Lessons Learnt from the Analysis of Large-scale Corporate Databases

Barbara Kitchenham
National ICT Australia
Bay 15, Australia Technology Park
Garden St, Everleigh
NSW 1439, Australia
Barbara.Kitchenham@nicta.com.au

Cat Kutay and Ross Jeffery
National ICT Australia and
University of New South Wales
Bay 15, Australia Technology Park
Garden St, Everleigh
NSW 1439, Australia
ckutay@cse.unsw.edu.au &
Ross.Jeffery@nicta.com.au

Colin Connaughton
IBM Global Services A/NZ
Level1, IBM Centre
601 Pacific Highway
St Leonards, NSW 2065
Australia
colinc@au1.ibm.com

ABSTRACT
This paper presents the lessons learnt during the analysis of the corporate databases developed by IBM Global Services (Australia). IBM is rated as CMM level 5. Following CMM level 4 and above practices, IBM designed several software metrics databases with associated data collection and reporting systems to manage its corporate goals. However, IBM quality staff believed the data were not as useful as they had expected. NICTA staff undertook a review of IBM's statistical process control procedures and found problems with the databases mainly due to a lack of links between the different data tables. Such problems might be avoided by using M^{3}P variant of the GQM paradigm to define a hierarchy of goals, with project goals at the lowest level, then process goals and corporate goals at the highest level. We propose using E-R models to identify problems with existing databases and to design databases once goals have been defined.

Categories and Subject Descriptors
D.2.8 D.2.9 [**Software**]: Software Engineering – *metrics, management.*

General Terms
Management, Measurement, Performance.

Keywords
Software metrics databases, corporate data, M^{3}P, GQM

1. INTRODUCTION
Metrics researchers have often complained of the sparseness of software measurement data. For example, one of the reasons researchers are interested in constructing cross company estimation models is because individual companies may not have sufficient data or the necessary resources to build their own models (see for example, Maxwell et al., 1999). However, the observations reported in this study suggest that it is not always useful to have a very large amount of data if the software development and maintenance process that generates the data is not well understood by those analyzing the data or the data structure does not clearly represent the intricacies of actual working practices.

IBM Corporate Services Australia (referred to as IBM in this paper) is a CMM level 5 company. In addition, as of 2005 all CMM practices have been transitioned to CMMI. In line with CMM level 4 and above practices, IBM implemented several large corporate databases to manage their corporate goals. The databases maintain data about the size, effort, schedule, and defect rates for each piece of work undertaken by the company. A number of the metrics adopted were based on those reported by companies who have achieved accreditation at CMM level 5. However, IBM quality managers felt the data were not providing appropriate support for software process control, nor were they proving as useful as might be expected to software project managers. Therefore, IBM staff asked researchers at the National ICT Australia Ltd (NICTA) to review their software process control practices. As part of that investigation, we reviewed the various corporate databases.

Although the problems we report come from a single company, we believe they are of more general interest because many large companies are increasing their data collection activities in order to achieve CMM-style accreditation. For example, Levels 3 and above require the collection and use of measurement data and in particular, CMM Level 4 requires the use of statistical process control. However the generation of the required databases is not a trivial exercise. In this paper, we identify some of the problems that can affect corporate databases even if the database structure has proven successful in other companies. We, also, suggest some methods to address these problems.

In Section 2, we discuss the problems we observed with the specific set of large corporate databases. In Section 3, we illustrate how the use of E-R diagrams helps to identify potential problems with corporate databases. We discuss our conclusions in Section 4.

2. PROBLEMS WITH THE CORPORATE DATABASES
We were asked to review the statistical process control (SPC) activities of IBM because QA staff were concerned that the SPC

ICSE'06, May 20–28, 2006, Shanghai, China.

results (although compliant with CMM) were not proving as useful to project managers as had been hoped. This involved a review of the data held in the corporate databases and of the database user manuals and an assessment of the current analysis practices. We were unable to interview the managers who had commissioned or specified the databases.

The data used in reporting to Project Managers were contained in three main databases which can be described as:

- Inspection database including inspection defects, test defects and post-production defects
- Effort and size database
- Schedule database

The databases were each established to support a specific business/development process. The effort and size data were collected to enable management of projects against year-on-year performance improvement targets, which arose from commitments to the major client at the time. In addition, the client has contractual requirements for some data to be reported. Project schedule data were collected, initially for the same reasons. Defect data collection was based on collecting defect data arising from work product inspections. This was intended to support a process improvement initiative aimed at maintaining or improving defect rates while increasing productivity. Defect data were also subsequently used, as a project management tool, to predict defect levels in development. Project on-time and on-budget data are now collected for all projects as a useful quality indicator. In this manner each database supports a specific business/development process or goal. What was not addressed, however, was the set of broader relationships between the entities – for example, between a project, an inspection, a development defect, a software release, a production problem and so on.

Now that the databases are established, it is also intended to support ongoing performance management and improvement aimed at outperforming other service suppliers. Most data are used for benchmarking. In this paper we see that some of these improvement initiatives cannot be supported by the existing data structures.

Initial problems arose because it was difficult to obtain a full specification of the databases and the data extracted for analysis was partial and fragmentary. We also encountered difficulties analysing the data that were due to problems with the data tables. These are discussed below.

2.1 Effort and Size Data

In general, effort and size data were not a problem. Effort and size data were recorded against a record key defining a specific unit of work, called a Work Order.

One goal of the corporate data collection activity was to monitor productivity. However in order to do this, it is necessary to maintain information about a variety of specific factors that might affect productivity. The company identified the following factors:

- *Application name*. Most of the work being undertaken by the organization was maintenance work. Thus, it is necessary to link the maintenance work to the specific application being developed, since application differences (such as application size or maturity) might be a cause of productivity differences among maintenance tasks.
- *Platform*. The same application might have mainframe, mini, or midi components (e.g. client-server applications). Thus, platform information was required.
- *Team location*. Different teams might work on the same applications at the same site or different sites. So it was necessary to record the location where the work was performed.

One major problem with the effort and productivity data was the lack of a single unique key in the effort-size data table. The database included a work order identifier, however, it transpired that different teams working on different components of the same application might use the same work order number even if they worked at different sites using different platforms. Thus, each individual record in the database has a multiple key, which is a combination of work order identifier, application name and team location. Furthermore this multiple key is not unique, in that a work order may be divided into different projects on the same application and platform. This does not present a problem for productivity analysis since the work done by a specific team is probably the most appropriate analysis unit. However, it does present a problem if you want to link effort data to defect data or delivery date data.

The size of all projects was measured in function points using the International Function Point Users Group (IFPUG) 3.2 standard. In this context, the size of maintenance projects measures the changes made to the application being enhanced, the size of a development project is the total size of the application produced by the project. A problem with the size measurement was that only final size measures were available. Thus, it is not possible to confirm that function point measures would have been good predictors of effort at an early stage in development. Nor is it possible to assess whether the estimate of size changed during the life of the project, for instance when requirements changes are incorporated into an existing project. This may have a significant impact on productivity assessment, since a major change in requirements may reduce overall productivity. For example, if project managers plan for a project of a particular size and subsequent changes in requirements mean they have to change their plans, their development team may have lost effort working on original requirements that are no longer necessary or need to be changed. This wasted development work may not be adequately reflected in the final change size, leading to an apparently low productivity level for the project.

2.2 Defect Data

The defect data exhibited a large number of problems. Some of the difficulties occurred because the database included both testing and inspection data but the distinction between the two was made by inference (i.e. the inspection effort field was non-zero) rather than by a simple attribute. However, the major difficulty was that inspection and test data were not linked to a single defined unit of work. This meant that it was impossible to identify:

- All inspections corresponding to a specific work order
- All defects found by testing the application developed or enhanced by a specific work order.

This lack of linking information means it is not possible to use the data to monitor defect detection for a specific work order across

different software development phases. The best that can be done to provide stratification of data is to identify all inspection and test data for particular team location and a particular application included in a specific release. Since defects are linked to release whereas effort and size data are linked to work order, productivity and defect data cannot be directly linked, so the results of any process improvement activity are difficult to assess. That is, it is not possible to assess the relationship between productivity and defect-related measures of quality for a specific work order. It is important to monitor the trade-off between quality and cost or efficiency during improvement. However, in this repository the direct impact of changes in working methods cannot be fully evaluated at a uniform level of granularity.

Furthermore inspections were categorized according to the work product type, rather than the nominal project phase in which the inspection occurs, or the type of document format used. So any inspection of test documentation (plans, scripts, whatever) is treated as a Test phase inspection, regardless of when in the life cycle the inspection took place. Hence any comparison of defect insertion or extraction across phases will not be a measurement of consecutive reviews, but of work product document types.

Also it is not possible to distinguish between a code inspection and a (low level) textual and graphical design document inspection, since both are classified as Build/Code phase inspections. This is a particular problem for statistical process control. Both Weller (2000) and Florac et al. (2000) make it clear that statistical process control of inspection data requires a homogeneous data set. Weller analyzed only code inspections and needed to separate inspections of new code from inspections of modified code. Similarly when Florac et al. analyzed code inspections, they needed to analyze inspection data from different subsystems separately and to make an ad-hoc separation between inspection data, based on inspection rate, before they were able to construct control charts for inspection effort. In the IBM repository, the lack of automated linkages to the inspection reports, and hence information about the specific document type, makes it difficult to analyze the effect of either different product types or document formats.

Further problems with the data set included:

- Ill-defined Process. The process stages are defined by the project manager at the start of a project, and correlated to the standard phases as modeled in the system. Hence maintenance projects are modeled as subsets of a full lifecycle model.
- Ill-defined Records. The database allows users to define a record as belonging to the Operations and Maintenance (O&M) phase but then it is not clear what test data and inspection data mean in the context of O&M. Furthermore the user manual says that the O&M phase can be used to record any information users want. This means that the contents of records if the phase is identified as O&M are not defined.
- Missing data. Inspection data values are averaged over document build effort to provide a standard unit for analysis but the build effort value was not always recorded. Also preparation data is not separated from inspection meeting data.
- The use of estimates instead of actual measurements. Document build effort is estimated by the person submitting the data, since the actual data are not available from any other source (such as timesheets used in the productivity database).
- Lack of a simple means of identifying whether an inspection referred to new or enhanced code.
- Inability to identify re-inspections.

2.3 Schedule data

Schedule data had different problems. The schedule data are collected in order to manage performance in terms of meeting the promised delivery dates as a major project goal and later an organizational marketing strategy. However the database allowed only the recording of the original estimate which may possibly be changed over time, the outlook date which tends to match the actual completed (and is used for rescheduling out of schedule projects), and the actual completed. This meant that the data was not sufficient to analyze estimation and planning performance and hence improve these sub-processes. It was impossible to determine whether the schedule had been well-estimated, or the estimate had been continually changed to reflect the actual schedule. Like the productivity database, there is no link to information about changes in requirements that might explain any deviation between estimates and actual schedules.

In addition, the problem of non-unique work order numbers in the effort-size database meant that even if the delivery date data had maintained a historical record of estimate changes, it would be difficult to link effort and schedule information. The schedule database records are indexed by work order and an account code (which does not appear in the productivity data extraction) but not by the multiple key (work order, application, team location) used to specify records in the effort-size database. That is the schedule data identifies the duration of a set of linked work orders not the duration of the individual component of a work order undertaken by an individual team at an individual location.

3. DESIGNING CORPORATE DATABASES

In our view the problems outlined in Section 2 make it clear that corporate databases need to be designed and should not be assumed just to be a central repository of any available software development data. The common approach of QIP/GQM suggests that metrics should derive from a set of goals (or requirements) and associated questions (see Rombach and Basili, 1990, or Soligen and Berghout, 1999). In our view, there is no reason to believe that the goals or requirements for project data are necessarily the same as they are for corporate data. In the M^3P approach Offen and Jeffery, 1997, recommend extending the GQM paradigm to provide links between the metrics collected and the surrounding development and business contexts. They suggest that companies need a hierarchy of goals with project goals at the lowest level, process goals at the next level and corporate goals at the highest level. This is a bottom-up composition approach. It is not clear that a system designed to support corporate goals will automatically decompose into software process goals and then into software project goals.

The conflict between project and corporate viewpoints is clearer when the structure of the databases are modeled. For example, Figure 1 is a simplified data model of the current set of databases. It illustrates the problem that:

- A "work order" against which effort is collected is only a component of the "work order" against which schedule data is collected.
- Individual Defect records are related to a specific application, a specific location and a specific release but not an individual "work order". Therefore, defect data cannot be linked either to records in the schedule database or to records in the effort-size database.

It is difficult to see how the current IBM database can be evolved to address all the requirements being placed on it by different stakeholders groups. However, we suggest the use of a generic ERA model that may help other organizations avoid these problems.

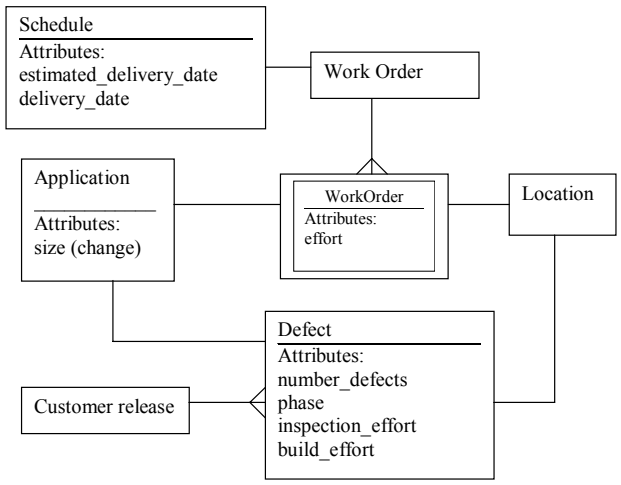

Figure 1 Current corporate data model

Basic software engineering principles require a specification of any management information system prior to implementation, which should include something similar to an ERA model. Furthermore the basic software development process consistent CMM and CMMI is well enough understood for a template ERA model to be useful. Figure 2 shows a data model that might be more appropriate for an integrated corporate repository as opposed to a set of independent databases. The data model is a straightforward representation of a "standard" software lifecycle consistent with any software engineering textbook (e.g. Pressman, 2005).

A critical element of this data model is that it identifies the subproject as the basic software production entity against which data is collected. Other issues are:

- A customer release is made up of a set of "projects". Each project may be broken down into one or more subprojects.
- Each subproject corresponds to work undertaken by an individual team at a specific location to develop or enhance an application.
- Applications exist on one or more platforms (mainframe, mid-range, micro), but a specific subproject refers to work on a specific platform.
- Each subproject is divided into phases. Test phases use executables and produce documents (this assumes that test plans and test specification are produced during the testing phase). Construction phases produce documents. Documents are inspected. Note Both Construction phases and Documents should have an attribute that defines their type.

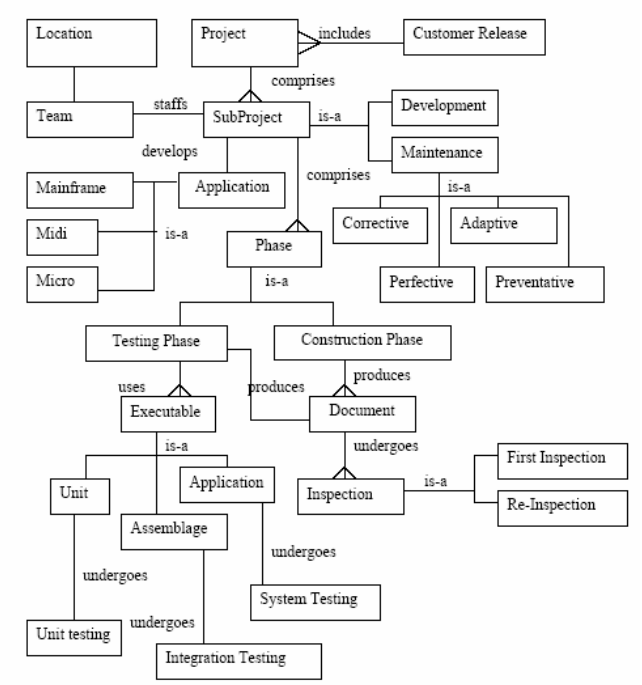

Figure 2 Data Model of Software Development from Corporate Viewpoint

- Each subproject should be categorized to identify whether it is an enhancement or a development project and each enhancement project should be categorized to define the type of enhancement.

The viewpoint of an individual development team is incorporated in the subproject representation and the corporate viewpoint incorporates the totality of projects included in a customer release. This data model is not necessarily correct but can act as a starting point for discussing the structure of a central repository once the requirements for an integrated data repository have been identified.

However, there are still practical problems with this representation:

- Complex projects are decomposed into subprojects and subprojects become the object against which software production data is collected. However, there will be preliminary design effort to define the requirements for each project and specify its breakdown into subprojects. This effort is a project level attribute. It is not clear how this effort is incorporated into productivity assessment.
- In some organizations, system testing (and integration testing) is not performed by the development team but by a

testing team whose responsibility is to test a complete application. In this case, integration and testing effort may only be collected for an application (for a specific release). Again it is not clear how this effort can be allocated to individual projects or subprojects. It may also be difficult to trace integration and system testing defects to the specific project or subproject unless defects are classified against the project or subproject when they are cleared. If information about testing defects is recorded by an independent testing group, the testing group may not be able to trace a defect back to a specific project or subproject.

- If teams are distributed across different locations or more than one team works at one location, a separate team identifer is required as well as a location identifier.

The issue of modeling the development process in order to determine appropriate measurement processes is also addressed in part by Lavazza and Barresi (2005) who are developing a toolset to link GQM plans with a software process model. They recognize that it is necessary to link software metrics to the development process but they do not present any data to support their hypothesis.

4. DISCUSSION AND CONCLUSIONS

Our results show that corporate databases may not prove as valuable an asset as might be expected. The design of an integrated data repository needs to model an organization's software development process (see M^3P framework developed by Offen & Jeffery, 1997). In the IBM case, lack of an overall design resulted in the data being stored in several discrete databases with links between the data items not being maintained. Our results indicate that a high CMM maturity level, a large quantity of data incorporating metrics related to the basic measurement dimensions of size, effort, defects and schedule are not enough to ensure that corporate data will become a valuable corporate asset. These factors may be necessary but they are demonstrably not sufficient. In other words, an effective and unified database will not develop naturally from an uncoordinated set of activities.

We believe that a failure to specify project and process goals/requirements separately from corporate information goals may have contributed to difficulties using corporate data. It may have been assumed that corporate databases are simply a concatenation of available software production data, but we believe that this is not the case. If data are collected against a project but intended for inclusion in an integrated corporate repository, data requirements at the corporate level may be different from those at the project level. There is a danger that context information, including the links between project entities needed for project level and process level analysis, will be lost if the requirements are not made explicit at each organizational level. We suggest that practitioners apply disciplines like M^3P at each relevant level: project, process and corporate.

In order to maintain software production data in a manner suitable for meaningful analysis, it is necessary to define the basic software entity about which data is to be collected. We suggest that the basis software entity for data collection should be a specific unit of work (or project) undertaken by a particular software development team. If a complex project is decomposed into subprojects undertaken by different teams, the individual subproject should become the basic data collection entity, although information about the project will also need to be collected. Although the project or subproject is appropriate for project level data, it may need to be concatenated or decomposed in different ways to provide information appropriate at the process or corporate level. Process information is often analyzed against lifecycle phases, which require project data be decomposed into relevant phases. In contrast, corporate data may need to be analyzed against produce release schedules, which require project data to be concatenated, so that all the projects included in a particular release can be analyzed as a whole. It is essential to collect the data need to construct different views of the data corresponding to each level in the information hierarchy.

We also suggest that the logical structure of a corporate data repository should be developed from an E-R model of the software process and its products; i.e. the entities that will be measured. This should help to ensure that links are maintained between measured entities and data can be collated/extracted at an appropriate level of granularity. However for E-R diagrams to be practical, it is necessary both to identify the goals (requirements) for subproject, project, process, and corporate information and to ensure that software metrics are well enough defined to avoid "losing" data, such application testing effort, or preliminary design effort

We speculate that the CMM and CMMI requirements to apply statistical process control may contribute to database structures that treat defect data, productivity data and schedule data as separate independent data sources. Simple statistical process control (SPC) is based on control charts related to a single outcome measure. In conventional SPC applications, a specific manufacturing process produces numerous items of exactly the same type to pre-defined specifications. It is relatively simple to confirm whether or not individual items are within specification. Furthermore, if items are out of specification, it is usually correct to assume that the process is somehow at fault. However, software processes do not produce identical outcomes so assessing whether an item is or is not within specification is less clear cut, for example there may very good reasons why a project exhibited unusually bad productivity or unusually high defect rates that have nothing to do with the process being at fault. As mentioned in Cangussu et al.(2003), a process may be in control, while not falling within specification limits. In order to interpret software measures correctly, data of different types needs to be collated and analyzed jointly. Thus, the data requirements for SPC in a conventional manufacturing environment differ from those needed to support software process monitoring and control. A literal translation of SPC procedures from manufacturing to software development may be counter-productive.

5. REFERENCES

[1] Cangussu, Joao W., DeCarlo, Raymond A., and Mathur, Adityar P., "Monitoring the software test process using statistical process control: a logarithmic approach," *Proceedings of the 9th European Software Engineering Conference* (2003), 158-167

[2] Florac, William A. Carleton, Anita D., and Barnard, Julie R. Statistical Process Control: Analyzing a Space Shuttle

Onboard Software Process*, IEEE Software* (July 2000), 97-106.

[3] Lavazza, L. and Barresi, G. Automated support for process-aware Definition and Execution of measurement Plans. *Proceedings of the International Conference on Software Engineering, IEEE Computer Society* (2005), 234-243.

[4] Maxwell, Katerina, Van Wassenhove, Luk, and Dutta, Soumitra. Performance evaluation of general and company specific models in software development effort estimation, *Management Sciences, 45*, 6 (1999), 787-803

[5] Offen, R and Jeffery, R. Establishing Software Measurement Programs*, IEEE Software* (March/April 1997), 45-53.

[6] Pressman, R.S. Software Engineering. 7th ed., Addison-Wesley, 2004

[7] Rombach, H.D. and V.R. Basili. 'Practical benefits of goal-oriented measurement', *in Proc. Annual Workshop of the Centre for Software Reliability: Reliability and Measurement*. Garmisch-Partenkirchen, Germany: Elsevier, 1990.

[8] van Solingen, Rini and Berghout, Egon. The Goal/Question/Metric Method. *A practical guide for quality improvement of software development*. McGraw Hill Publishing Company, Maidenhead, UK, 1999.

[9] Edward F. Weller. Practical Applications of Statistical Process Control, *IEEE Software* (May 2000), 48-55.