Strongly Formative Pilot Studies on Constraints in Early Life-Cycle Work

Andrew Brooks
Department of Computer Science
University of Strathclyde
Glasgow G1 1XH
Scotland, UK
+44 (0)141 548 3583
andy@cs.strath.ac.uk

Shingo Takada
Department of Inf and Comp Sci
Keio University
Yokohama 223-8522, Japan
+81 45 560 1153
michigan@ics.keio.ac.jp

Louise Scott
JRCASE
Macquarie University
Sydney, NSW 2109
Australia
+61 (0)2 9850 9105
louise@mpce.mq.edu.au

ABSTRACT
In our work, two pilot studies radically changed the way in which data was to be gathered and interpreted i.e. the pilot studies were strongly rather than weakly formative. We report on these two pilot studies focussing on the key lessons learnt for empirical software engineering which include new ways of thinking about productivity and quality issues.

Keywords
CASE tools, Constraints, Pilot studies, Productivity, Quality

1 INTRODUCTION

In empirical software engineering, pilot studies typically result in relatively minor adjustments of a planned experimental design. In our work, two pilot studies radically changed the way in which data was to be gathered and interpreted i.e. the pilot studies were strongly rather than weakly formative. We report on these two pilot studies focussing on the key lessons learnt for empirical software engineering which include new ways of thinking about productivity and quality issues. A brief outline of the research project follows.

Methodological constraints are embodied in computerised tools to help guide activities like analysis and design. For example, a data-flow diagramming tool may automatically prevent direct links between data stores and a class diagramming tool may automatically prevent cyclical inheritance relationships. Too much enforced guidance, however, can hinder rather than help during creative problem solving. Earlier survey work [5] suggested that perceptions of high degrees of constraint in computerised tools coupled with unfavourable attitudes toward such constraint were associated with low user satisfaction and resistant behaviour. Subsequently, Day et al [6] have developed a research model linking individual differences, task characteristics, and constraint characteristics to attitudes toward and belief about constraints. The research model ultimately links these constructs to user productivity and product quality. Day et al in [6] outline the CADPRO (Constraints And the Decision PROcess) project which seeks to confirm such effects in a controlled setting. The ultimate goal is to specify how best to configure constraint environments in commercial CASE tools.

The original plan in the CADPRO project [6] was to have professional software developers undertake analysis/design tasks in 50 minute sessions in a usability laboratory using a CASE tool generated by CASEMaker, a meta-CASE tool [15]. CASEMaker is sufficiently flexible to allow configuration of the methodological constraint environment. Productivity and quality metrics were to be applied to artifacts created under different configurations of the constraint environment, thus allowing a test of the research model. Pilot studies were undertaken to help check that the planned experimentation would run smoothly and they are reported in the following two sections of the paper. Full details of the pilots are in [1,16].

---

1 This work was done while the author was at the Nara Institute of Science and Technology.

2 Now at Fraunhofer IESE, Sauerwiesen 6, D-67661, Kaiserslautern, Germany (scott@iese.fhg.de)
2 THE FIRST PILOT

2.1 Method

2.1.1 Introduction
The aim was to have subjects undertake analysis/design tasks under conditions similar to those intended for the CADPRO usability laboratory experiments in order to evaluate the proposed productivity and quality metrics. So, subjects worked alone, problem descriptions were no more than half a page, there was a 50 minute time limit, and pen and paper was disallowed. No attempt was made to reproduce the facilities of an actual usability laboratory.

2.1.2 Subjects
The authors themselves acted as subjects to get as much personal insight as possible. How authors’ experience levels with a methodology or tool impacted on the artifacts produced will become apparent later. Having the investigators as subjects inevitably means it is not possible to discount bias.

2.1.3 Tasks
Candidate analysis/design tasks typical of those used in higher education were considered. They were supplied by one of the authors and by Dr D Day (now at Towson University, USA). Two problem descriptions were chosen: a Mail Order System for data-flow diagram (DFD) modelling and an Automatic Map Labelling System for class modelling. All three investigators attempted these problems and produced solutions. This work was all undertaken on the same day at JRCASE. The Mail Order System problem was tackled first followed by a break to make notes before tackling the Automatic Map Labelling System problem. The investigators started at the same time and worked alone.

2.1.4 Tools
CASEMaker was still under development, so CASE tools currently available at JRCASE were chosen for convenience. For each problem, different tools were used to help provide an understanding of the impact of other tool issues (e.g. human-computer interface issues). The Toolkit for Conceptual Modelling¹ (TCM Version 1.1.0, Unix, May 1996) was used for DFD modelling and class modelling. A demonstration version of the MetaEdit² CASE tool (PC, 1993) was used for DFD modelling. Paradigm Plus³ (Unix, Version 3.5, 1996) was used for class modelling.

2.1.5 Self-reporting of experiences
After each 50-minute work period, the investigators met to make notes of their experiences. On the day following, further discussion took place to clarify the notes made.

2.2 Evaluation of metrics
Productivity and quality metrics for the CADPRO project were proposed in [9,11]. They were reviewed for their applicability to the artifacts produced in this first pilot study and where possible, metric values were calculated and reasons sought for any variations. The evaluation was partly judgmental. Comprehensive metric validation requires not only access to intrinsic quality factors (such as counts of errors that result in operational failures) but also that the metrics and quality factors be shown to be statistically associated [12].

Design modularity metrics were not calculated for solutions to the Mail Order System as they should be applied to structure charts and not DFDs. A data structure complexity metric was also proposed but the lack of data structure detail in the artifacts created precluded its consideration. Many of the proposed object-oriented metrics were unsuitable as the artifacts produced did not have the necessary detail.

2.2.1 Productivity Metrics
Count of nodes and edges
This metric involved counting all the nodes (e.g. processes/classes) and edges (e.g. data-flows/inheritances) in the artifacts produced and is a naive measure of productivity. For the class model, the metric (Class model 1) was extended to include counts of attributes and methods (Class model 2). The results are in Table 1 below.

<table>
<thead>
<tr>
<th>Invest.</th>
<th>DFD model</th>
<th>Class model 1</th>
<th>Class model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>48</td>
<td>19</td>
<td>41</td>
</tr>
<tr>
<td>S</td>
<td>16</td>
<td>22</td>
<td>43</td>
</tr>
<tr>
<td>A</td>
<td>39</td>
<td>12</td>
<td>56</td>
</tr>
</tbody>
</table>

There is a measure of variation in the results that could be exploited. There were, however, a variety of explanations involved as follows:

Investigator S had the lowest score on the DFD model due to a combination of a lack of experience with DFD modelling and problems becoming familiar with the MetaEdit user interface. Investigator L had the highest score on the DFD model due to having to repeat nodes to work around the lack of explicit tool support for levelling.

Investigator S had the highest score on class modelling (excluding attributes and methods) due to the inclusion of several classes at a low-level of detail: investigator S had

1 http://www.cs.vu.nl/~tcm/tcm.html
2 http://www.jsp.it/metacase
assumed that these were required. Investigator A had the highest score on class modelling (including attributes and methods) due to assumptions concerning the storing and retrieval of maps which resulted in many more attributes and methods being specified.

Does the metric tell us which investigator was the most productive? No.

The investigators agreed that such simple counting of elements must always be accompanied by a careful examination of the artifact produced if meaningful interpretations are to be made. Simple counting metrics unassociated with intrinsic quality factors have little or no direct interpretative power. But they should not be entirely dismissed. Product delivery measures profiled against time could help determine how software engineers went about problem solving and the effects of constraints on their behaviour. For example, software engineers who iterate toward a solution may delete as much as they create: so the delivery profile may have the shape of a jagged saw tooth.

2.2.2 Quality Metrics

Fan-in/Fan-out

Fan-in for a structure chart is the number of lines entering a component and is equivalent to the number of other components that call that component. A high fan-in suggests coupling is high. A high fan-out suggests that the complexity of the calling component is high [13]. Informational fan-in and fan-out can be likewise considered. An impediment in applying these metrics to the DFD models produced was how to count flows and composite flows without any data dictionary information. The investigators judged that the DFD models were not detailed enough to apply these metrics. The class models were likewise not big or detailed enough. Inheritance hierarchies that were produced were minimal and message passing was far from complete. The investigators decided not to calculate values for these metrics.

Inheritance Usage and Depth (O-O only)

Survey data has revealed that many object-oriented practitioners believe that difficulties in understanding object-oriented software occur when inheritance hierarchies are too deep [4]. Laboratory data has indicated that performance can deteriorate if it is not obvious which class should be specialised from and if tracing through the hierarchies is required for a sound comprehension [3]. Ease of maintenance is a recognised quality indicator: a simple quality metric could be the maximum depth of any inheritance tree. The investigators, however, used very little inheritance: the maximum depth of inheritance used was 1. The depth metric is clearly inappropriate for such small artifacts.

What is of possible interest, however, is whether or not there should be inheritance, and whether the correct balance is achieved between superclass and subclass. The investigators who had included inheritance, however, disagreed with one another on the appropriateness of the use of inheritance in the other's solution, and the extent of abstraction in the superclass to facilitate other subclass possibilities in the future. Clearly it would be impractical to establish a metric when there is much to disagree about with early life-cycle artifacts.

Cohesion and coupling (O-O only)

High cohesion and low coupling should suggest adaptability of software [13]. Metrics for cohesion and coupling have been proposed by Chidamber and Kemerer [14], but detailed information, not present in the artifacts produced, is required for their computation. Thus, as a measure of cohesion, we counted the maximum number of methods in any one class: an object with too many methods may be trying to do too much. As a measure of coupling, we chose to count the maximum number of inter-class relations between any two classes. The results are in Table 2 below.

<table>
<thead>
<tr>
<th>Invest.</th>
<th>Maximum number of methods</th>
<th>Maximum number of inter-class relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>S</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

There is a measure of variation in the results that could be exploited. Do the metrics measure cohesion and coupling? No. Investigator A had the highest score for cohesion but this was due to assumptions concerning the storing and retrieval of maps, which resulted in a few more methods being specified in a controller class. Investigator A had the low score for inter-class relations because of lack of modelling experience in this area.

The investigators agreed that these metrics really needed to be applied to larger, more complete, artifacts, if they were to make any sense.

Number of loops in a class diagram (O-O only)

There was variation present that could be exploited and loops in class diagrams may indicate a lack of quality: investigators A and L had no loops whilst investigator S had 3 loops. Two of the loops were formed simply from mutual "uses" relationships and were not judged to be indicators of quality or productivity. These two loops had arisen because the investigator had chosen to insert more detail in part of the class model. Investigator L suggested that the other loop could reflect the quality of the solution.
as it may imply redundancy of relationships. Does the metric reliably measure quality? No.

2.3 Summary and recommendations
The first pilot established that a traditional metric approach to productivity and quality measurement of early life-cycle artifacts would be inappropriate for the proposed usability laboratory experiments.

Investigators making different assumptions, largely as a result of uncertain problem boundaries brought about by using short problem descriptions, caused variation in the values obtained for metrics. To a lesser extent, different experience levels with a particular modelling technique or tool also caused variation. The lack of explicit support of levelling in TCM was another explanation for unsatisfactory DFD models. User-interface concerns also impinged more on the work of the investigators than one might have expected. (This was the first hint that interface constraints could be as significant as methodological constraints in shaping end-user behaviour of CASE tools.)

Various recommendations were made for the second pilot, including: To reduce unwanted variability, problem descriptions should be more detailed, more time should be allowed, completion times should be noted, user-interfaces should be of a high quality, and subjects should preferably be experienced with the modelling technique and tool. Quality can be assessed by reference to a sample solution (with checklists) or by a panel of experts. Product delivery measures should be profiled against time — this last recommendation turned out to be the most important.

3 THE SECOND PILOT

3.1 Method
3.1.1 Introduction
Subjects undertook analysis/design tasks modified to take account of concerns from the first pilot. A main objective was to record and interpret delivery profiles. Subjects worked in an actual usability laboratory, part of the software engineering laboratory at the Nara Institute of Science and Technology. Video records were made of the computer display that included audio records of subjects’ utterances. Subjects worked alone, problem descriptions were no more than a page, there was a 2 hour time limit, and use of pen and paper was disallowed. As the CASEMaker tool was now developed, subjects for the DFD task used a DFD tool generated by CASEMaker. As recommended from the first pilot: sample solutions were used as reference points for evaluation, completion times were noted, and profiles of product delivery against time provided by use of CASEMaker were analysed. The DFD tool generated by CASEMaker did not enforce methodological constraints but did record methodological constraint violations.

3.1.2 Subjects
Three subjects (henceforth, Subjects A, B, and C), all from the staff at the Nara Institute of Science and Technology, took part. Their experience levels varied and we took this into account in the analyses. Subject B had the most experience of DFD modelling. All three subjects had little experience of class modelling but Subject C did have extensive object-oriented programming experience.

3.1.3 Tasks
Subjects were required to perform DFD modelling for a Mail Order System and class modelling for an Automatic Map Labelling System. The problem descriptions were modified to take account of concerns from the first pilot. The problem descriptions and instructions were given both in English and Japanese. Care was taken to try and avoid the problems that occurred in the first pilot when subjects made unexpected assumptions while solving the problems. But at the end of the first day of this pilot, it was felt necessary to make further small changes to the instructions and problem descriptions e.g. to make clear that class create and destroy operations should not be modelled and to clarify a singular-plural ambiguity in Japanese. Subjects were given a maximum of two hours to solve the problem (a recommendation from the first pilot).

3.1.4 Modelling tools
The Toolkit for Conceptual Modelling (TCM Version 1.6.6) was used for class modelling while CASEMaker was used for DFD modelling. Subjects were given 10 minutes to familiarise themselves with their assigned tool following a brief introduction.

3.1.5 Use of think-alouds
Subject A was instructed to think-aloud, while the other subjects were simply encouraged to verbalise any comments they wished to make about the tool etc.

3.1.6 Subject debriefings
Investigators monitored (video and audio) what took place from the control room in the usability laboratory. Lists of questions were drawn up for subject debriefings that took place immediately after subjects had finished. Sample solutions to the tasks were used as an aid during debriefing.

3.1.7 Unfulfilled recommendations from the first pilot
Due to time constraints, some of the specific recommendations from the first pilot were not acted upon. Also, two specific recommendations were only partially fulfilled: although sample solutions were provided, checklists of good and bad solution characteristics were not, and CASEMaker had not been extensively prototyped.

4 http://wwwis.cs.utwente.nl:8080/~tc/m/index.html

617
3.2 Timing results

<table>
<thead>
<tr>
<th>Table 3. Time to produce a model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject A</td>
</tr>
<tr>
<td>Subject B</td>
</tr>
<tr>
<td>Subject C</td>
</tr>
</tbody>
</table>

Table 3 shows the time each subject took to produce a model (in hours and minutes). Subject B basically finished the DFD model after about 1:08 but then he checked the diagram for the next 44 minutes making some changes and moving nodes and edges. We partly attribute this behaviour to the subject’s greater experience. All three subjects were more or less finished their basic DFD models in about 1 hour. For the class model, only subject A came close to the allotted maximum 2 hours. This subject continually made changes to his model for much of the time. Subject C was probably the quickest because of his extensive object-oriented programming experience.

3.3 Solution Characteristics of the Models

Ideas about the quality of DFDs and class diagrams are, in part, derived from [8,10].

3.3.1 Solution Characteristics of the DFD models

Data stores and multiple processes in the context diagram: Subjects A and C did not have much experience with DFDs. Both subjects violated the constraint for context diagrams that there can be only one process. So poor quality is attributable to lack of experience. Interestingly, Subject A originally had three external entities; but later changed two of them to processes as they seemed to be more ‘internal’ than ‘external’. These changes need not be viewed as resulting in a poorer quality of solution: determining system boundaries is not always easy, especially in the absence of a context where analysts are free to ask questions.

Missing customer inquiry data-flow: Subjects A and C did not model customer inquiries due to their not appreciating the role of a customer inquiry. Poor quality here was due to lack of domain experience.

No outgoing or incoming data-flows from data stores: Subjects A and B both had a data store (“Billing Info”) which had no outgoing data-flow. In the sample solution there is only a single outgoing flow to help with answering a customer inquiry; since subject A was unclear about what this was, poor quality here can again be attributed to a lack of domain experience. Subject B was not sure of the purpose of the “Billing Info” data store. He thought that “Customer Info” contained all possible information, including billing information, and so constructed a model in which many types of information were stored in the “Customer Info” data store. This was due to the Japanese translation where “customer mailing information” was translated as “information concerning the customers’ addresses, etc”. So poor quality here is attributable to problems in translation. Subject B also had one data store (“Inventory”) which had no incoming data-flow. He thought that the incoming data-flow would be handled by an external system. Again, this boundary issue would be easily resolved in context when analysts can ask questions. Given this subject’s reasoning to regard his solution to be of poor quality with respect to the missing data-flow may not be appropriate.

Overloading of data-flows: Module interfaces should be narrow not broad. Subject B, however, overloaded a number of data-flows with customer information. (This was not explicitly shown on his diagram.) This overloading of flows is indicative of a design/implementation decision. Subject B thought that separately sending “Customer Info” to various processes was redundant. From an analysis perspective, this overloading is poor not only because the flows are doing too much but also because some data is ‘tramped’ (or unnecessarily directed) through a process unused. So lack of quality here has arisen because the subject considered issues of relevance later in the life cycle.

Inconsistencies: In Subject B’s solution, there is a combined “Order/Cancellation Info” data-flow between “Handle order” and “Customer info”, yet these flows are separately indicated between “Business Customer” and “Dispatch Input”. From Subject B’s viewpoint, however, this was not viewed as being inconsistent as he was handling cancellation as one particular form of order. His motivation was to make the diagram not have any extra flows which would all need to be checked if some changes were made. Lack of quality has arisen because the subject considered issues of relevance later in the life cycle.

Additional Data Stores: Subject B introduced an additional data store “Inventory” which represented the physical store and which was managed by another system, the current system simply making withdrawals from this “Inventory”. Mixing logical and physical models could be considered as poor practice but real designers have been found to not comply with the rules of structured analysis [2].

3.3.2 Solution Characteristics of the Class Models

Subclassing: Too much subclassing in class models can be difficult to understand and manage. But too little subclassing reduces the benefits associated with an object-oriented approach. Inappropriate subclassing, where conceptual integrity is of poor, can mean difficulties in understanding [7]. All three subjects had a class for a map feature but only Subject C modelled the “circle” and “rectangular” features as subclasses. The other two subjects made use of an attribute as a discriminator between “circle”
and "rectangular" features. Given the early stage of analysis, it is premature to regard use of a simple attribute as a discriminator as poor. Subject C had an "optimisation" class to handle the actual placing of map labels. He had subclasses from that class for each level of optimisation, and reasoned that if a new optimisation level was necessary, it only needed to be "plugged-in". This subclassing seems excessive, especially since the differences between optimisation levels are likely to be concerned only with the sizes of search spaces and the resolutions employed i.e. these differences could be modelled using attribute(s). But again, given the early stage of analysis, optimisation subclassing cannot be categorically stated as poor. Subject B had two additional inheritance relationships: "Labeled Map" derived from "Map" and "Text Label" derived from "Label". This subject interpreted that labels could also be placed on maps independently of any features and had assumed there would be more than one type of label. This latter assumption may have arisen because the problem description mentions 'default labels' and 'default text label'. From this viewpoint, the additional subclassing cannot be categorically stated as poor.

Multiplicity (Cardinality): Subject A had a one-to-one relationship between the "MAP making" class and the "MAP" class when it should have been a one-to-many relationship. This was due to an ambiguity in the Japanese version of the Class1 problem description: "map" in the Japanese version could be interpreted to be either singular or plural. So poor quality here is attributable to misunderstandings caused by the translation process. Subject B had a one-to-many relationship between "Labeled Map" and "Feature" which allowed for such maps to have no features. Having maps with no features may be viewed as an error, but as noted above, Subject B interpreted the problem description as suggesting that labels could also be placed on maps independently of any features. From this viewpoint, the stated multiplicity cannot be viewed as incorrect.

Design and implementation concerns: Subject C introduced a "coordinate" class and he also specified method overriding in a few places in his class model i.e. design and implementation concerns influenced his solution. But it cannot be categorically stated that the influence has been detrimental. The object-oriented approach is relatively seamless and that this is considered one of its strengths: it might be naïve to expect analysis-only models given the ease with which design and implementation concerns can be introduced. The instructions given to subjects stated they should not model low level classes such as those used to represent strings or numbers. This instruction, in retrospect, was insufficient to rule out the modelling of classes like a "coordinate" class, which would be modelled if a grammatical analysis approach is applied to the problem description. (Also note that Subject C had extensive object-oriented programming experience.)

Missing system class: Only Subject A had classes for the system itself and a map. In an operational context, such a class would not be missing for long.

3.4 Delivery Profiles for DFD Models
CASEMaker recorded time-stamped delivery profiles for the three DFD models produced. Creations and deletions of nodes (entities and processes) and edges (data-flows) were recorded. Renaming actions were also recorded but these were incomplete. Delivery counts were slightly inaccurate because of phantom edge creations. The delivery profiles were visually examined for the presence of dips that may represent productivity loss. The inaccuracies in delivery profiles did not effect the detection and interpretation of dips. Each subject, however, suffered one major crash of the CASEMaker DFD tool.

3.4.1 Subject A
Figure 1 is Subject A's pre-crash delivery profile that shows how the number of model nodes and edges (y-axis) developed with time (x-axis).

![Figure 1. A's Delivery Profile (pre-crash)](image-url)

Figure 1 has two major dips, the first caused by the subject deleting an external entity which automatically caused the deletion of several connected edges. The video record revealed that the subject had wanted to change the external entity into a process, an action supportable by CASEMaker but which had been disabled because of the way the interface to this action operated. (Some CASE tools support the dynamic re-typing of diagram objects: such features aid productivity.) Retrospectively, the subject can be seen to have met an interface constraint that reduced his productivity: he was required to redraw several edges.

The second major dip had two causes. Firstly, the subject had accidentally pasted an extra copy of an external entity onto his Level 0 diagram. He chose to cut it out without realizing that this removed all instances of the entity from the Level 0 diagram. Retrospectively, the subject can be seen to have met an interface constraint (there was no action supporting the erasure of a single instance of a
duplicated diagram object) which reduced his productivity in that he had now to re-create the external entity. Secondly, in attempting to recreate the external entity on the Level 0 diagram using cut and paste from the Context Diagram, two copies of the entity appeared on the Level 0 diagram. The subject chose to delete one of the extra copies, rather than cut it, and the effect of this was that all instances of the external entity were removed including the original on the Context Diagram. So, again productivity was reduced because the subject had now to re-create the external entity. That there was no separate interface action to remove just an instance of a duplicated diagram object again caused difficulties.

![Figure 2. A's Delivery Profile (post-crash)](image)

On the post-crash delivery profile (Figure 2) there was one major dip. The video record showed that this was due to the subject having accidentally performed additional create actions due to delays in display updating.

So for Subject A, interface constraints were responsible for productivity loss.

3.4.2 Subject B
The pre-crash delivery profile showed no unusual activity. The post-crash delivery profile (Figure 3) proved to have a number of noteworthy features.

The video record was examined from about 1600 seconds onwards. This was the start of a checking and revising period as noted earlier. The first major dip was caused by the subject deciding to route customer information through a dispatcher process, and so independent accesses to the customer information data-store were deleted. The second major dip involved typical modeling activity but included one deletion as a result of the subject deciding to treat a cancellation as just another order i.e. an independent flow about cancellation was deleted. The subject overloaded the dispatcher process and order information flows. The video record also revealed that the subject spent several minutes reorganizing the position of diagram elements where the delivery profile is flat as part of checking his model.

![Figure 3. B's Delivery Profile (post-crash)](image)

Subject B’s behaviour illustrates the complex relationship between productivity and quality. In Section 3.3.1, Subject’s B final model was regarded as lacking quality because he had revised the model to take into account design/implementation considerations. In one respect, there has been loss of productivity by having spent another 44 minutes working on the model. But in other situations, revising work could lead to improved quality.

3.4.3 Subject C
Subject C’s pre-crash data was unfortunately lost. The post-crash delivery profile showed nothing unusual.

3.5 Summary and Recommendations
Various influences contributed to a lessening of quality: lack of experience with the modelling technique and the domain, translation problems, considering issues of relevance later in the life cycle, and a lack of a context where analysts are free to ask questions.

Several issues about quality, however, could not be resolved given the early stage of analysis. Moreover, without video and audio records and subject debriefings, incorrect interpretations might have been drawn concerning the quality of subjects’ solutions. It is very difficult to constrain problem descriptions in such a way as to prevent subjects’ making unexpected assumptions and to constrain subjects into considering purely the analysis phase.

A number of dips in delivery profiles were interpretable as lost time due to interface constraints present in the DFD tool. The influence of interface constraints cannot be ignored. Without video and audio records and subject debriefings, incorrect interpretations might have been drawn with respect to these dips. The relationship between productivity and quality is a complex one: dips in delivery profiles may reflect productivity loss or quality gain.

Various recommendations for the next phase of CADPRO experimentation [16] include a suggestion to supplement
delivery profiles with profiles which show the activity of simply moving nodes and edges about the screen.

4 CONCLUSIONS

Our two pilots proved to be strongly formative. Key lessons have been learnt for those interested in studying productivity and quality issues in early life-cycle work:

A traditional metric approach to productivity and quality issues is inappropriate for small, early life-cycle artifacts. Preventing experimental subjects from making unexpected assumptions, we believe, is impossible, for early-life cycle problems that have to be undertaken in conditions of isolation and accomplished in 1 or 2 hours. It is difficult preventing subjects from considering design and implementation issues in early life-cycle work. Delivery profiles hold great promise for determining the scale of productivity loss suffered by an individual when a constraint takes effect (interface or methodological). It may be possible to use metrics based on the proportion of time lost by an individual. But careful interpretation is needed to determine what the effects on model quality have been.

ACKNOWLEDGEMENTS

This work was supported by: an ARC Large Grant to UNSW (CAESAR, Centre of Advanced Empirical Software Research), an ARC Small Grant to Macquarie University (JRCA), a DEETYA TIL Grant to Macquarie University in conjunction with UNSW (JRCA and CAESAR), JRCA Centre funding, CAESAR Centre funding; the Nara Institute of Science and Technology, Macquarie University, and the University of Strathclyde.

REFERENCES


