Consequences of Postponing Data Modelling to Allow Flexible Data Collection

Steve Draper
University of Glasgow
s.draper@psy.gla.ac.uk

Richard Thomas
University of Western Australia
richard@csse.uwa.edu.au

Rebecca Mancy
University of Glasgow
mancyr@dcs.gla.ac.uk

Abstract

Current technology is opening up the possibility of interactive investigations, where data analysis is only determined after data collection. What might normally be called data cleaning thus becomes a major phase of analysis and software development since it must take over the functions formerly performed by analysis before any data was collected. Early experience of this is described, and its implications discussed.

1. REDDIs

Falling prices and expanding capacity on bulk digital storage, together with increased network bandwidths allow us to consider collecting more data than ever before with less and less need for a precise advance idea of its uses. This is occurring both in research and in industry (e-commerce purchase logging, stock market). In support of this, data mining has also provided examples of important discoveries from data collected for another purpose. The Generic Remote Usage Measurement Production System (GRUMPS) [2] [3] project is exploring these current and emerging technical possibilities.

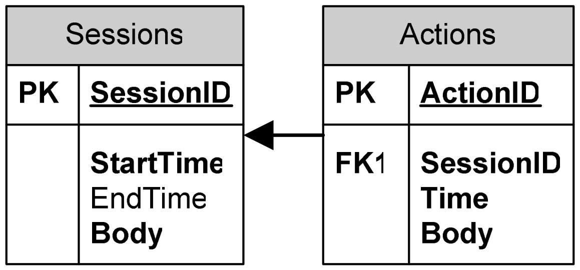
The new flexibility supports investigations that are neither traditional database approaches, where collection and processing are carefully designed in advance, nor data mining where again collection is fixed in advance, but then retrospective and often speculative analyses are done post hoc. We call this emerging middle ground REDDIs: Rapidly Evolving Digitally-Derived Investigations. These may have no prior hypothesis, do not usually control or manipulate the circumstances being studied, and often involve changes in what is collected in the light of successive examinations of the data.

In the GRUMPS project, collection is from any source that produces digital data e.g. CCTV. The collector used for the work described here was the User Action Recorder (UAR). The UAR can run on any Windows machine, using Windows system calls (events) to capture keystrokes, mouse moves, window events etc. without having access to the source code of the applications. The example we discuss here is the collection of such data in a first year student programming laboratory. The UAR was used to collect user interface events from the first year Ada programming laboratories at the Department of Computing Science, University of Glasgow. In spring 2003, 141 students consented to participate, yielding 4.7 million events over 2655 user sessions (periods between login and logout).

The first question that we investigated using this data aimed to look for an indication of student persistence with programming tasks by looking at pause times in the programming environment [7].

2. The Collection Data Schema

A consequence of collecting data without being aware of its potential uses at system design time, is that there should be minimal fixed attributes in the data model. Much of what is particular to the case should be encoded in XML fields, and the repository schema optimised for rapid collection.



```
<p>gexecute.exe</p><wl>23</wl><wt>
54</wt><wr>916</wr><wb>678</wb>
<ws>nor</ws> …
```

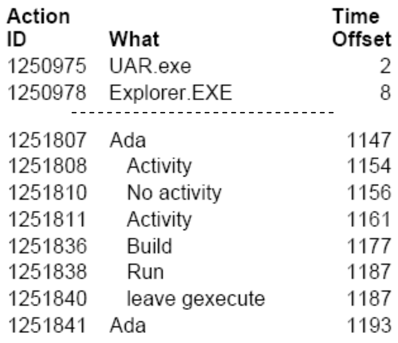
Figure 1. The collection data schema

However, proceeding in this manner does not avoid, but only postpones, the work of data modelling – developing a structure that reflects the real-world meaning of the data – that is classically carried out at design time. Data modelling may be carried out in various paradigms such as object oriented, aspect oriented, procedural, or in the case of database applications, ER (Entity Relationship) analysis.

3. The data transformations required

Intuitively, it is easy to see that the UAR and the data it collects hold possibilities for multiple and varied investigations. Generally speaking, the data is particularly useful for studies that are longitudinal in nature, that are concerned with usage across applications or that may evolve over time. For example, a study has been looking at keystroke data to determine typing speeds and set-up times and their evolution over time, linking back to success in programming. The same data are being used for researching how people recover from interruptions [6] and another researcher is interested in possible differences between window layout between male and female subjects.

The data collected from the UAR are very general, and so useful for multiple investigations. However, they are recorded at a very low level of abstraction and it is therefore a difficult task to search these data for patterns related to the high-level research questions detailed above. In order for the data to be useful to the investigators, they must be transformed to a level of abstraction appropriate to the research question. For example, in the investigation concerning perseverance with programming tasks, data at the level of individual keystrokes is not useful, but become so, re-expressed in terms of user-sessions, and summaries of the number of compiles, builds, and runs.



Action ID	What	Time Offset
1250975	UAR.exe	2
1250978	Explorer.EXE	8
1251807	Ada	1147
1251808	Activity	1154
1251810	No activity	1156
1251811	Activity	1161
1251836	Build	1177
1251838	Run	1187
1251840	leave gexecute	1187
1251841	Ada	1193

Figure 2. Example client's representation

Data processing is therefore organised into two major phases. These phases join at a "lynch pin" data format, the *client's representation*, that allows the data to be represented, not at the level of data collectors, but at a basic level of domain meaning. There is substantial software work to be done in both these phases: in constructing and computing this client's representation from the raw data, and then doing various further analyses on it. In the data mining area, the software work for phase 2 would be carried out using various statistical tools, whereas in more traditional database applications this stage would be done by "reports": various custom-made data processing procedures. In both these areas however, phase 1 would be largely accomplished at design time, when the data model was laid down based on a careful but fixed analysis, and any processing done in effect by the database management system. What is different about supporting REDDIs is that phase 1 must be done after, not before, data collection, and often changes as the investigation, along with the client's ideas, evolve.

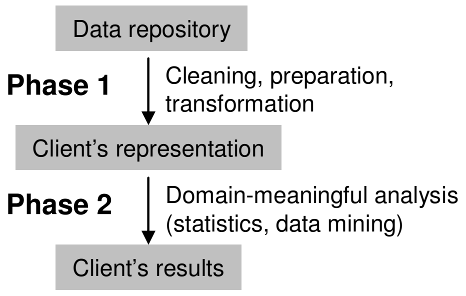

Data repository
Phase 1
Cleaning, preparation, transformation
Client's representation
Phase 2
Domain-meaningful analysis (statistics, data mining)
Client's results

Figure 3. The two major stages in data processing

4. Eliciting the client's representation

For the purposes of this discussion of phase 1 data transformation, we will assume the common case of a client who is an expert only in the domain being studied (in our example, novice programmer usage studies), and a consultant who must organise and carry out the phase 1 data preparation in order to deliver something meaningful for the client to process further.

The consultant must begin by eliciting from the client a representation of the data that the latter can work with. The client's representation may depend on many factors, such as the tools (spreadsheet, data mining packages etc.) to be used by the client, and the skill level of the client with these tools. However it will probably mainly reflect the client's understanding of the meaningful entities in this domain. In the case of the student perseverance investigation, the administrator and client worked together to elucidate a sensible form for the client's representation: figure 2 (and also figure 5).

However this analysis is in general not fixed for at least three reasons:

• The definition of an entity may change as understanding of the case improves during the investigation. For example, in a lab of student programmers, it is natural to assume that machine login sessions correspond to a single student and their work. However if it emerged that typical students logged into two machines side by side, or conversely that students often worked in pairs so that one login corresponded to two people's work (and typing speeds, etc.), then that way of identifying the domain-entities in the data would have to be changed.

• The client's interests will change during the investigation. Many of these changes in hypothesis will only affect phase 2 processing e.g. by requiring different statistical summaries or tests based on the client's representation, but sometimes they will require a change in phase 1 processing i.e. in the entities extracted. For instance, if instead of comparing compile, build, and run actions for each student, the client decided to treat sequences of consecutive compile-build-run as objects in their own right, then this would require a change in phase 1 data preparation.

• Several clients may come to base investigations on the same data, as mentioned above. Thus it is very likely that several distinct client representations will be required.

Thus the dialogue between the client(s) and the consultant needs to be maintained throughout the duration of the investigation. As the research questions change, the consultant needs to ensure that sufficient information is recorded to enable or facilitate the generation of the client's representation.

5. Technical implications for phase 1 transformation

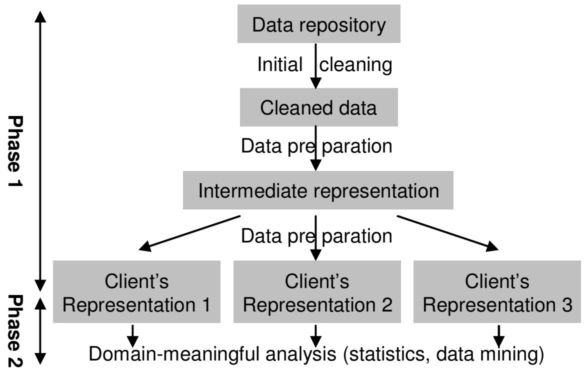
Once the client's representation is decided upon, the consultant has to go about transforming the data. Whilst providing for the client's needs, two other (partially conflicting) objectives are noted: (1) re-usability and (2) speed of query execution. These aim to reduce the effort and time spent generating the client's representation at each iteration or new investigation.

We chose to use the database to do the transformations. Given the large volumes of data, this was appropriate in order to benefit from indexing and parallel processing. It is worth choosing the DBMS carefully, to be sure that it offers the support required (native XML support, programmability). We chose to use SQL Server; because it is programmable using T-SQL, it is appropriate in the case of rapidly changing demands.

However, manipulating temporal data within the relational model brings with it difficulties [1]. Sequence information is lost unless stored explicitly and standard techniques for calculating durations, for example, may be very slow to execute. Our experience is that these difficulties can be circumvented, but new techniques need to be developed. These must be optimised to run in an acceptable time (a few minutes rather than a few days) so that development of the client's representation and consequently investigations are not held back.

Our experience showed that different investigations, because they were based on similar data, often shared concerns and it was therefore valuable to optimise techniques and queries for re-use. Using stored procedures provided a structure for this and allowed the storage of procedures rather than the intermediate data to avoid further augmenting the data volumes.

We also found it useful to group investigations according to shared concerns, and so to develop an intermediate representation between the raw data and that of several clients, thus allowing us to calculate several clients' data fairly easily.



Phase 1
Phase 2
Data repository
Initial cleaning
Cleaned data
Data preparation
Intermediate representation
Data preparation
Client's Representation 1
Client's Representation 2
Client's Representation 3
Domain-meaningful analysis (statistics, data mining)

Figure 4. Data preparation using an intermediate representation

6. An Illustrative Case

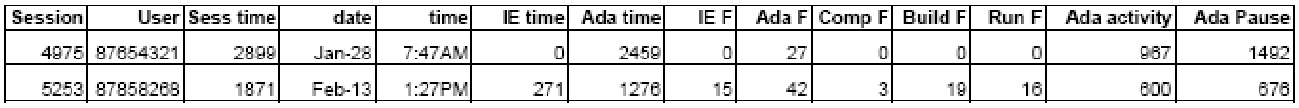
This case study draws on the example of searching for indicators of persistence mentioned earlier. Following ethics approval, data were collected on students studying the first year CS1P programming course in Ada at the University of Glasgow. The UAR was used to capture keystrokes and mouse events for user sessions, as students carried out fortnightly programming exercises using the AdaGide development environment.

Data were transported over the network and stored in the data repository, an SQL Server database, using the data model shown in figure 1. There are two main tables in the schema: Sessions, containing general information on each user session, and Actions, used to store details of each interface event. In total, 4.7M low-level user events were recorded with the Actions table using 2GB. At this time, the investigations that would use this data had not been fully determined.

The first question to be investigated using the UAR data aimed to identify factors that impact on students' persistence with programming tasks. Working with one of the authors as consultant, the client determined two reports that he required to carry out these analyses. A high-level summary report (figure 5) gave statistics such as total pause time using Ada, presented by user session. A second, more detailed report (figure 2) showed on-task and off-task behaviour over the course of each session.

There was a certain degree of overlap between the reports and in order to generate these, an intermediate representation was developed and both reports created from this, as in figure 4. The repository database management system was chosen to carry out transformations. A number of techniques were developed and optimised, to calculate, for example, the duration and context of an action, using self-joins. Tables were indexed to improve performance, achieving an 80-fold reduction in execution time.

The client's purpose in asking for the two reports was to aid him in searching for patterns of usage that might indicate persistence with a programming task. By looking first at the high-level report, the investigator could select particular user sessions as representative, investigate these further using the



Session	User	Sess time	date	time	IE time	Ada time	IE F	Ada F	Comp F	Build F	Run F	Ada activity	Ada Pause
4975	87654321	2899	Jan-28	7:47AM	0	2459	0	27	0	0	0	967	1492
5253	87858268	1871	Feb-13	1:27PM	271	1276	15	42	3	19	16	600	676

Figure 5. Top level report showing summary statistics (such as pause times in Ada) by user session

more detailed report, and finally the raw data, referring back to examination results. This client had already used a similar (phase 2) iterative technique using special purpose audit trails [5], with data captured by purpose built extensions to source code.

7. Implications

New technology is making flexible investigations based on digital data ever more attractive. However, one implication of supporting these is that data modelling must be done after, and not before, data collection. Work has not been avoided, but only postponed. The problem now appears as one of restructuring the data. The raw data is not unstructured as free English text would be, but does require transformation from a form that represents only the circumstances of collection, to another that is domain-meaningful. Projects addressing this must expect a major software support effort to cover this: what we call "phase 1" data preparation. Furthermore the process of analysis needs to be fast and easy to update (in contrast to the circumstances of most data modelling hitherto); and to support partially overlapping analyses for clients using the same data for different purposes. In the programming world, these might be met by facilities for incremental change, version control, change management, rapid prototyping etc. Support for such approaches might now be much more important in analysis tools designed to produce data models.

Acknowledgements

The GRUMPS team gratefully acknowledges the funding provided by the UK's EPSRC (GR/N381141). We also thank Malcolm Atkinson, Richard Cooper, Murray Crease, Gregor Kennedy, Chris Mitchell, and the Level 1 Computing Science students at the University of Glasgow.

References

[1] L. Golab and M.T. Ozsu, M.T. (2003) "Issues in data stream management." *SIGMOD Record*, 32 (22), June 2003, pp. 5-14.

[2] P. Gray, I. Mcleod, S. Draper, M. Crease, R. Thomas, "A Distributed Usage Monitoring System". In Proc *CADUI (Computer Aided Design of User Interfaces)*. Kluwer, 2004. pp. 121-132.

[3] The GRUMPS Research Project. [Online]. Available: http://grumps.dcs.gla.ac.uk [16th June 2003].

[4] D. M. Hilbert, and D.F. Redmiles, "Extracting Usability Information from User Interface Events", *ACM Computing Surveys (CSUR) archive,* 32(4), 2000, pp. 384 - 421.

[5] G. E. Kennedy and T.S. Judd, "Iterative analysis and interpretation of audit trail data" In Crisp, G. & Thiele D. (eds) *Proc ASCILITE Conference*, Adelaide, Australia, 1, December 7-10, 2003, pp. 273-282,

[6] K. Renaud, P. Gray, "Making sense of low-level usage data to understand user activities", TR-2004-159, Department of Computing Science, University of Glasgow, 2004

[7] R. Thomas, G. Kennedy, S. Draper, R. Mancy, M. Crease, H. Evans, P. Gray, "Generic usage monitoring of programming students". In Crisp, G. & Thiele D. (eds) *Proc ASCILITE Conference*, Adelaide, Australia, 1, December 7-10, 2003, pp. 715-719.