Evaluating the application of software metrics to data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #1)

Andy Brooks

Louise Scott

Shingo Takada

Nara Institute of Science and Technology

Information Science Technical Report

NAIST-IS-TR98010

This report has also been issued as a technical report by the CSIRO/Macquarie University Joint Research Centre for Advanced Systems Engineering (Technical Report # R.R. 98/14)

Evaluating the application of software metrics to data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #1)

ANDY BROOKS Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, Scotland, UK (andy@cs.strath.ac.uk)

LOUISE SCOTT Joint Research Centre for Advanced Systems Engineering (JRCASE), School of MPCE, Macquarie University, Sydney, NSW 2109, Australia (louise@mpce.mq.edu.au)

SHINGO TAKADA Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma-shi, Nara-ken, 630-0101, Japan (michigan@is.aist-nara.ac.jp)

Abstract

This paper reports an evaluation of a number of metrics proposed to measure user productivity and product quality in a usability laboratory setting. The examined metrics were found to be unsuitable indicators of productivity and quality in the context of the experimental study. Many insights were obtained, however, into what will be required to make future experimentation successful. A series of recommendations are made.

1. Introduction

Methodological constraints are often 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 the software engineer during creative problem solving. Earlier survey work [1,2] 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 [3] 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 [3] outline the CADPRO (Constraints And the Decision PROcess) project which seeks to confirm such effects in a controlled experimental setting. The ultimate goal of the project is to specify how best to configure constraint environments in commercial CASE tools. The original plan in the CADPRO project [3] 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 [11]. 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. A principle concern was: Would 50 minutes be enough time to allow subjects to produce artifacts that metrics could be meaningfully applied to? The purpose of this paper, therefore, is to report an experimental evaluation of a number of metrics proposed to measure the final outcomes (user productivity and product quality) in the CADPRO project [4,5] i.e. to report on the first pilot study.

The evaluation involved subjects undertaking analysis/design tasks under conditions similar to those intended in the planned usability laboratory experiments. Various metrics were then applied to the artifacts produced to see if the values obtained genuinely measured productivity and quality. This evaluation of the suitability of the metrics was necessarily judgmental. Comprehensive metric validation requires not only access to intrinsic quality factors (such as counts of the errors in software that result in operational failures) but also that the proposed metrics and quality factors be shown to be statistically associated [6].

Some of the metrics were found to be directly unsuitable in the context of this experimental study because they lacked detailed rationale as to how they should calculated for the artifacts produced. Other metrics could not be meaningfully calculated because the artifacts produced were small and lacked the detail necessary for how they should be calculated. Yet other metrics were applicable only to artifacts produced later in the software life cycle.

Those metrics that could be calculated were found to be unsuitable indicators of user productivity and product quality in the context of this experimental study. Variation in the values obtained for these metrics was caused by investigators making different assumptions about what was needed to solve the problem. This was largely as a result of poorly specified problem boundaries brought about by using short problem descriptions. To a lesser extent, variation was also caused by the different experience levels of the investigators with a particular modelling technique or tool.

The exercise of producing analysis/design artifacts under conditions similar to those intended for CADPRO usability laboratory experimentation did, however, produce many insights into what will be required to make future experimentation successful. A number of recommendations are given. In particular, one or more additional pilot studies should be carried out to iterate problem descriptions and examine profiles of product delivery and constraint activations against time.

2. Method

2.1 Introduction

This first pilot study involved subjects undertaking analysis/design tasks under conditions similar to those intended for the CADPRO usability laboratory experimentation. Subjects worked alone, problem descriptions were no more than half a page, there was a time limit of 50 minutes, and subjects did not use pen and paper to help them solve the problems. Various decisions had to be made concerning choice of subjects and tasks. The tools to be used had also to be decided upon as CASEMaker was still under development. No attempt at this stage was made to reproduce the facilities offered by an actual usability laboratory: no video or other protocols were recorded.

2.2 Subjects

It was decided that the authors themselves would be the subjects for this first pilot. It was felt important to get as much personal insight as possible and there was no better way than to have the investigators themselves tackle the tasks within the proposed 50-minute time limit. How subjects' 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: during task selection (see next section), subjects were exposed to the problem descriptions prior to the 50-minute work period allocated. Only the reader can judge whether the lessons learnt from this experimental study were unduly compromised.

2.3 Tasks

A set of candidate analysis/design tasks was considered. The problems were typical of those used in systems analysis and design classes given in the tertiary education sector and were supplied by one of the authors (Dr A Brooks) and by Dr D Day, both of whom are experienced in delivering such classes. Some of the problem descriptions were regarded to be too lengthy for a 50 minute work period i.e. had they been used, a considerable proportion of available time would have been spent simply reading the problem description. Some of the problem descriptions were viewed as overly specific i.e. had they been used, subjects would have essentially been capturing the solution presented rather than solving a problem. Another consideration was how interesting the problem descriptions were: we were concerned about using problem descriptions that professionals might simply find boring. Two problem descriptions were eventually chosen: their descriptions are reproduced in Appendix A. It was decided to perform data-flow diagram (DFD) modelling for the Mail Order System and class modelling for the MAPLEX system. All three investigators attempted these problems and produced corresponding DFD models and class models. These models are shown in Appendix B. This work was all undertaken on the same day at the same location (JRCASE). The investigators tackled the problems at available machines within this research laboratory. The Mail Order System problem was tackled first followed by a break to make notes before tackling the MAPLEX problem. The investigators started work on the problems at the same time.

2.4 Tools

As stated earlier, CASEMaker was still under development, so other tools had to be considered. CASE tools that were currently available at JRCASE were chosen for the simple reason that they were familiar to two of the investigators and were known to be immediately operable. Furthermore, it was decided that, for each problem, different tools were to be used to help provide an understanding of the impact of tool issues (e.g. human-computer interface issues) on later experimentation. The Toolkit for Conceptual Modelling [1](TCM Version 1.1.0, Unix, May 1996) was used for DFD modelling and class modelling. A demonstration version of the MetaEdit[2] CASE tool (PC, 1993) was used for DFD modelling. Paradigm Plus [3](Unix, Version 3.5, 1996) was used for class modelling.

2.5 Self-reporting of experiences

At the conclusion of each 50-minute work period, the investigators met to relate and make notes of their experiences. On the day following the experiment, further discussion took place to clarify the notes made.

2.6 Metric calculation

Also on the day following the experiment, the investigators met to discuss the proposed metrics. Where metrics were calculated, the investigators analysed the artifacts created to determine if the metrics genuinely reflected user productivity and product quality.

3. Discussion of artifacts produced and self-reports.

Small artifacts

All three investigators recognised that the artifacts they had produced were small (no more than 10 bubbles or classes).

Incomplete artifacts

All three investigators recognised that both their DFD models and class models were incomplete, more so for the DFD models than the class models. Investigator L also knew that the DFD model was inconsistent (especially with respect to levelling), but didn't have time to fix the inconsistencies.

All three investigators believed their class models to be more complete, certainly with respect to class identification. The investigators, however, believed much more could be produced if there was a strong requirement to specify class relationships and message passing between classes.

Iteration

Investigator L was not unhappy with the way the DFD model was developing. Investigator S was unsure about the way the DFD model was developing but this may have been due to lack of experience with the methodology. Investigator A had made the decision to start over on the DFD model and had just begun to

[1] http://www.cs.vu.nl/~tcm/tcm.html
[2] http://www.jsp.fi/metacase
[3] http://www.platinum.com/products/appdev/pplus_ps.htm

draw a context diagram when a halt was called at 50 minutes. Investigator A specifically regarded his first attempt as a first cut that needed to be iterated. (This investigator also believed that the lack of support for levelling in the version of TCM used contributed to the need to iterate the first attempt.) So a time limit of 50 minutes did not allow the natural process of iteration to improve the solution.

The need to iterate further with the class models was not explicitly expressed by any of the investigators.

Assumptions

Both the Mail Order System and MAPLEX problem descriptions were incomplete in the sense that the investigators were required to make a variety of assumptions about what they should and should not do. The assumptions that arose from this study are detailed in Appendix C. It is likely further examination will yield additional assumptions requiring consideration.

It is very difficult to specify problem boundaries with short written descriptions. For DFD modelling, one solution might be to actually provide the context diagram. For both problems, it may be worth considering a maintenance activity so that the context is fully specified and assumptions are kept to a minimum.

The time limit also had an effect on how one investigator decided on the problem boundaries. Aware of the time constraint, investigator S assumed a couple of sentences in the Mail Order System problem description could be ignored.

Solution characteristics of the DFD models

Investigator L worked around the lack of specific notation to handle physical material flow in TCM by simply writing "(product)" on relevant flows.

Investigator A, realising that levelling was not directly supported in the tool used, attempted to draw a complete diagram, but became dissatisfied with the solution developing. Attempts at drawing a context diagram were cut short because the time limit of 50 minutes was reached. This investigator had also chosen to distinguish between magazine subscribers and non-subscribers believing that these two sets of customers would be treated differently. But this distinction was to be removed when an attempt was made to draw a context diagram. Investigator L worked around the lack of support for levelling but found it difficult and time-consuming.

Solutions for investigators S and A had no mention of the role of the regional offices in the final diagram. Investigator A said this role had caused confusion. If these offices were external entities then direct flows between these external entities to the external entities representing customers are prohibited by the norms of the methodology. Investigator A had also misinterpreted the problem description believing that only information flowed to the regional offices and not the products. Shortly before the 50-minute time limit was reached, investigator A had decided to make the regional offices a process.

Investigator S had a regional office (and Customer as externals) part way through. Due to difficulties with levelling the model he decided to make just a one level diagram. He deleted the Regional Office external entity then forgot about it until investigator A mentioned it in the subsequent discussion. Problems with misinterpretations and forgetfulness do usually get dealt with in the daily routine: that they were determinants of behaviour here suggests that it may be inappropriate to measure user productivity and product quality over such a short space of time.

Solution characteristics of the class models

Through lack of experience with notation, investigator A had used incorrect notation to specify the aggregation relation. This investigator had also failed to specify the (1:1?) relationship between features and labels.

Investigator S had outlined separate classes for low-level label details (colour, number, etc). This is indicative of a boundary problem. Everything in object-oriented analysis and design can be made a class, but to what extent should the designer be concerned about lower level details for which there are likely to be existing classes/class libraries is an open question. This investigator had a loop of class relationships and it was agreed that such loops required careful examination: loops may imply redundancy and poor quality of solution.

Despite investigator S being more conscious of inheritance relationships (due to the possibility of using depth of inheritance as a metric) he couldn't find any suitable relationships in the MAPLEX case to model with inheritance. Investigator L had two inheritance relationships (depth 1) and investigator A had one inheritance relationship (depth 1).

Unlike the other investigators, Investigator L had provided visibility information for class attributes and methods. Paradigm Plus had provided default visibilities on creation and Investigator L had made some edits to make some public methods private.

The investigators differed on their approach to dealing with the "optimisation level". There is a strong encapsulation argument for associating the level with the map, but what if the level is never examined after the processed map is saved? Being unable to decide, Investigator S deliberately chose to compromise and stored the level information in two places. Such redundancy could be considered as an indicator of poor quality of solution, a solution with unresolved issues.

All investigators had an unspecified user interface class to deal with input/output to the user i.e. they had all assumed that developing a model of the GUI was not part of the exercise.

Investigator L did not feel she could adequately capture the system using a class diagram only and drew a state diagram for the map class as well.

The role of constraints

There was general agreement that constraints had played less of a part in doing the class modelling exercise. It was speculated that given the iterative, prototyping philosophy underlying object-oriented software development, a highly constrained environment would be extremely artificial and inappropriate for such an exercise.

The lack of explicit levelling support (in effect, an interface constraint) was the strongest influence on behaviour and quality of DFD models built using TCM.

The role of the human-computer interface

Interface problems impinged on the investigators' work. Appendix D lists the interface problems encountered. Retrospectively, use should have been made of the most modern versions of MetaEdit/TCM/ Paradigm Plus or equivalents. It has at least been demonstrated that interface neutrality is extremely important if behavioural analysis is to be relatively uncomplicated.

4. Evaluation of Metrics

Productivity and quality metrics for the CADPRO project were proposed in [4] and [5]. In this section they are reviewed for their applicability to the artifacts produced in this pilot study and where possible, metric values are calculated and reasons sought for any variations.

4.1 Productivity Metrics

Two metrics were selected as productivity metrics in [5]: McCabe Cyclomatic Complexity and a simple count of nodes and edges. It is unclear why McCabe's metric was selected as a productivity metric as it is usually considered simply as an indicator of the amount of testing required.

McCabe Cyclomatic Complexity

The investigators decided it was not possible to make any meaningful calculations for this metric: they would have had to make a series of unsubstantiated assumptions to perform any calculation. For example: How should edges be counted when there are multiple edges between the same nodes - as one edge or as many? Do external entities count as nodes? Do data-stores count as nodes? A closed loop in a control graph signifies two independent paths: should a diagrammatic loop brought about by read/write flows to a data-store really be considered a loop? It is difficult to reconcile the concept of direction in cyclomatic complexity with DFD models.

The McCabe Cyclomatic Complexity metric has similar difficulties when applied to class models.

To apply graph complexity measures, a more detailed rationale is required.

Count of nodes and edges

This metric involves counting all the nodes and edges in the artifacts produced and is a naive measure of productivity. For the class model, the metric was extended to include counts of attributes and methods. Two investigators did the counting and there was no disagreement between them. The results are as follows:

Investigator	DFD model	Class model (excluding attributes and methods)	Class model (including attributes and methods)
L	48	19	41
S	16	22	43
A	39	12	56

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

1. Investigator S had the lowest score on the DFD model. This was considered due to combination of a lack of experience with DFD modelling and problems becoming familiar with the MetaEdit user interface.
2. Investigator L had the highest score on the DFD model. This was due to having to repeat many nodes to work around the lack of explicit tool support for levelling.
3. Investigator S had the highest score on class modelling (excluding attributes and methods). This was due to the inclusion of several classes at a low-level of detail. Investigator S had effectively assumed that these were required.
4. Investigator A had the highest score on class modelling (including attributes and methods). This was 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? The answer is no. On the DFD model, investigator L created more nodes and edges, but this did not mean that a more finished artifact had been produced. On the class model (including attributes and methods), investigator A had created the most, but it is simply not possible to say that the resultant artifact is more complete, because additional effort was undertaken as a result of assumptions being made about what was required. Also notice how investigator A jumps from being the least to the most 'productive' on the class model when attributes and methods are counted in.

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 and constraint activations can help determine how software engineers went about problem solving and the effects of constraint activations on their behaviour. For example, software engineers who greatly preferred to try things out and iterate toward a solution may delete as much as they create: so the delivery profile may have the shape of a jagged sawtooth as the number of nodes and edges rises and falls with each iteration.

4.2 Quality Metrics

Several metrics were proposed as quality metrics in [4]. Information flow and system complexity metrics were proposed based on the notions of fan-in and fan-out. Design modularity metrics were proposed based on the notions of coupling, cohesion, and span of control. These design modularity metrics were not calculated for solutions to the Mail Order System as they should be applied to structure charts and not data-flow diagrams. A data structure complexity metric was also proposed but the lack of data structure detail in the artifacts created also precluded any consideration of this metric. In addition, over twenty object-oriented metrics were listed in the appendix to [4]. Many of these object-oriented metrics were unsuitable as the artifacts produced did not have the necessary detail for their calculation. Also, we were aware of only one object-oriented metric whose utility was supported by empirical studies (depth of inheritance). Focussing on metrics that seemed the most promising, the investigators reviewed: fan-in/fan-out, inheritance-depth, cohesion and coupling, and number of loops in a class diagram.

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 [9]. 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 in the absence of data dictionary information. The investigators judged that the DFD models were simply 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 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 [7]. Laboratory data has indicated that performance can deteriorate when maintainers are asked to extend from a depth of 5 levels if it is not obvious which class should be specialised from and if tracing through the hierarchies is required for a sound comprehension [8]. Ease of maintenance is a recognised quality indicator: a simple quality metric, therefore, is the maximum depth of any inheritance tree. The investigators, however, used very little inheritance. The results of the calculation of these metrics are as follows:

Investigator	Number of times inheritance used	Maximum depth of inheritance
A	1	1
L	2	1
S	0	0

Does the depth metric tell us which investigator produced the best solution? The answer is no. The depth metric is clearly inappropriate for such small artifacts: the maximum depth of inheritance used is well below levels at which maintenance difficulties are known to occur and there is not enough variation.

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 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 that may be needed in future. A rationale could possibly be established for a specific problem for deciding if use of inheritance was appropriate and if the extent of abstraction was appropriate.

Cohesion and coupling (O-O only)

High cohesion and low coupling should suggest adaptability of software [9]. If a change is needed then it is likely to impact on only one module. On the other hand, if modules perform a variety of tasks, and considerable inter-module communication is required, then the likelihood is that even quite a small change will require a considerable effort by maintenance personnel to understand and work with many modules.

Metrics for cohesion and coupling have been proposed by Chidamber and Kemerer [10], but detailed information, not present in the artifacts produced, is required for their computation. Thus, as a measure of cohesion, we chose to count the maximum number of methods in any one class, the rationale being that an object with too many methods may be trying to do too much and lack cohesion as a result. As a measure of coupling, we chose to count the maximum number of inter-class relations between any two classes, the rationale being that the greater the number of relations the greater would be inter-object communication. The results are:

Investigator	Maximum number of methods	Maximum number of inter-class relations
A	8	1
L	5	3
S	3	3

There is a measure of variation in the results that could be exploited. Do the metrics measure cohesion and coupling? The answer is 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 object. 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. Other concerns include (a) Should get/set methods for each attribute be counted? Investigator A had included some of these methods, but should such standard access methods be part of user productivity and product quality measures for creative problem solving work? (b) The solutions were incomplete and hence, the values are underestimates.

(Note that according to Briand et al [12], very few object-oriented cohesion metrics have been empirically validated and that where such validations exist, they are sometimes seriously flawed.)

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

When reviewing the class models produced, the investigators agreed it was worth considering a count of the number of loops present in the class diagrams. There was variation present that could be exploited and loops in class diagrams may indicate a lack of quality. The results are:

Investigator	Number of loops in class diagram

A	0
L	0
S	3

The solution by investigator S was the only solution to register non-zero for this measure. On inspection, 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 simply arisen because the investigator had chosen to insert more detail in one 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. It was agreed that this should be explored further. Does the metric measure quality? Perhaps.

5. Summary and Recommendations

Some of the metrics were found to be directly unsuitable in the context of this experimental study because they lacked detailed rationale as to how they should calculated for the artifacts produced. Other metrics could not be meaningfully calculated because the artifacts produced were small and lacked the detail necessary for how they should be calculated. Yet other metrics were applicable only to artifacts produced later in the software life cycle.

Those metrics that could be calculated were found to be unsuitable indicators of user productivity and product quality in the context of this experimental study. Variation in the values obtained for these metrics was caused by investigators making different assumptions about what was needed to solve the problem. This was largely as a result of poorly specified problem boundaries brought about by using short problem descriptions. To a lesser extent, variation was also caused by the different experience levels of the investigators with a particular modelling technique or tool.

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.

The exercise of producing analysis/design artifacts under conditions similar to those intended for CADPRO usability laboratory experimentation did, however, produce many insights into what will be required to make future experimentation successful. The investigators make the following recommendations for consideration in further CADPRO experimentation:

Recommendation 1
Given the inapplicability of the productivity and quality metrics that had been proposed for use in the CADPRO project, an “ideal” solution for comparison could be used to determine quality of the solutions produced by subjects. Explicit checklists of good and bad solution characteristics could be drawn up. This effectively means defining task-specific metrics and reflects the way solutions are assessed under university examination conditions. Checklist development may benefit from trialing the problems with large numbers of students in a laboratory setting. An alternative approach would be to have a panel of experts rank the solutions and to have the experts elucidate their ranking criteria.

Recommendation 2
The time allowed to produce solutions should be extended to allow for iteration, a better chance of completeness, and for larger artifacts to be produced. It is suggested that at least 2 hours be allowed. This could be further extended if subjects were allowed to take natural comfort breaks. It will be important, however, to check that subjects neither get tired or bored with such prolonged sessions. Subjects could be allowed a full morning or afternoon though mornings are possibly the preferred option so that subjects don’t carryover concerns from their morning activities. An extended period will allow problem statements to be larger than the anticipated half page: larger problem statements will be needed

to avoid problems with assumptions. With more complete artefacts, the rejected metrics may become applicable.

Recommendation 3
If subjects are given an extended time, their completion times should be noted. It would not make sense to hold subjects back for possibly quite lengthy periods of time if they feel they have done all that they can in terms of solving the problem given them. Data concerning product quality and the amount of product delivered should be trended with the completion time data to explore for any trade-offs. Quality/time may prove to be a useful productivity metric. There should be at least two product delivery measures to include the case of counting deleted units. It simply may not be possible, however, to determine sensible productivity measurements over only a few hours of work.

Recommendation 4
Subjects could be asked to perform a maintenance task instead of solving a problem from scratch. Providing a context should help to minimise the number of assumptions subjects may feel they have to make and the expectation would be for subjects to complete the maintenance task. In DFD modelling, the provision of a context diagram would quickly and clearly convey the problem boundary to subjects.

Recommendation 5
Problem statements could be reverse-engineered from sample solutions. Such an approach could help to alleviate problems with assumptions etc.

Recommendation 6
Problem statements could be carefully re-worded to minimise the need for users to make assumptions. For example, all three investigators assumed they did need not to develop classes for a graphical-user interface for the MAPLEX problem. The problem statement could state "do not create user-interface classes". Such an approach may require repeated re-wording and trialing. It is necessary to perform an iteration of this approach to determine to what extent the assumption problem persists.

Recommendation 7
As the main goal of the CADPRO project is to measure the effect that constraints have on productivity and quality, problems could be specified to ensure that subjects will trigger the constraints present in the tool. A rationale is needed to decide on the constraint environments of interest. For example, it may be of interest to observe subject behaviour when they are forced to revise the current solution either when a DFD model has reached a set limit for the number of processes or when a class diagram has reached a set limit for depth of inheritance.

Recommendation 8
There is a need to establish a rationale concerning the number of diagram types subjects work with. Analysis will be complicated if DFD modelling incorporates both DFD and entity-relationship diagrams or if class modelling incorporates both inheritance and state transition diagrams. Should subjects be concerned about the details of data dictionary entries? Should subjects try and be as complete as possible in modelling inter-class relationships such as aggregation and message passing? These issues need to be resolved. What may be natural for the subjects to do is not obvious without greater insights into industrial/everyday practice.

Recommendation 9
The tools subjects work with must have good user interfaces. If existing tools are used, the latest versions must be employed. If custom tools are used, their interfaces must be extensively prototyped to as near to walk-up-and-use quality as possible. Appendix D lists user interface problems encountered by the investigators.

Recommendation 10
In future data analysis, product delivery measures should be profiled against time and constraint activations to determine how subjects went about problem solving and the effects of constraint activations on subject behaviour. It is recommended that, if possible, such an analysis should take place on the next pilot study and that video records are made.

6. Conclusions

The metrics examined were not found to be suitable indicators of user productivity and product quality for the proposed usability laboratory experiments. To reduce unwanted variability in future experimentation, problem descriptions should be more detailed, more time should be allowed, user-interfaces should be of a high quality, and users should be experienced with the modelling technique and tool. Quality can be assessed by specific checklists or by a panel of experts. In future data analysis, product delivery measures should be profiled against time and constraint activations.

Acknowledgments

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 (JRCASE), a DEETYA TIL Grant to Macquarie University in conjunction with UNSW (JRCASE and CAESAR), JRCASE Centre funding, CAESAR Centre funding, the Nara Institute of Science and Technology, Macquarie University, and the University of Strathclyde.

Thanks to Professor R. Jeffery for comments on a draft of this paper. Thanks to Dr D. Day for providing candidate problem descriptions and useful discussions.

References

[1] D. Day. User responses to constraints in computerised design tools: An extended abstract. *Software Engineering Notes*, 21 (5):47-50, September 1996.
[2] D. Day. *User Responses to Constraints in Computerized Design Tools*. PhD thesis, School of Information Studies, Syracuse University, 1995. UMI Order Number 95-44905.
[3] D.Day, M. Ahuja, and L. Scott. Constraints in design engineering: a report of research in progress. In *8th Australian Conference on Information Systems*, pages 509-516, 1997.
[4] M. Parchkova and D. Day. Selection of Software Quality Metrics for Limited Products in Usability Laboratory Observation. CAESAR Technical Report 98/1, School of Information Systems, University of New South Wales, 1998.
[5] R. McDonald and D. Day. Productivity in the Context of Constraints: Selecting Metrics for CASE Use. CAESAR Technical Report 98/6, School of Information Systems, University of New South Wales, 1998.
[6] N. F. Schneidewind. Methodology for validating software metrics. In J. J. Marciniak, editor, ENCYCLOPEDIA OF SOFTWARE ENGINEERING, volume 1, pages 666-676. John Wiley & Sons, Inc., 1994.
[7] J. Daly, J. Miller, A. Brooks, M. Roper, and M. Wood. A survey of experiences amongst object-oriented practitioners. In *Proceedings of the IEEE Second Asia-pacific Software Engineering Conference* , pages 137-146, 1996.
[8] J. Daly, A. Brooks, J. Miller, M. Roper, and M. Wood. Evaluating inheritance depth on the maintainability of object-oriented software. *Empirical Software Engineering*, 1:109-132, 1996.
[9] I. Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, fourth edition, pages 600-603, 1992.
[10] S.R. Chidamber and C.F.Kemerer. A Metric Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, 20(6):476-493, 1994.
[11] Scott, L. Hypernode model support for software design methodology modelling. JRCASE Technical Report 97/2, School of MPCE, Macquarie University, 1997.
[12] Lionel C. Briand, John W. Daly, and Jurgen Wust. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. *Empirical Software Engineering*, 3:65-117, 1998.

APPENDIX A

Problem Descriptions

Mail Order System

A mail order company advertises products in magazines. Most orders are initiated by magazine subscribers who fill in and send coupons to the mail order company. The company takes orders over the phone, answer inquiries about products, and handles payments and cancellations of orders. Products that have been ordered are sent either directly to the customer or to the regional offices of the company, which then handle the required distribution. The mail order company has three basic data files that retain customer mailing information, product inventory information, and billing information based upon invoice number. During next few years, the company expects to become a multimillion-dollar operation. Recognizing the need to computerize much of the mail order business, the company has begun the process by calling in a systems analyst. You are that analyst.

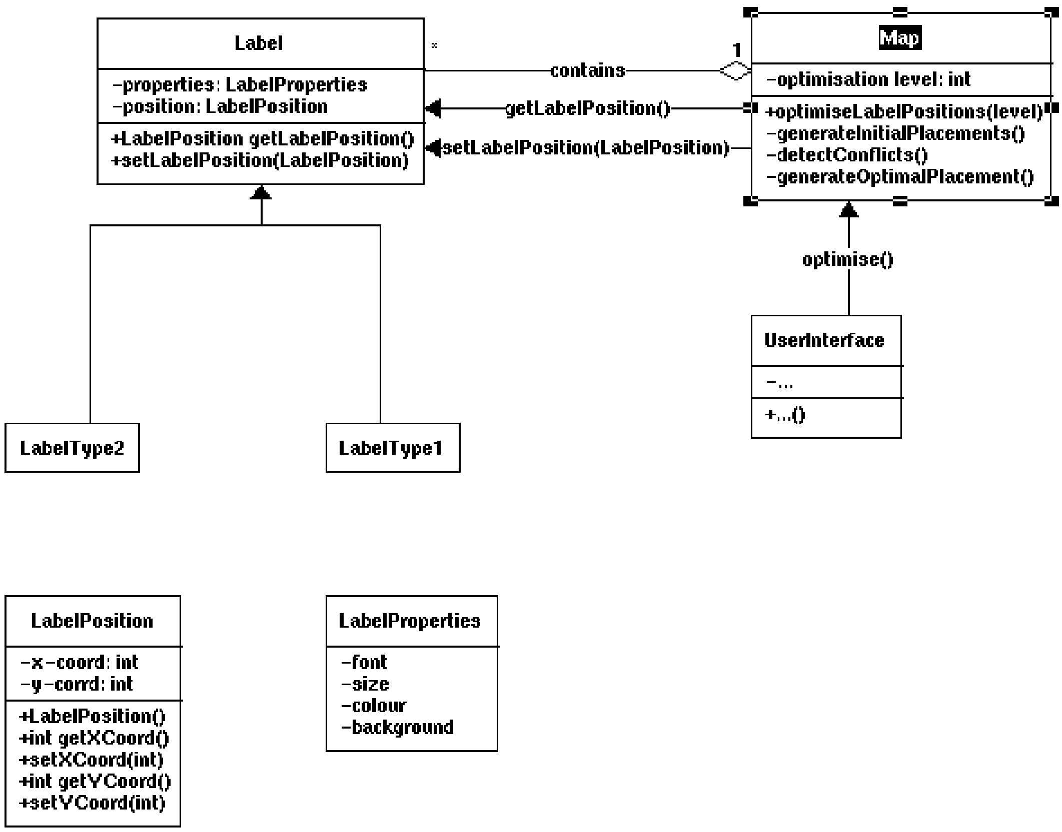
Draw a complete set of logical DFDs for the above system.

MAPLEX (The Computer Bulletin MARCH 1998)

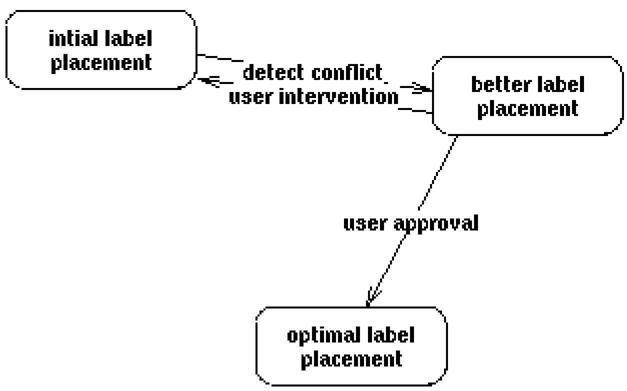
Maplex is an automatic system to put names on maps. Although it has only been on sale since 1996, it had already been bought and used by seven of the world's major map makers. Then it was taken over by Environmental Systems Research Institute, the major US geographic information systems firm. Putting text onto a map is an extremely complex task. Each name must be placed so that it belongs to the feature it is meant to label - but it must not obscure anything else. If it takes 30 seconds to position each label by hand, the 20,000 labels on a typical large map will take almost 170 hours, half of the total production time. Maplex can place 1,000 labels in five minutes. The effort of Chris Jones and his team in developing Maplex has been split between realising these speeds and making it easy to use. Initially it generates possible label locations for the different kinds of labels needed on the map. The program then detects overlaps, and resolves any resulting conflicts to produce an optimal set of label positions. Starting from an initial overall trial solution for the whole map, a multiphase optimisation process moves towards the final optimum. The user chooses one of three levels of optimisation the lowest will produce a map in two minutes, the highest level might take 15 minutes. Map publishers need to balance information, legibility and artistic effect. A flexible user interface gives cartographers a comprehensive range of options. They have complete control over every aspect of the label to be placed on the map - font, size, colour, background and so on. With these options a firm can establish a unique house style to be applied to every map in a publication.

APPENDIX B

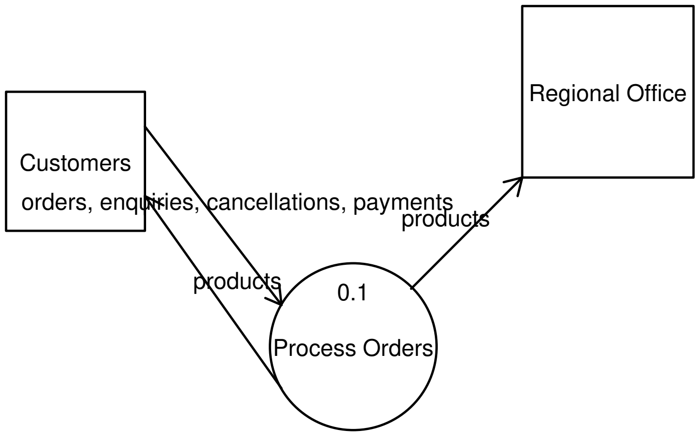
Solutions (diagrams)



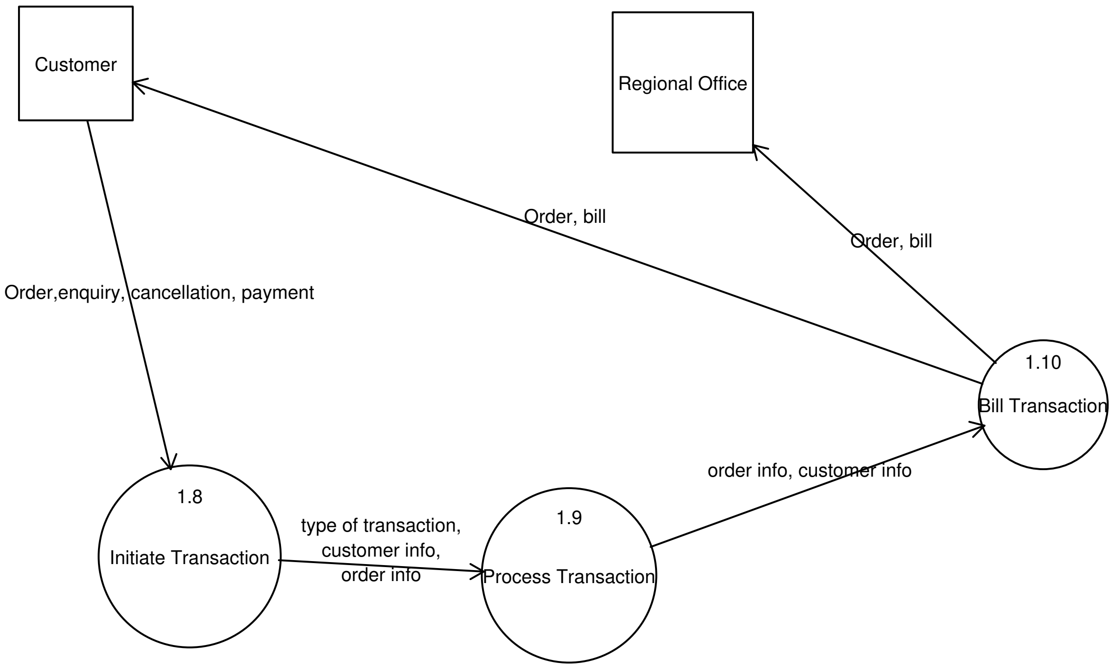
Investigator L's class diagram for MAPLEX system



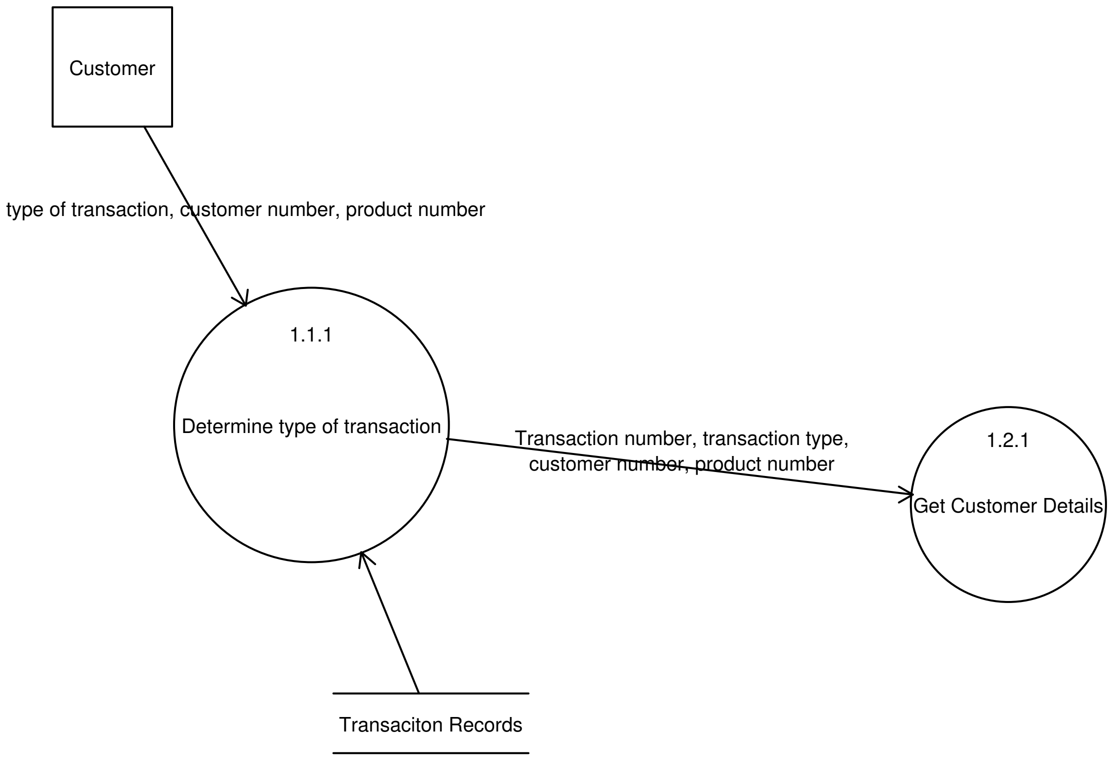
Investigator L's state diagram for MAPLEX system



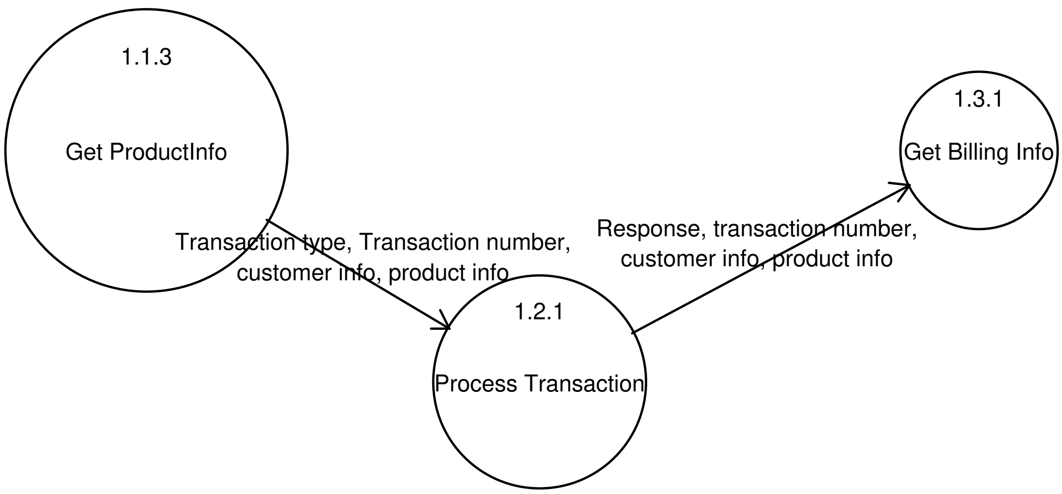
Investigator L's context diagram for the Mail Order System



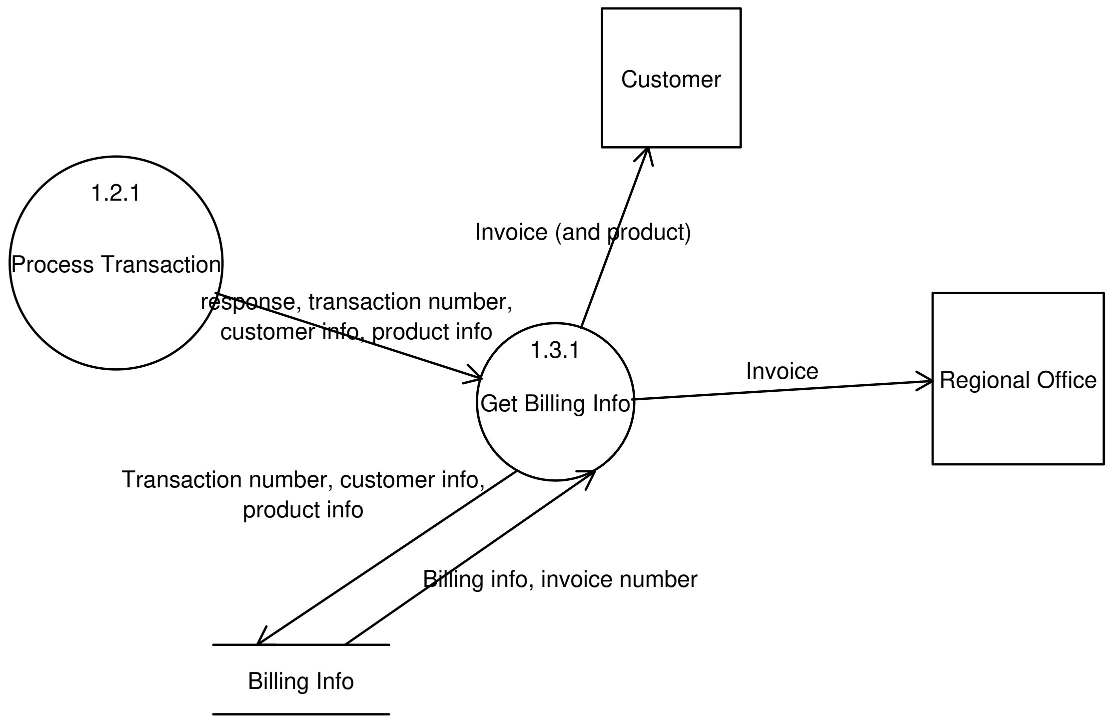
Investigator L's level 1 DFD diagram for the Mail Order System



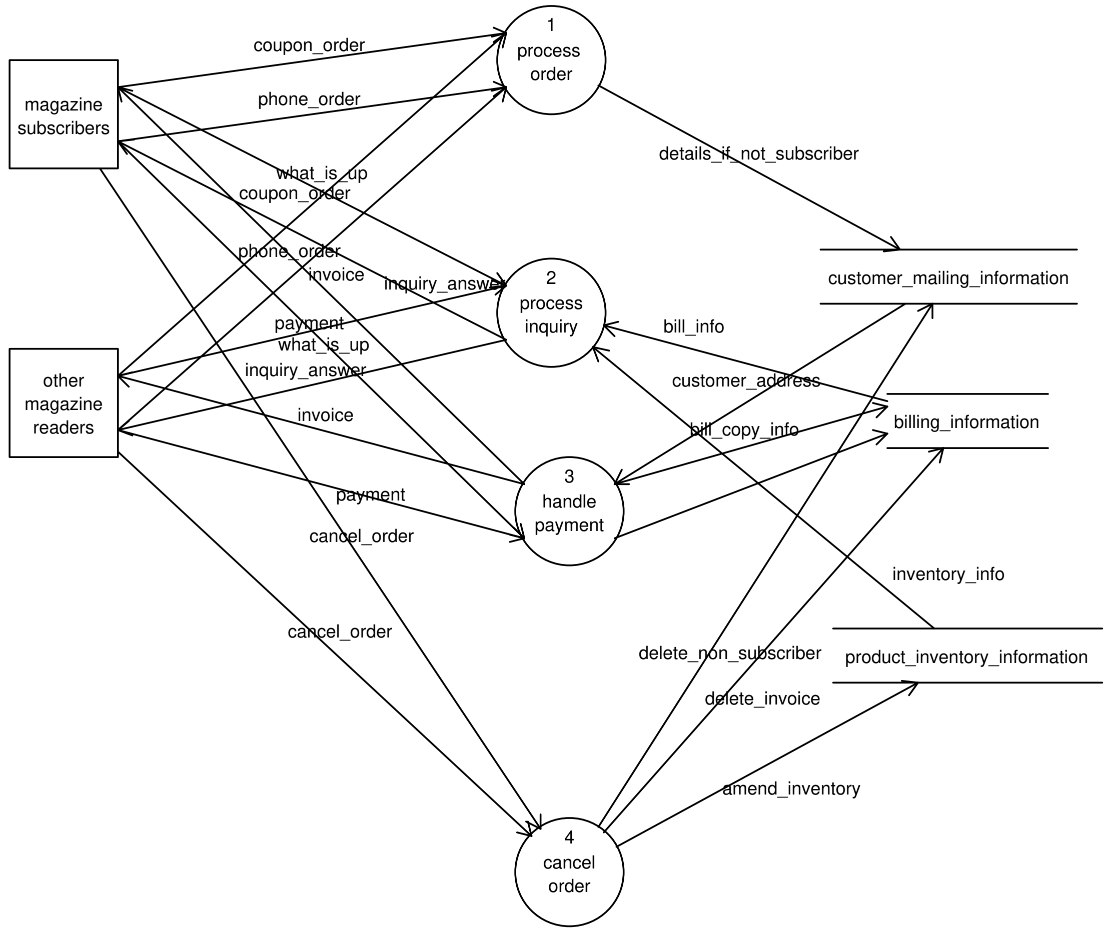
Investigator L's level 1.1 DFD diagram for the Mail Order System



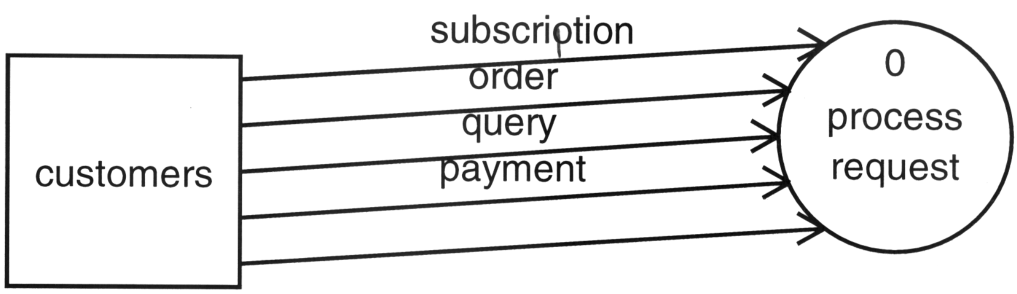
Investigator L's level 1.2 DFD diagram or the Mail Order System



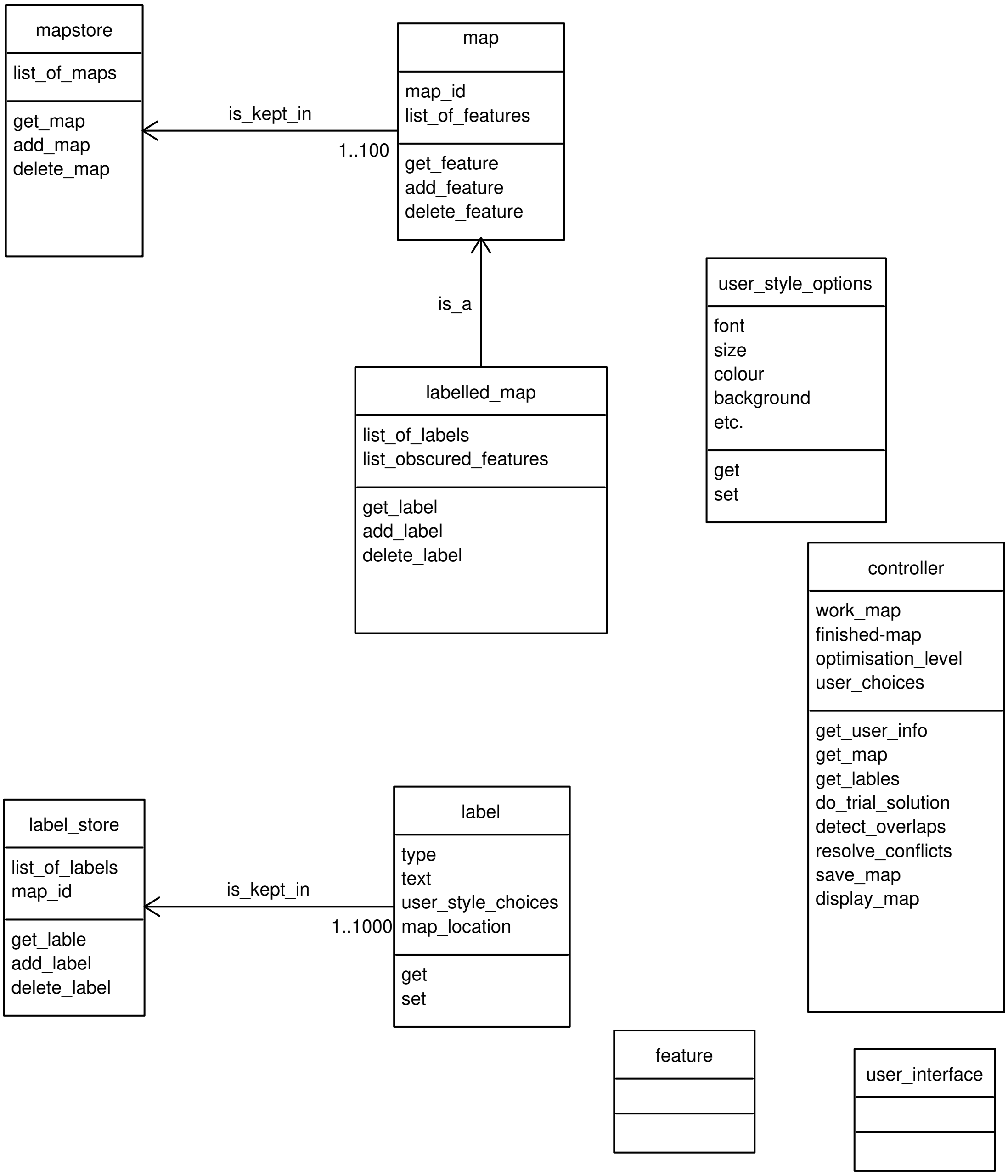
Investigator L's level 1.3 DFD for the Mail Order System



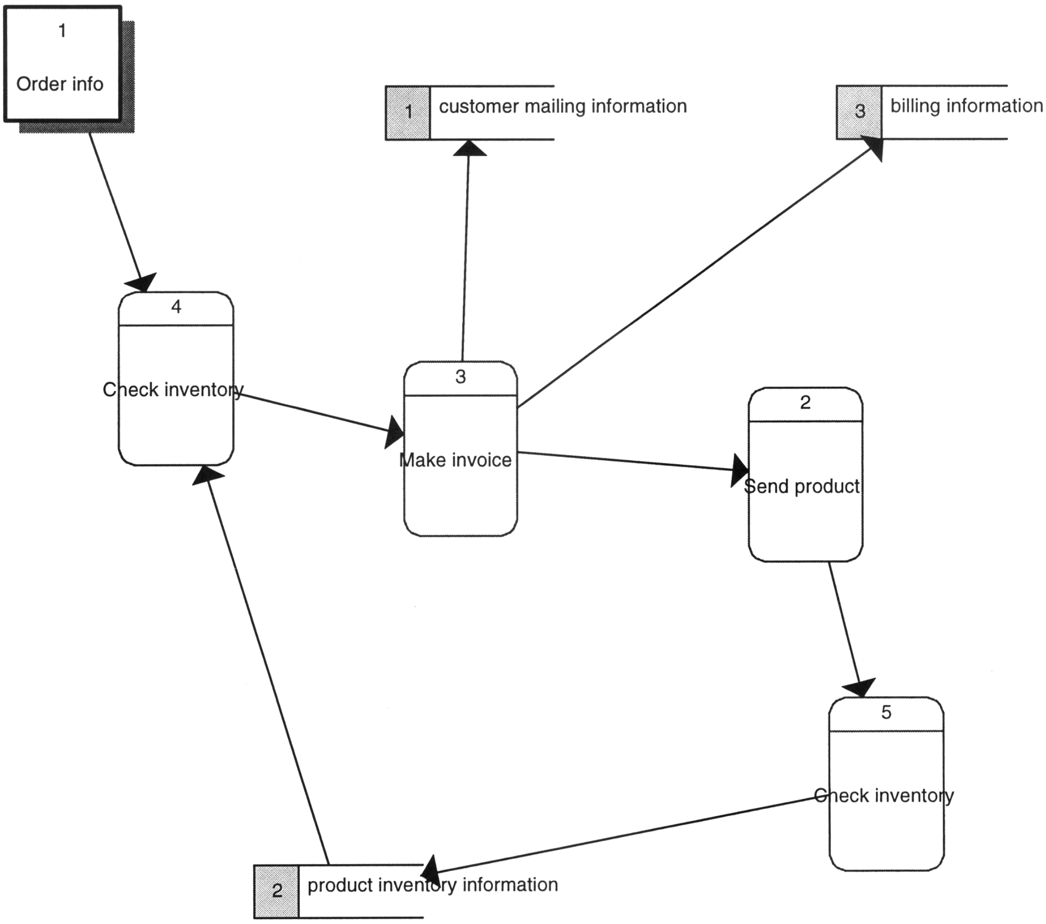
Investigator A's DFD diagram for the Mail Order System



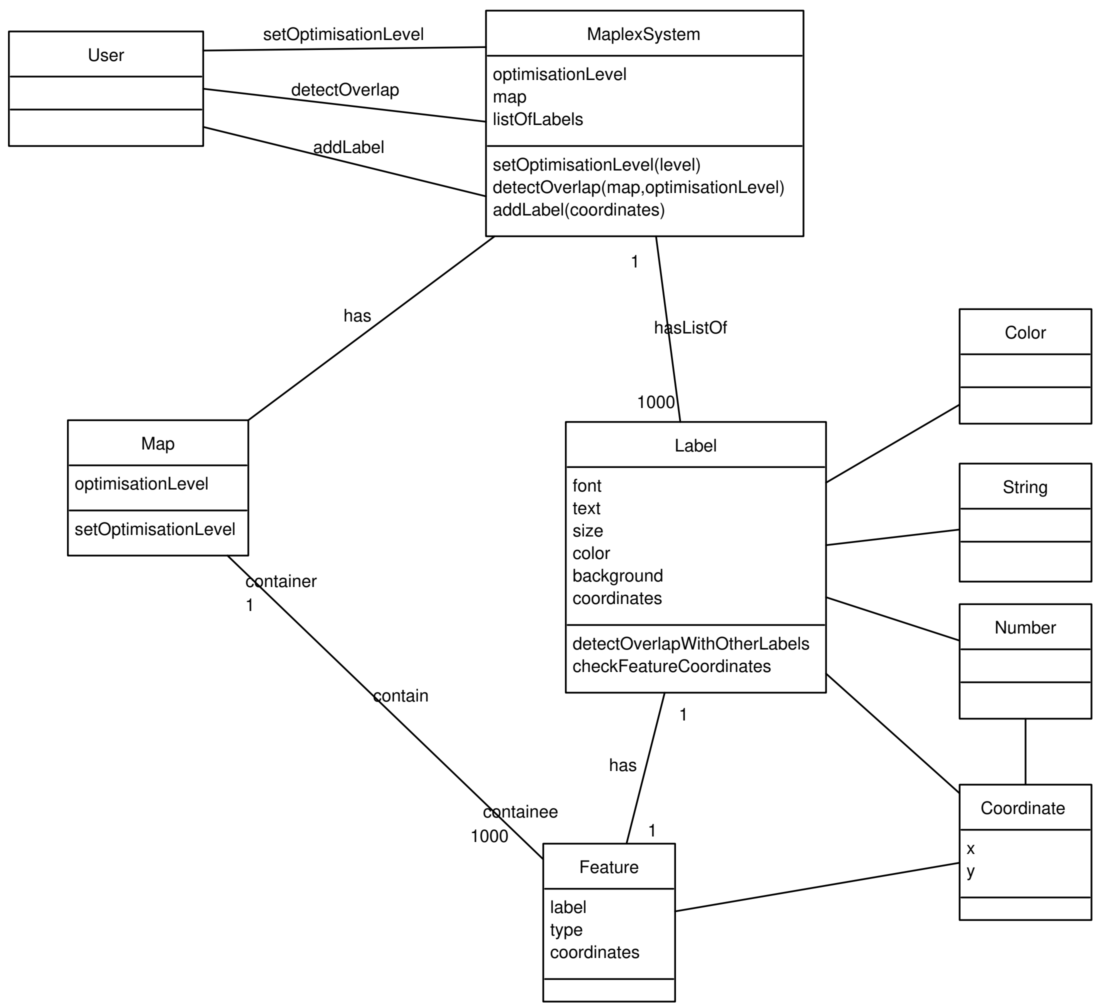
Investigator's A's late attempt at drawing a context diagram for the Mail Order System



Investigator A's class diagram for the MAPLEX system



Investigator S's DFD diagram for the Mail Order System.



Investigator S's class diagram for the MAPLEX system

APPENDIX C

Assumptions

List of encountered assumptions for the Mail Order System

Do you have to delete a non-subscriber's mailing details after a cancellation?
Do you allow for prepayment with cheques, credit cards etc?
Do you have to deal with the process of subscribing to the magazine?
Do you have to model the process of paying for advertisements to be designed and placed in the magazine?
Do you have to model the Internet and its business use?
Do you have to fully document processes/flow/entities or just simply name them? For example, should you write a process description?

List of encountered assumptions for the MAPLEX system

Do you have to model the process of feature addition and deletion to maps?
Do you have to model the process of label addition and deletion to the system?
Do you have to model feature details?
Do you have to model the displaying of maps?
Do you simply assume features and labels have a 1:1 cardinality relationship?
(Perhaps a feature can have more than one label?)
To what extent do you have to model the list of labels? (Can you assume they are already in a data-store?)
Do you have to model the storage of raw/processed maps? (Do "raw" maps need to be kept?)
To what extent should class relationships and message passing between classes be specified?
Can other diagram types be used to aid the design work?

Appendix D

Interface Problems

List of interface problems encountered for the Mail Order System

In TCM, to directly edit a comment, you are required to use arrow keys and not mouse to select insertion/deletion point. This is a non-intuitive implementation of editing.
In TCM, no special data-flow symbols to show record deletion.
In TCM, levelling not explicitly supported (but levelled diagrams still achievable). There was no automated support for numbering processes or checking consistency between levels.
In TCM, data-flows had to be connected at both ends, making simulated diagram levelling awkward.
In TCM, only allowed to work with one diagram at a time, so had to print out diagrams while trying to simulate levelling.
In TCM, problems with delete button not functioning as it should (had to "cut" instead of delete).
In TCM, no special symbols to signify material flows and stores (workaround by L, stress by A)
In TCM, automatic process numbering goes awry when creating more than one diagram.
In TCM, disappearing label problem during creation when several data-flows already entered and labelled from the same external entity.
In MetaEdit, no automatic numbering.
In MetaEdit, data-flows had to be connected at both ends.
In MetaEdit, mode approach meant you could easily start creating an unwanted flow or process.
In MetaEdit, the process of drawing data flows was error prone because the action was inconsistent with previous experience. Initially it was unclear how to create data flows.

List of interface problems encountered for the MAPLEX system

In TCM, often started creating an unwanted class when another action was desired.
In TCM, could not use 1..N for cardinality so guessed a number for N such as 100 or 1,000. (But would "*" be acceptable to TCM?)
In TCM, with a 4-digit cardinality, user forced to move linked classes apart so that cardinality number is clearly visible.
In TCM, class relation symbols unclear for novice user.
In TCM, mouse sensitive areas very close together so the activity of specify roles or cardinality on class relations was prone to selection errors.
In TCM, could not separately move cardinality Role to make diagram clear.
In Paradigm, cardinality had 1:* default, with apparently no way of specifying 1:1 cardinality on class relations.
In Paradigm, often started creating an unwanted class when another action was desired.