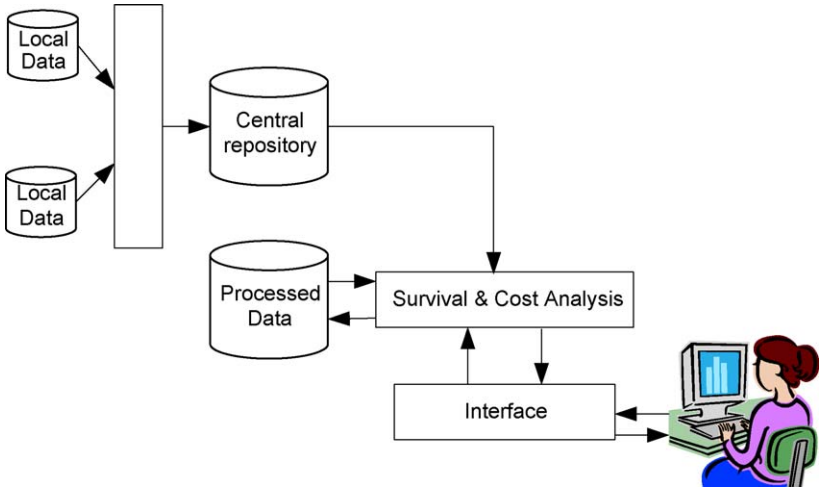

Fig. 2 – The architecture of the data-gathering and analysis application.

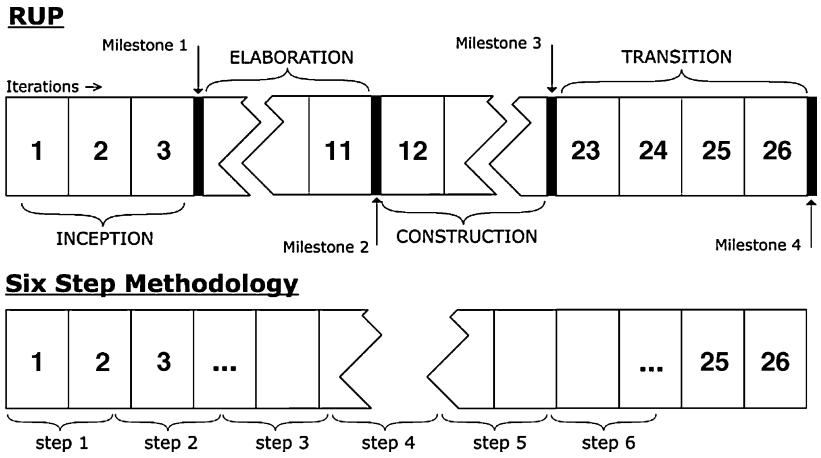

Fig. 3 – An example of alignment of a RUP-style project timeline and the six-step methodology.

tion (both strongly emphasized in RUP), and domain specific knowledge reuse, IT projects in the medical and healthcare domain will not deliver the expected benefits.

3. The methodology

Our methodology consists of six steps. From a simplistic point of view, one can argue that a six-step methodology is just another version of the previously used waterfall model [8], which has been criticized for the lack of flexibility achieved 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 iterations as short time chunks that are forming the steps, during which a small part of the required functionality is implemented. In this view the methodology can be seen not only as an extension of RUP, but also of any iterative methodology, such as Agile Development [11] or XP (extreme programming) [12], where short iterations that are strongly time-boxed (i.e. have to start and end at a very precise time) are part of more "elastic" phases. Methodological constraints, not project dates, define these phases, whose ends can be seen as conceptual milestones for the project manager—of course, in strong relation with time/budget milestones. Hence, 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 intermediate results, and eventually change requirements for the next iterations.

For example, RUP has four phases (inception, elaboration, construction, and transition), each consisting of a number of iterations. Typically, the inception takes 10–15% of the time budget, the elaboration takes 20–35%, the construction phase 30–50%, and the transition phase 10–30%. A 1-year project, consisting of 26 two-week iterations, can be structured as a {3,8,11,4} 2-week iterations sequence over the four phases (see Fig. 3). The first milestone is the "go/no-go" decision, the second is about freezing the software requirements, the third is the moment when the software is completely built, and the last is achieved when the software runs normally in the target organization. The alignment of our six-step methodology with a framework like RUP is easily achievable by leaving in place the iterations and milestone structure, and superimposing the six-step structure over the existing software project plan (as in Fig. 3), as a supplementary methodological guideline.

It is also possible to use the six-step methodology as a standalone framework. Nevertheless, we strongly recommend the use of an iterative substructure for project management.

3.1. *Step one—prove that the hardest requirement can be solved*

The first step, which 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 building the data warehouse. 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 a 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 in a LTC warehouse (at borough, county or national level) is that the data should be anonymous according to the UK data protection 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. 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 "prove" data owners and stakeholders that it is possible to fulfil this requirement. Here we have used a mapping file, owned by the owners of the data sources. In a mapping file, the unique identity of the patient (given by his social security number) was associated with a unique identifier generated by

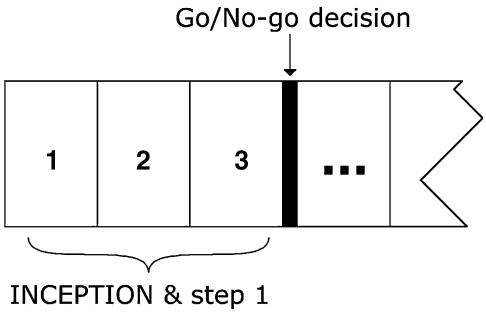

Fig. 4 – RUP's "inception phase" and our "step one" may be aligned.

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.

All stakeholders, who have their data added to our central repository (kept at the University), must be convinced that all the data we keep and use for model testing and validation are anonymous. To this effect, we have developed a schema involving a "cookie-based", stakeholder owned key-mapping mechanism, that allows the cyclic update of the central repository without using names, social security numbers or other non-anonymous information from the local records.

For the project management, reaching this milestone (i.e. solving the critical aspect(s)), can be also viewed as the "go/no-go" decision point in the project (identical with the end of the "inception" phase in the RUP framework). A proposed alignment with RUP is shown in Fig. 4.

For example, in nation-wide data-gathering undertakings, 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 [14] that ensures data privacy. Healthcare networked data repositories typically use a public key infrastructure (PKI), but experience shows that the implementation of this infrastructure could be more costly and politically sensitive than the gathering of anonymous data (for a study of the trade-offs with a PKI-based system, see Ref. [15]). 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 [20]. For small-scale data gathering, we advise a customised, less costly scheme than the above ones.

Finally, we advise that for data gathering in general, one should identify the critical requirements first. Typical requirements are high levels of data privacy, data accuracy, analysis speed, ability to filter large amounts of data, good connectedness, short-term data availability, etc. After identifying what it is the most important requirement for the stakeholders, a solution (at a formal or at an empirical level, depending on the problem) should be selected. The solution has to be clearly feasible at this stage, from a human, ethical, financial, and technical point of view. It could be a standard solution, but also an adapted one. Many projects fail because people select first a solution (usually a technology driven one), and try to fit it to the requirements. This leads often to the selection of requirements that are not the relevant ones.

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 the updating rate (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 otherwise they have to be reverse-engineered.

Time granularity is important for each database. In our case, some databases kept financial information on an annual basis, others on a monthly or weekly basis. 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. Often stakeholders ask late in the project for new data, which exist in the sources, but the designers are not aware or prepared to tackle these new requirements. If designers 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 concerns complementarities of data sources, usually leading to 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 volume of data is low but the structure of the data is very complex (for instance 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 last iterations of the inception phase, or at least in the very first iteration of the elaboration phase. It will end after the first iterations of elaboration phase (see Fig. 5).

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

The third step is to determine the data requirements to achieve 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

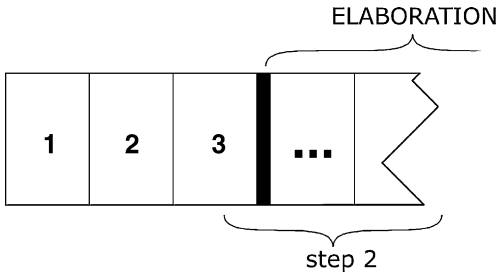

Fig. 5 – The alignment of the second step with RUP.

of the analysis tools that have been already developed. However, the implementers must realize that it is possible that new tools and models may be required in the future, in addition to the current ones, and that data that are not currently used may be valuable in the future. Another aspect in this step is 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 wait several 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, data that are frequently used are buffered in small "caches", 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 easily reachable, and data that are not, could be placed "at distance" from the main query starting points. If necessary, new queries can be written for new tools, or the old queries can be rewritten. In extreme cases, the whole database schema 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 a RUP-based project, the 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 is given in Fig. 6.

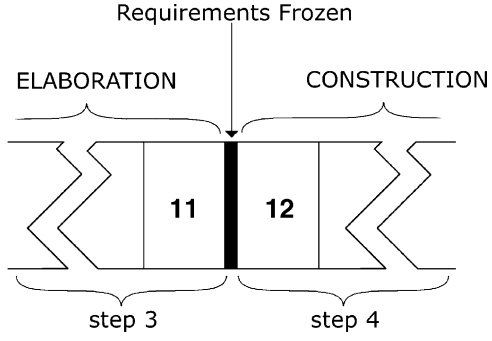

Fig. 6 – The alignment of steps three and four with RUP.

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 and healthcare IT systems (see Ref. [16]). Currently, there are various proposals for Semantic Web ontologies, specialized for various fields of biomedicine and healthcare (see Ref. [21]).

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. We have observed that different people, from different environments and with different backgrounds use local specialized languages that are incomprehensible for the outsiders. They use sometimes the same words, but with slight differences in their meaning. This makes it enormously difficult for stakeholders to discuss together their local decisions. A very important byproduct of the development of data integration is that stakeholders who 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 like in Fig. 6.

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 can also 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

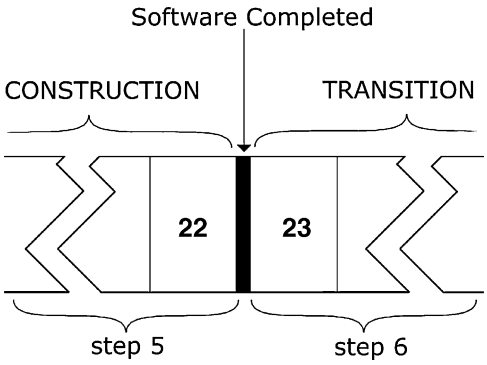

Fig. 7 – Linking the last two steps with the RUP framework.

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 implies that the update mechanisms should be fully functional at this point. Hence, step five is part of the construction as shown in Fig. 7.

3.6. *Step six—enact exception handling protocols*

The sixth step is the first real-life data gathering, and tests if the filtering software deposits correctly the right information into the repository. The collected data should be analyzed immediately with the simplest methods and compared with the results of the analysis of sources using the same methods, to detect anomalies that are induced by the data gathering process. In our case, we have discovered in this phase that it is necessary to enact some methods to "clean" automatically noise from the gathered data. 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 simple queries that eliminate these. In other projects—with different nature of information gathered, 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 users should not work already with the system, take this opportunity to obtain final training for system use.

Step six (as in Fig. 7) 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 application environment is prone to a high number of exception handling occurrences.

4. The six-step methodology and project management

Our proposed methodology only shows "what should be done" and in which order 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 (in Section 3), but we have not offered yet clear guidelines on how to plan the project, how to assign the plan to a team and how to link these to a budget and control mechanism.

At the first glance, the methodology seems to be best suited for the use of the so-called "predictive model" (see Ref. [8]) for project management. In software development, the predictive model is still used today because of the participants' 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 them 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 still be low. The second stage is much more predictable because plans are already in place.

There is a debate about whether software projects can be really predictable [9]. 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 late in the development process. The requirement changes usually make a predictive plan impossible. Freezing the requirements early on and not permitting changes is possible, but this leads typically to the outcome of delivering a system that does not fulfil 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 requirement changes are 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 at the beginning in the former, and continuously in the latter. Changing requirements will solely change the 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. Also, we should extend the framework by including team organization advices, 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 four (ontology construction). Of course, for environments where the interaction between stakeholders is not really important, this step can even be ignored. However, in our specific case, the building of the ontology triggered a conscious effort by the stakeholders to improve each other's understanding. We would like to investigate if this happened in other projects and to extend the ontology building process from a mere step in the framework to a full process that takes place in parallel with, and supports the development process from the beginning. Also, it may be useful to study the novel technologies arising from ontology building research in the field of the Semantic Web, especially investigations in the Genome Ontology Project involving huge data gathering (see Ref. [10]). These can bring new insights into 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 main problems with any framework for software development or other domains is that it typically covers only partial aspects and too many gaps are left, especially at conceptual and implementation levels. However trying to fill these gaps always reduces the comprehensibility and usability of the framework, because the potential user becomes lost in too many details. To learn how to use our framework, a user needs to try it out and fill the gaps her/himself. 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 the methodology could be very useful, especially for environments with lots of heterogeneity, in terms of conceptual domains, people skills and backgrounds, goals; and containing heterogeneous information systems.

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. These steps are: recognizing and resolving the most critical issue(s); identifying the scope and granularity of data sources, including overlapping data; identifying the most needed data in the repository; ontological alignment of the data and also of the people; determining update policies, validating 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 four) is playing the role of third trusted party that teaches 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. An insight into the problems faced by others (by understanding their language, the data, and the results of the analysis of the 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 Social 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 partly supported by the Engineering and Physical Sciences Research Council (GR/R86430/01).

REFERENCES

[1] DOH, Making it happen—the key areas for action. http://www.publications.doh.gov.uk/ipus/socialcare/happen/, accessed November 27, 2004.
[2] NHS, Single assessment process (SAP) dataset. http://www.nhsia.nhs.uk/datasets/pages/ds_older.asp, accessed November 27, 2004.
[3] C. Pelletier, T. Chaussalet, N. Szirbik, C. Vasilakis, Integration of data on long-term care from heterogeneous sources for research purposes, in: Proceedings of the MEDICON'04, Ischia, Italy, 2004.
[4] H. Xie, T. Chaussalet, P. Millard, Continuous-time Markov models for the flow of elderly people in institutional long-term care, J. R. Stat. Soc., Ser. A 168 (2005) 51–61.
[5] C. Pelletier, T. Chaussalet, H. Xie, A framework for predicting gross institutional long-term care cost arising from known commitments at local authority level, Operat. Res. Soc. 56 (2005) 144–152.
[6] H. Xie, T. Chaussalet, W. Thompson, P. Millard, Modelling decisions of a multidisciplinary panel for admission to long-term care, Health Care Manage. Sci. 5 (2002) 291–295.
[7] H. Xie, Modelling Issues in Institutional Long-term Care: Placement, Survival and Costs, Department of Computer Science, University of Westminster, London, 2004.
[8] P. Kruchten, The Rational Unified Process. An Introduction, 3rd ed., Addison Wesley, Object Technology, 2004.
[9] M. Fowler, The New Methodology. http://www.martinfowler.com/articles/newMethodology.html, accessed March 19, 2004.
[10] Gene Ontology Consortium, An Introduction to the Gene Ontology. http://www.geneontology.org/GO.doc.shtml, accessed March 19, 2004.
[11] http://www.agilemanifesto.org.
[12] http://www.extremeprogramming.org.
[13] R. Lenz, K.A. Kuhn, Towards a continuous evolution and adaptation of information systems in healthcare, Intl, J. Med. Inform. 73 (2004) 75–89.
[14] G. Kelly, B. MacKenzie, Security, privacy, and confidentiality issues on the internet, J. Med. Internet Res. 4 (2.) (2002).
[15] http://www.merit-9.mi.hama-med.ac.jp/.
[16] H. Kanoui, M. Joubert, G. Maury, A semantic-based kernel for advanced health information systems, Med. Inform. 25 (1) (2000) 19–43.

[17] I.M. Carey, et al., Developing a large electronic primary care database for research, Int. J. Med. Inform. 73 (2004) 443–453.
[18] M.C. Müller, et al., Cross-institutional data exchange using the clinical document architecture, Int. J. Med. Inform. 74 (2005) 245–256.
[19] K. Bernstein, et al., Modelling and implementing electronic health records in Denmark, Int. J. Med. Inform. 74 (2005) 213–220.
[20] L. Ohno-Machado, et al., Protecting patient privacy by quantifiable control of disclosures in disseminated databases, Int. J. Med. Inform. 73 (2005) 599–606.
[21] L.F. Soualmia, S.J. Darmoni, Combining different standards and different approaches for health information retrieval in a quality-controlled gateway, Int. J. Med. Inform. 74 (2005) 141–150.
[22] J.H. van Bemmel et al., Databases for knowledge discovery—examples from biomedicine and health care, Int. J. Med. Inform., in press.
[23] H. Xie, T. Chaussalet, W. Thompson, P. Millard, A simple graphical decision aid for the placement of elderly people in long-term care, J. Operat. Res. Soc. (2006), doi:10.1057/palgrave.jors.2602179.