INFORMATION
SCIENCE
TECHNICAL
REPORT

NAIST-IS-TR99003
ISSN 0919-9527

Evaluating data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #2)

Shingo Takada, Louise Scott, Andy Brooks

February 1999

NAIST

〒 630-0101
奈良県生駒市高山町 8916-5
奈良先端科学技術大学院大学
情報科学研究科

Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, Japan

Evaluating data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #2)

Shingo Takada

Louise Scott

Andy Brooks

Nara Institute of Science and Technology

Information Science Technical Report

NAIST-IS-TR99003

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. 98/15)

Evaluating data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #2)

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)

LOUISE SCOTT Joint Research Centre for Advanced Systems Engineering (JRCASE), School of MPCE, Macquarie University, Sydney, NSW 2109, Australia (louise@mpce.mq.edu.au)[1]

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

Abstract

This paper reports on a second pilot experiment for the CADPRO project taking into account a number of the recommendations arising from the first pilot. Three subjects produced data flow diagrams and class diagrams in a usability laboratory setting. Characteristics of subjects' solutions are discussed and the usefulness of profiles of product delivery and constraint activations against time is determined. Reports are given of the current status of the CASEMaker tool, which is being developed for the main CADPRO experiment, and the current status of the experimental design. Many insights were obtained 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. If the program of research eventually leads to such a determination, lessons may also be learnt regarding the design of computer-user interfaces for any computerised tool. 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 [7]. 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 first pilot study was undertaken because of concerns about the planned duration of experiments (50 minutes) and the need to evaluate a number of metrics proposed to measure the final outcomes (user productivity and product quality) in the CADPRO project [4,5]. In the report of the first pilot [6], it was recommended that one or more additional pilot studies be carried out to iterate problem descriptions and to examine profiles of product delivery and constraint activations against time.

This paper reports on the results of the second pilot study that took into account a number of the recommendations given in [6]. The purpose of this paper is:

- To report on factors discovered influencing user productivity and product quality
- To report on how productivity and quality might be measured in the future
- To evaluate the current status of the CASEMaker tool which will be used in later CADPRO experimentation.

[1] Now at Fraunhofer IESE, Sauerwiesen 6, D-67661, Kaiserslautern, Germany

- To evaluate the current status of experimental design, including the problem descriptions, duration, etc.

Three subjects undertook analysis/design tasks using problem descriptions modified from the first pilot. Video and audio records were made of the computer displays and subjects' utterances. For tasks involving use of CASEMaker, profiles of product delivery and constraint activations against time were also recorded. Our analysis was based on all these recordings, the artifacts produced, and various subject debriefings. This second pilot produced further insights into what will be required to make future experimentation successful. A number of recommendations are given.

Section 2 describes the experimental method including which recommendations from the first pilot were acted upon. Section 3 gives timing, quality self-evaluation, and perceived problem difficulty results. Section 4 discusses the characteristics of subjects' solutions. Section 5 analyzes subjects' time-stamped delivery profiles for the DFD diagramming tasks: we believe such profiles can be used in the assessment of user productivity and perhaps quality as well. Section 6 evaluates the current status of CASEMaker. A summary and recommendations for the research program are given in Section 7. Conclusions are given in Section 8.

Appendix A gives the English versions of instructions to subjects and problem descriptions along with the Japanese versions of the problem descriptions. Appendix B reproduces subjects' solutions. Appendix C lists assumptions made by the subjects. Appendix D lists interface problems encountered. Appendix E gives the CASEMaker training sheet given to subjects. Appendix F details the configuration of CASEMaker used during the experiment. Appendix G lists required fixes to CASEMaker. Appendix H provides sample solutions by one of the authors to the problems given in Appendix A.

2. Method

2.1 Introduction

This second pilot study involved subjects undertaking analysis/design tasks modified to take account of concerns that arose in the first pilot. In the first pilot, subjects had undertaken the tasks in a normal setting: in this study, subjects undertook the tasks in an actual usability laboratory, part of the software engineering laboratory at the Nara Institute of Science and Technology. This usability laboratory had two recording areas separated by a control and observation room, allowing two subjects to take part at the same time. 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 time limit of 2 hours, and subjects did not use pen and paper to help them solve the problems. As the CASEMaker tool was now sufficiently developed, subjects for the task requiring data-flow diagram (DFD) modelling used a DFD tool generated by CASEMaker. In what follows, it will be made clear which recommendations from the first pilot were taken into account.

Sample solutions (Appendix H) were used as reference points in evaluating the solutions produced (Recommendation 1 from the first pilot). Subjects' completion times were noted (Recommendation 3). Profiles of product delivery and constraint activations against time provided by CASEMaker were analysed (Recommendation 10).

2.2 Subjects

In the first pilot, the investigators themselves were the subjects. In this second pilot, three subjects (henceforth, Subject A, Subject B, and Subject C), all from the staff at the Nara Institute of Science and Technology, took part in the experiment. The experience level of the three subjects with respect to DFD modelling and class modelling are shown in Table 1. (In the first pilot, the different experience levels of the investigators with a particular modelling technique or tool contributed to the variation between the artifacts produced.) All three subjects had not made previous use of the tools used in this experiment.

Table 1. Modelling experience of subjects

	DFD modelling	Class modelling
Subject A	Only experience was about 4 years ago in a class.	Very little experience; one small C++ program.
Subject B	Taught DFD modelling in classes some 5-7 years ago.	Very little experience.
Subject C	Has done some modelling (2-3	Has done some modelling (2-3

	times) in the past.	times) in the past; has programming experience in C++ (5 years) and Java (2 years).

2.3 Tasks

As with the first pilot, subjects were required to perform DFD modelling for a Mail Order System and class modelling for an Automatic Map Labelling System. The problem descriptions for these two systems were modifications of those used in the first pilot to take account of concerns that arose as a result of the first pilot. The problem descriptions and instructions were given in English, but as the subjects were Japanese, they were also provided with Japanese translations. It was anticipated that the subjects might wish to consult the English version to aid with naming, as the tools they would be using did not have a Japanese language interface. The English versions of the instructions and problem descriptions along with the Japanese versions of the problem descriptions are given in Appendix A. Care was taken to try and avoid the problems that occurred in the first pilot when subjects made unexpected assumptions while solving the problems (Recommendation 6 from the first pilot). At the end of the first day of experimentation, however, it was felt necessary to make some additional changes to the instructions and problem descriptions as follows:

Changes in the DFD modelling problem instruction (DFD1 -> DFD2)

- The DFD1 description explicitly noted to "not assume the existence of any additional external entities or data stores". This was changed to "not assume the existence of any additional external entities" in the DFD2 description.

Changes in the class modelling instruction and problem description (Class1 -> Class2)

- The instructions were made clearer so that the subjects would not model classes or methods for the user interface and basic class operations concerning creating and destroying.
- The Japanese version was found to be ambiguous as to whether labels are placed on one map or multiple maps. Thus, the Japanese description was changed to explicitly read "one or more maps".

Table 2 shows the schedule for the experiment. DFD1, DFD2, Class1, and Class2 refer to the actual problem descriptions and instructions that were used. Subjects were given a maximum of two hours to solve the problem (Recommendation 2 from the first pilot).

Table 2. Schedule for the experiment in Pilot #2

	DFD	Class Modelling
August 6, 1998 PM	Subject C (DFD1)	Subject A (Class1)
August 7, 1998 AM	Subject B (DFD2)	Subject C (Class2)
August 7, 1998 PM	Subject A (DFD1)	Subject B (Class2)

2.4 Modelling tools

For each task, different tools were used. The *Toolkit for Conceptual Modelling* (TCM Version 1.6.6)[2] was used for class modelling while CASEMaker was used for DFD modelling. The version for TCM was the latest version available on the web (Recommendation 9 from the first pilot). Prior to using the Toolkit for Conceptual Modelling, subjects were introduced to the tool and given 10 minutes to familiarise themselves with the tool. Prior to using the CASEMaker tool, subjects were given a one-page training sheet containing instructions on how to use the tool (reproduced in Appendix E), a walkthrough demonstration of how to do basic editing, and 10 minutes to familiarise themselves with the tool. (It had been the intention to use the demonstration version of the Rational Rose Case Tool for class modelling but an installation problem prevented its use.)

2.5 Use of think-alouds

The usability laboratory also had provision for audio records to be made through the use of unobtrusive microphones. It was decided that such records would help with the analyses. 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, problems, etc.

[2] http://wwwis.cs.utwente.nl:8080/~tcm/index.html

2.6 Subject debriefings

While subjects conducted the tasks, one or more investigators monitored (video and audio) what took place from the control room in the usability laboratory. Sometimes recording cameras were zoomed or panned to allow a clearer picture of what was taking place on the computer screen. (Only on one or two occasions was it necessary to enter the work area and ask the subject to sit more to one side to avoid blocking the camera's view of the screen.) Lists of questions were drawn up for subject debriefings. One or more investigators debriefed the subjects immediately after they had finished. The sample solutions to the tasks were used as an aid during debriefing.

Later, subjects were debriefed to determine their self-evaluations of the quality of their solutions and their assessments of the relative problem difficulty between the two tasks.

2.7 Unfulfilled recommendations from the first pilot

Some of the recommendations from the first pilot were not acted upon. They are:

- Recommendation 4 (have subjects perform a maintenance task to help minimise assumption making)
- Recommendation 5 (reverse-engineer from a design solution to help minimise assumption making)
- Recommendation 7 (specify problems to ensure constraints are triggered)
- Recommendation 8 (establish a rationale concerning the number and nature of diagram types subjects should work with rather than have subjects work with just one diagram type)

Also, two recommendations were only partially fulfilled:

- Recommendation 1 (Although sample solutions were provided, checklists of good and bad solution characteristics were not)
- Recommendation 9 (Because of time constraints, CASEMaker had not been extensively prototyped beforehand)

3. Timing, self-evaluation, and problem difficulty results

3.1 Timing results

Table 3 shows the time each subject took to produce a model (in hours and minutes).

Table 3. Time taken to produce a model

	DFD	Class Modelling
Subject A	1:03	1:57
Subject B	1:52	0:59
Subject C	0:56	0:43

In DFD, only Subject B came close to taking the whole allotted maximum 2 hours. Note that Subject B was the subject with the most experience with DFD modelling. One interpretation of this is that the experience led the subject to think of many things making him take that much longer to finish the model. Also, although it took him 1:52 to say that he was finished, he was basically finished after about 1:08. As will be noted again in section 5.2, at this point Subject B basically thought that he was finished but wanted to go over the diagram to make sure. So, for about the next 44 minutes, he checked the diagram and made some changes to it while moving the produced nodes and edges. Thus, all three subjects were more or less finished in about one hour.

In the class modelling problem, only Subject A came close to the allotted maximum 2 hours. Unlike Subject B in the DFD modelling, this was not caused by time taken to extensively check the model. Subject A continually made changes to his model for much of the time. All three subjects did not have a lot of experience with class modelling, although Subject C did have extensive object-oriented programming experience. As will be noted again in section 3.3, Subject C thought that the problem was one that could be seen in examples, thus the class modelling problem may have been fairly easy for him causing him to finish fairly quickly.

3.2 Subjects' self-evaluations of quality

Table 4 gives subjects' self-evaluations of the quality of their own diagrams on a scale of 1 (bad) to 10 (excellent). Each subject was first asked to self-evaluate his own diagrams on the morning of August 8 ("First" columns in Table

4). All three subjects were then asked to go to another room and discuss between themselves their biggest problems with their diagrams. Until this meeting, they were prohibited from talking to each other about the diagrams. After this discussion, they were again asked for a self-evaluation ("Second" columns in Table 4). Note that only Subject B's class diagram evaluation went up. Originally he was unsure as to how good his diagram was because of his inexperience. Looking at the diagrams produced by the other subjects and the discussion between them had made him a little more confident of his diagram.

Table 4. Self-evaluation of solution quality

	DFD		Class	
	First	Second	First	Second
Subject A	5	5	5	5
Subject B	8	8	3	4
Subject C	3	3	5	5

3.3 Subjects' assessments of problem difficulty

Subject A noted that he had little experience in data flow and class diagramming, but thought that DFD modelling was easier since "it was easier to understand the flow of data". Subject B also thought that the DFD problem was easier, but this was due to his experience with data flow diagramming and lack of experience with class diagramming. Finally, Subject C thought that the class diagramming was much easier, one of the reasons being that the problem was one that could be seen in examples. There being no unanimity, and given subjects' reasoning, no real conclusion can be drawn about the relative difficulty of the two modelling problems.

3.4 Methodology checker

Subjects never called the methodology checker either when DFD modelling or when class modelling. None of the subjects were told about the existence of the methodology checker in the class modelling tool, while this was explicitly written in the training sheet for the DFD tool.

4. Discussion of artifacts produced

The artifacts produced by subjects are given in Appendix B. This section discusses the solution characteristics of these DFDs and class diagrams. Some of these characteristics arose because of assumptions made by subjects concerning the problems, i.e. the assumptions problem had not been eliminated. Appendix C lists these assumptions. Appendix H gives the sample solutions to the problems worked up by one of the authors. This section discusses factors discovered influencing the quality of subjects' solutions. Ideas about the quality of DFDs are, in part, derived from [8]. Ideas about the quality of class diagrams are, in part, derived from [9]. Subjects' self-evaluations of solution quality point to the importance of experience with the modeling technique. Subject B gave himself by far the highest rating of solution quality of any model for his DFD model: this subject had by far the greatest experience in any modelling technique namely in DFD modelling. But several other factors influenced solution quality as discussed below.

4.1 Solution characteristics of the DFD models

- Data stores and multiple processes in the context diagram

Violations of constraints for context diagrams are an indication of a poor quality solution. Subject A and Subject C did not have much experience with data flow diagramming and it became clear that they did not have an understanding of what a context diagram really is. Both subjects' top diagrams violated the basic constraint for context diagrams, i.e. that there can be only one process. Subject C's "context" diagram was really his Level 0 diagram. So poor quality here is simply attributable to lack of experience with the modelling technique.

Interestingly, Subject A originally had three external entities; one each for customer, company, and regional office. But later he changed "company" and "regional office" to processes, as he thought that "the company is the system itself" and "the regional office seemed to be more 'internal' than 'external'." This modeling change need not necessarily be viewed as resulting in a poorer quality solution: determining system boundaries and the sources and sinks of data is not always easy, especially in the absence of a context where analysts are free to ask questions to clarify system boundaries.

- Missing customer inquiry data flow

Missing flows are an indication of poor quality. Both Subject A and Subject C did not have data flows and processes to handle customer inquiries. This was due to their not understanding what a customer inquiry would be, even though a Japanese translation was given to them. Subject A noted that "customer inquiry is too ambiguous compared with others, such as cancellation". Poor quality here is probably due to lack of domain experience combined with lack of experience with the modelling technique.

- No outgoing or incoming data flows from data stores

Data stores that are not read from, or that are not written to, suggest incompleteness and thus poor quality. Subject A and Subject B both had a data store ("Billing Info") which had no outgoing data flow.

Subject A did not realize this until debriefing but noted that an outgoing data flow would probably be unnecessary unless it was necessary to model for contingencies. 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 (see above), then poor quality here can again be attributed to a lack of experience.

Subject B, who was experienced with this modelling technique, 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, including payment information, answers to inquiries, shipping information, and order/cancellation information. 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 misunderstandings caused by the translation process.

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 another system, and that this was outside the scope of the problem. Again, this boundary issue would have been easily resolved in context when analysts can ask questions to clarify the problem boundaries. 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

Having data flows that do too much reduces quality: module interfaces should be narrow not broad. Subject B originally had "customer info" data flows going from the "Customer Info" data store to the "Handle order" process. But he decided to redirect the "customer info" data flow to the "Dispatch Input" process from where various data flows such as "Inquiry Info", "Cancellation Info", and "Order Info" would go to processes such as "Handle Order", "Handle Cancellation", and "Handle Inquiry". The subject intended these data flows to also contain information on the customer, although this was not explicitly shown in the diagram. (Any ambiguity in the subject's intentions could have been immediately resolved if the task had included the setting up of a data dictionary.) This overloading of flows is indicative of a design/implementation decision. Subject B thought that separately sending "Customer Info" to each process was redundant. By sending "Customer Info" to "Dispatch Input", he thought that any future changes could be limited to that flow/process only, while if this overloading was not done, each flow/process would have to have been checked. From an analysis perspective, this overloading represents a poorer quality of solution not only because the flows are doing too much but also because some data is 'tramped' (or unnecessarily directed) through a process unused. In terms of a design/implementation solution, there is not a persuasive argument that such overloading improves maintainability. So lack of quality here has arisen because the subject moved on from the analysis phase to consider issues of relevance later in the software life cycle, despite having been instructed to prepare a logical model.

- Inconsistencies

Any inconsistency represents a worsening in solution quality. 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. Again, lack of quality has arisen because the subject moved on from the analysis phase to consider issues of relevance later in the software life cycle. (Again, any ambiguity in the subject's intentions could have been immediately resolved if the task had included the setting up of a data dictionary.)

- Naming

Badly named entities, stores, processes or flows reduce the quality of the solution: they make the solution more difficult to understand. Subject C had "payment" going to the customer: he meant this to be "invoice". This bad naming of the data flow was found to be due to nothing more than a difficulty with English.

- Additional Data Stores

The instructions in DFD2 given to Subject B were different from the instructions in DFD1 given to Subjects A and C who were told explicitly to not model any additional data stores. Subject C had originally wanted to add an extra data store but didn't due to this constraint in the instructions. As it was possible a subject, free of the data store constraint, might introduce additional data stores, which might have a bearing on the quality of solution or the constraints met, the instructions for Subject B were modified by removing the data store constraint. As it happened, 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". Given this subject's reasoning it would not be appropriate to regard this extra data store as resulting in a poorer quality of solution. It could be argued that mixing logical and physical models is wrong as this is not in keeping with the textbook description of structured analysis: real designers, however, have been found to not comply with the rules and procedures of structured analysis [10].

4.2 Solution characteristics of the class models

- Naming

As with DFDs, badly named classes, attributes, operations, and relationships reduce the quality of solution.

Subject A assumed that the map only contained references to buildings (e.g. the buildings that make up a university), and that there would be no features for roads, mountains, rivers, etc. So instead of a class named "feature" he had a class named "building". The reason for his assumption was partially due to the notion of "circle" and "rectangular" features. Subject A said that: "Probably only buildings could be considered as circle or rectangle. It's difficult to think of roads and mountains as either circle or rectangle." Given the assumption made, the class was not inappropriately named.

Subject A's "label" class had an attribute named "label" and an attribute named "text" i.e. the name "label" was being used twice, which can cause confusion. The problem description noted both "display text" and "default text label". Subject A thought that there may be two or more types of "texts", with one of them actually being the one to be displayed on the map. Thus, the attributes could have been "text1" and "text2", but since he wasn't sure what would be the best possible name, he left it as "label" and "text". Note that the subject himself is aware of a naming problem and this issue would likely be resolved in a context where analysts are free to ask questions.

- Subclassing

Class models can suffer from too much subclassing: too much specialisation can be difficult to understand and difficult to manage. Too little subclassing, on the other hand, reduces the benefits associated with an object-oriented approach. Inappropriate subclassing, where conceptual integrity of inheritance hierarchies is poor, can mean difficulties in understanding [11].

All three subjects had a class for a feature (though in the case of Subject A, it was named the "building" class) 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 perhaps premature to regard this use of a simple attribute as a discriminator as inappropriate: too little is known about differences of behavior between the two types of feature.

Subject C had an "optimisation" class to handle the actual placing of 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 does seem 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, too little is known about the differences of behavior between the three levels of optimisation i.e. optimisation subclassing cannot be categorically stated as a poor approach.

Subject B had two additional inheritance relationships: "Labeled Map" derived from "Map" and "Text Label" derived from "Label". This subject interpreted the problem description as suggesting 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 well have arisen because the problem description mentions 'default labels' and 'default text label'. From this viewpoint, the additional subclassing cannot be categorically stated as a poor approach.

- Multiplicity (Cardinality)

Incorrect multiplicities clearly reduce the quality of solution.

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. The English version stated that the system "automatically places labels on maps" where "maps" is plural. Due to the nature of the Japanese language, "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. Thus the text was changed to explicitly read "one or more maps" in the Japanese version of the second problem description (Class2).

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

The quality of any analysis model can suffer if design and implementation concerns unduly influence the model.

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, on this occasion, the influence has been detrimental. Given that 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.

- Missing system class

A model missing a class for the system may represent a poor quality analysis.

Only Subject A had classes for both the system itself and a map. It is not clear why the system class was not modelled by the other two subjects. It could be due to lack of experience or because the instructions stated not to model the long term storage and retrieval of maps, behavior which might belong to a system class. In an operational context, such a class would not be missing for long.

5. Product delivery and constraint activation profiles in DFD modelling

CASEMaker recorded time-stamped delivery profiles for the DFD diagramming performed by the three subjects. We consider these profiles for measuring productivity. Creations and deletions of nodes and edges were recorded. Renaming actions were also recorded but these were incomplete if a subject chose to rename using the Properties dialogue box. Delivery counts were found to be slightly inaccurate because of phantom edge creations: in edge-create mode, if a subject clicked on a process/external entity/data store, but then chose not to complete by performing a drag, CASEMaker recorded an edge creation. The delivery profiles were visually examined for the presence of dips. A number of deletions followed by creations may represent productivity loss. Though there were problems with the recording of the delivery profiles by CASEMaker, this did not effect the detection and interpretation of dips in the delivery profiles.

There were, however, more serious problems with the recording of the constraint profiles by CASEMaker. For example, the profiles did not always accurately show the duration of constraint violations.

5.1 Subject A

Subject A suffered two crashes but no profile or solution data was lost. The first crash required minor intervention work: there were only 1 or 2 minutes of disruption to the subject and recording of the delivery profile (Figure 1) was maintained. The second crash, however, involved the creation of an additional delivery profile (Figure 2). Note that

Figure 2 shows the number of newly created objects (nodes and edges) after the crash. Thus the final number of objects in Figure 1 must be added to the results in Figure 2 to get the actual number of objects.

The pre-crash delivery profile had two major dips. The first dip (at about 1200 seconds) was caused by the subject deleting an external entity, an action 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. Retrospectively, the subject can be seen to have met an interface constraint that reduced his productivity in the sense that he was required to redraw several edges. The second major dip (at about 2500 seconds) 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 object) which reduced his productivity in the sense 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 in the sense that the subject had now to re-create the external entity. That there was no separate interface action to remove just an instance of an object again caused the subject difficulties.

On the post-crash delivery profile there was one major dip. The video record showed that this was due to the subject having accidentally performed additional create actions due to CASEMaker having failed to update the display: the subject was forced to delete unwanted objects.

The constraint profile for Subject A (pre-crash) is shown in Figure 3. The y-axis corresponds to particular constraint violations. Due to a problem with recording violations of edge constraints, not all of the points associated with a particular constraint are shown. Dashed lines have been inserted into Figure 3 to show the violations should have continued to be recorded. The vertical line shows the point that corresponds to the delete action that resulted in a sharp drop in object numbers in Figure 1. The constraint profile shows several violated constraints up to this point. Up until this point, the subject had three external entities in the context diagram and had inserted edges between the entities. The pre-delete constraints are all violations of edge constraints that define that there should be no edges between external entities in a context diagram. The delete action which results in the discontinuation of these constraints removes two of these external entities and all of the edges between them, so none of these constraints persist after the delete. Soon after the delete the subject inserts two processes into the context diagram, resulting in constraint labeled 3 (that there should be no more than one process in a context diagram) being violated.

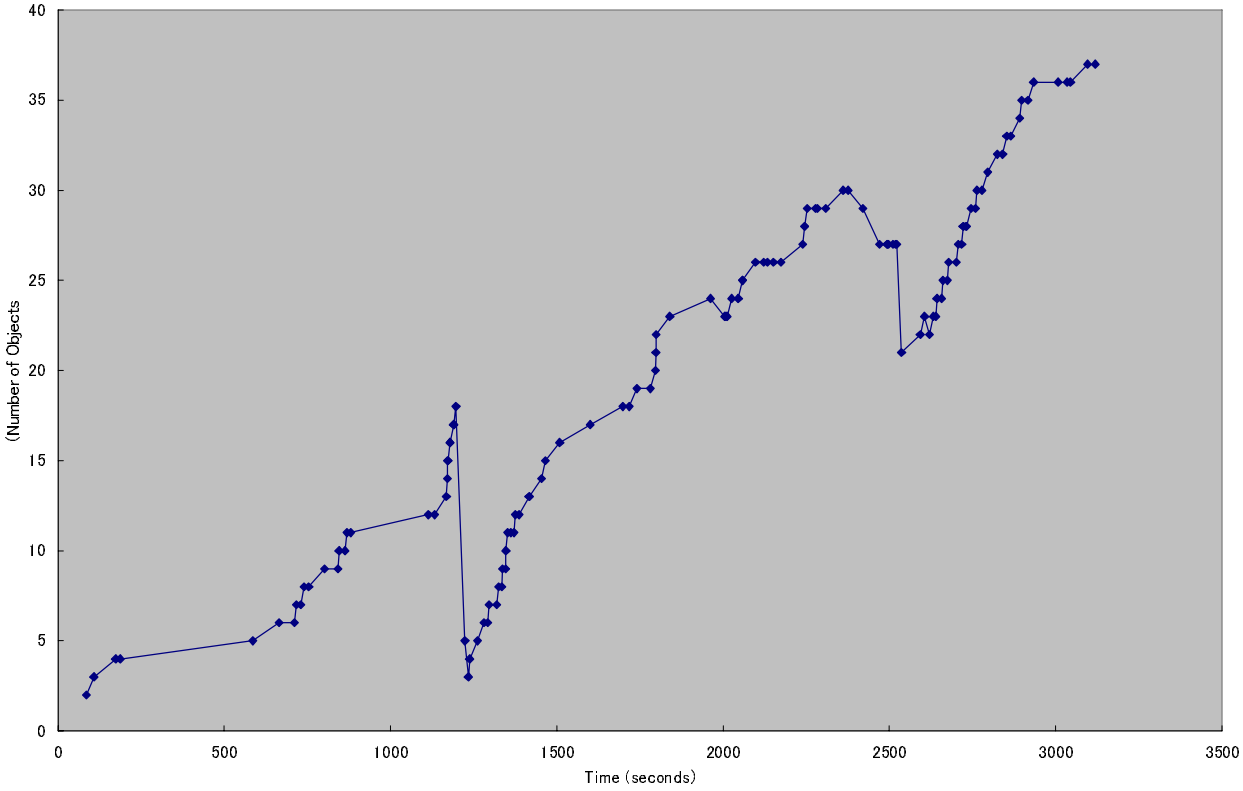


Figure 1. Delivery Profile for Subject A (pre-crash)

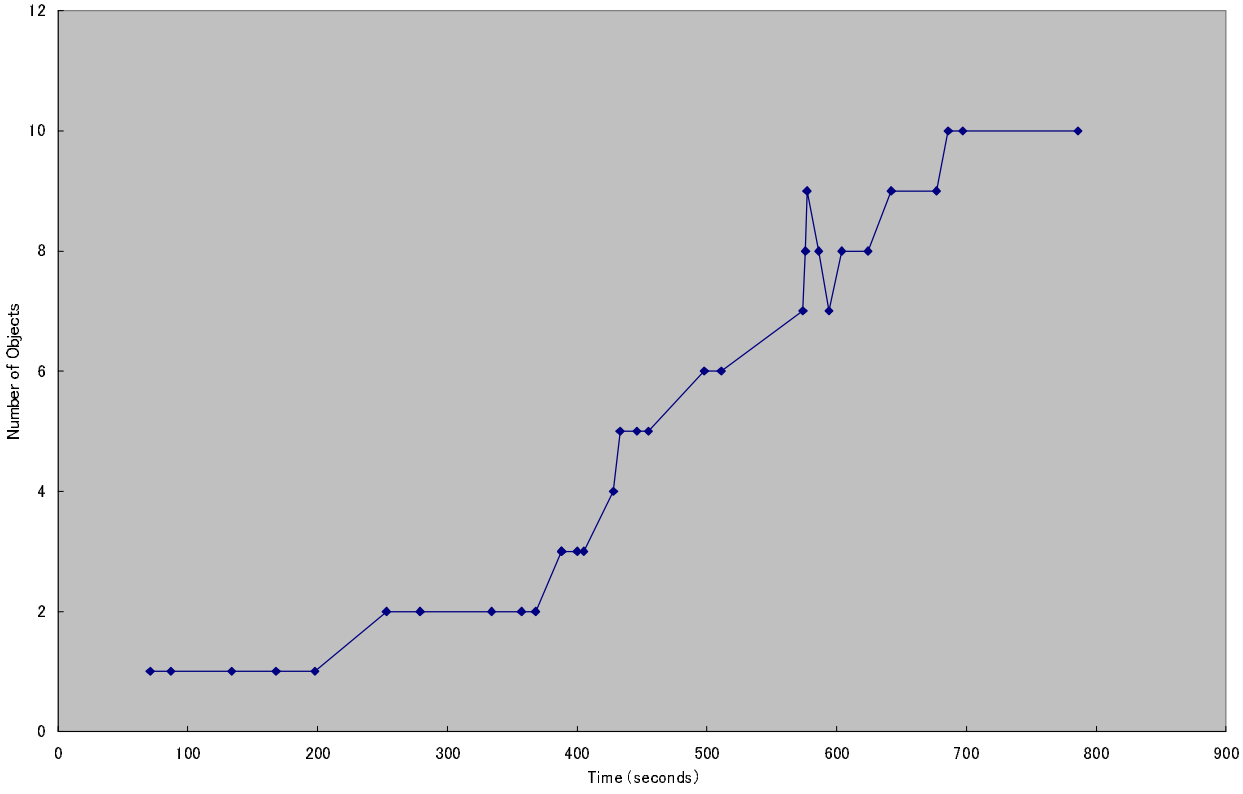


Figure 2. Delivery Profile for Subject A (post-crash)

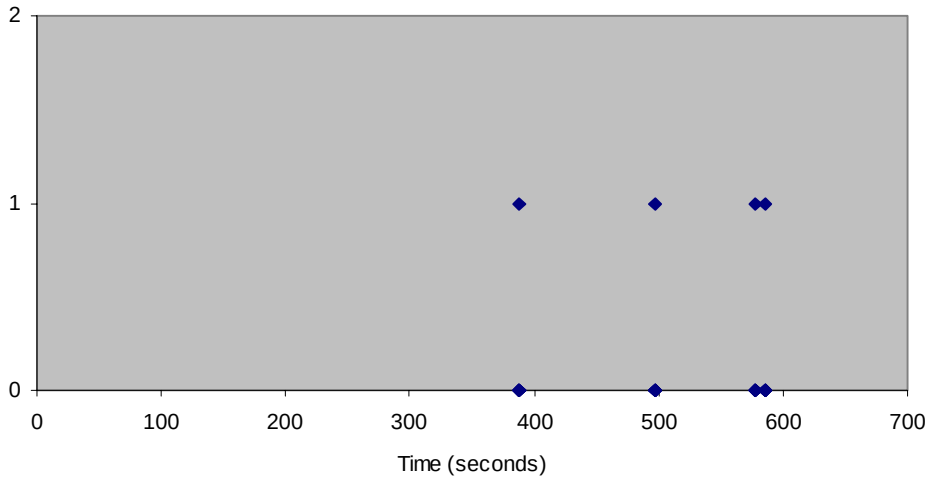
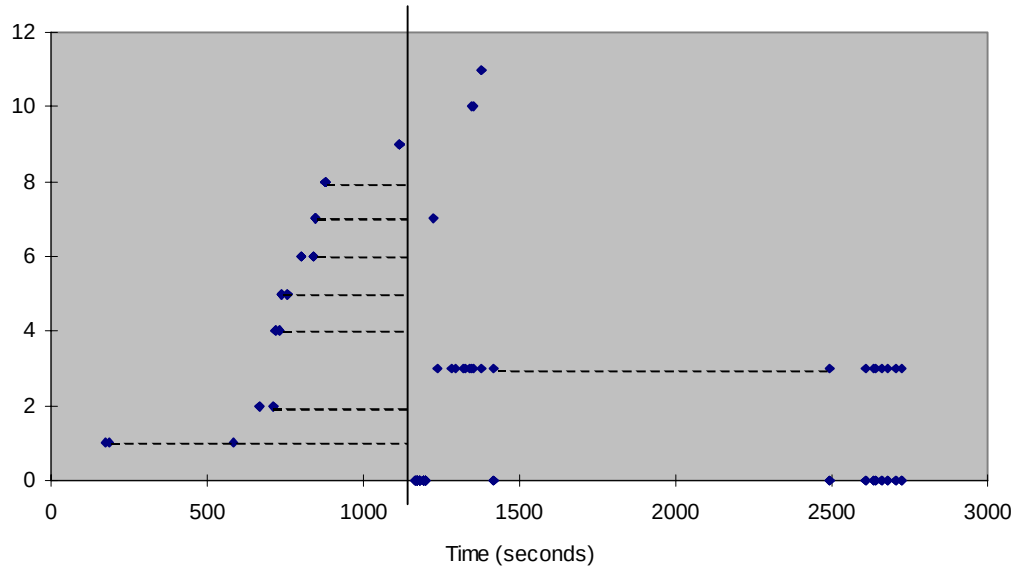


Figure 3. The constraint profile for Subject A (pre-crash).

Figure 4. Constraint profile for Subject A (post-crash)

There are two constraints shown violated in Figure 4. The constraint labeled 0 is a spurious constraint violation triggered by the accidental creation of an edge with the source and target nodes the same. This edge does not appear on the diagram and should not have counted as a constraint violation. The constraint labelled 1 is the constraint that



there are too many process nodes in the context diagram, as noted on Figure 3 which precedes this timeframe.

5.2 Subject B

Subject B suffered one crash but no profile or solution data was lost. The pre-crash delivery profile (Figure 5) showed no unusual activity except when the crash took place. The post-crash delivery profile (Figure 6) proved to have a number of noteworthy features. The video record was examined in detail from about time 1600 seconds onwards. 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. A first interpretation is that 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. A first interpretation is that this reorganizing was done simply to reduce clutter. During debriefing, it was found that this was not the primary reason. Subject B thought that his diagram was probably finished, but he wanted to go over it. For example, he wanted to check if there were any unnecessary flows.

The constraint profile file for subject B (pre-crash) was empty (i.e. no constraints were violated before the crash).

5.3 Subject C

Subject C suffered one crash and the associated profile data was lost. To resume, the investigators had to hand draw the current state of the solution from the video record and the subject proceeded to capture the details in the DFD tool before completing the analysis. The post-crash delivery profile shows one major dip starting at time 1565 seconds (Figure 7). The video record was examined in detail and the investigators viewed the deletions as representative of nothing more than typical modeling activity.

Figure 8 shows the constraint profile for Subject C (post-crash). Constraint number 1 (on the y-axis) is the constraint that context diagrams should not contain data store types and constraint labelled 2 is the constraint that no more than one process should be contained in a context diagram. Subject C violated both of these constraints early in the design, probably due to his lack of experience and knowledge about data flow modelling. The remaining constraints indicate violations in the insertion of edges. Most of these are edges between processes and data stores, which are undefined in context diagrams because data stores should not be present.

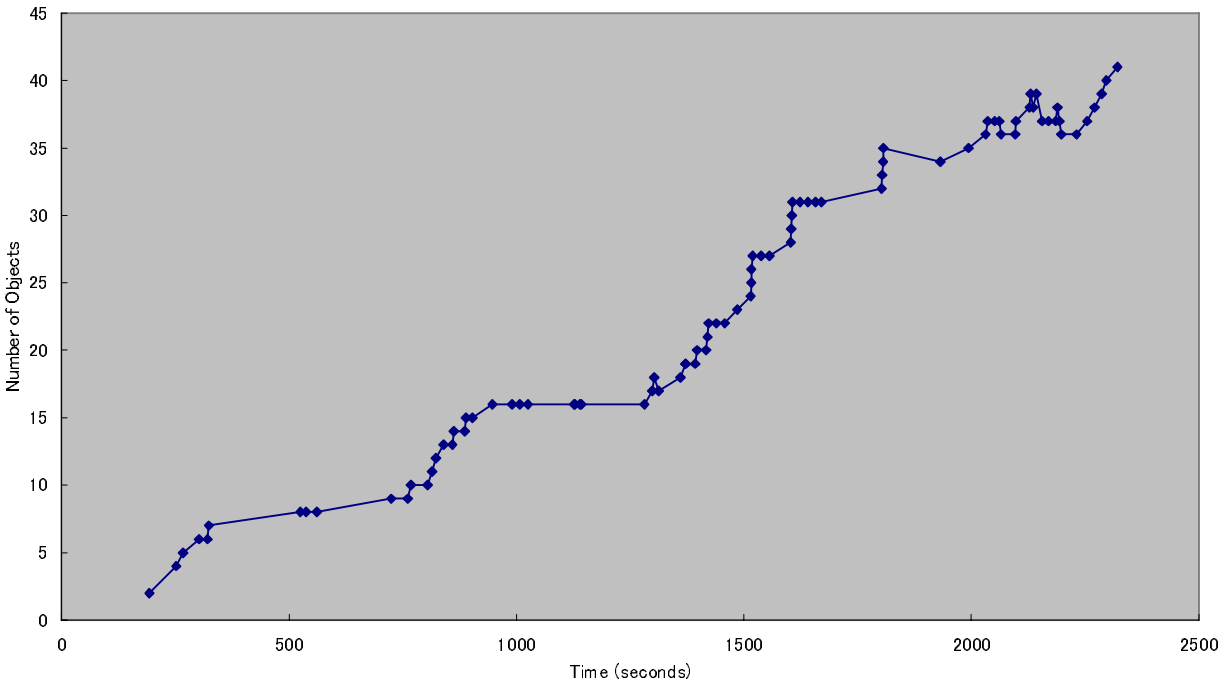


Figure 5. Delivery Profile for Subject B (pre-crash)

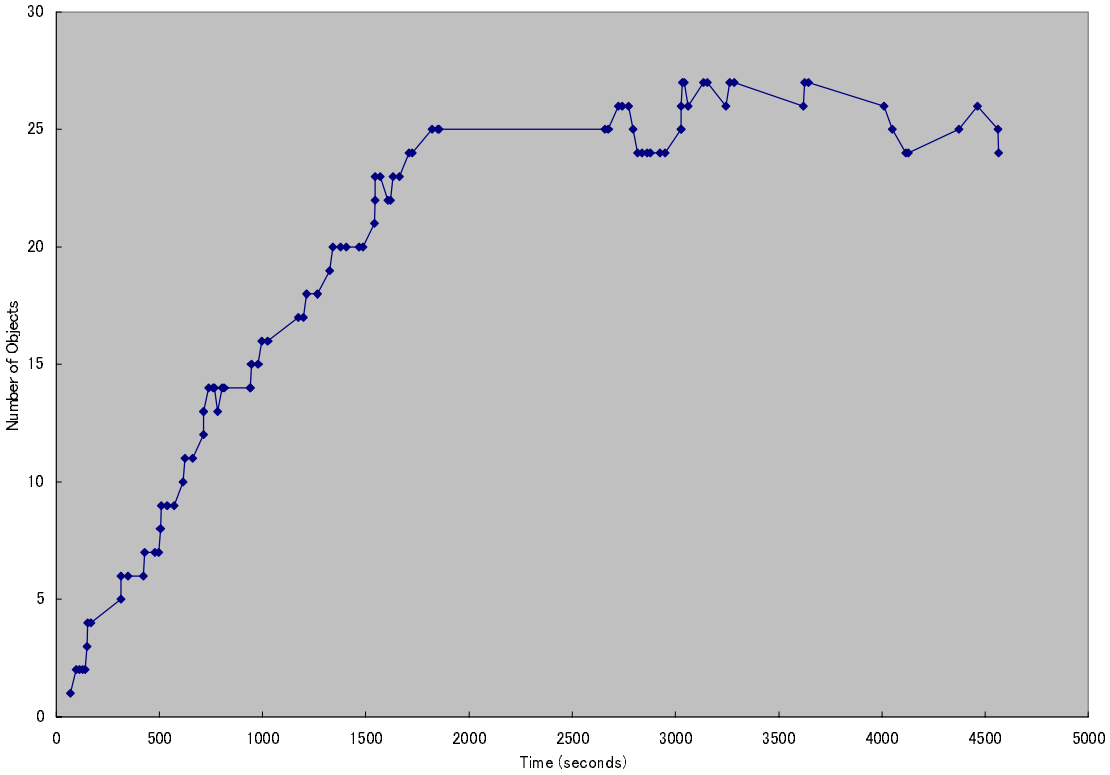


Figure 6. Delivery Profile for Subject B (post-crash)

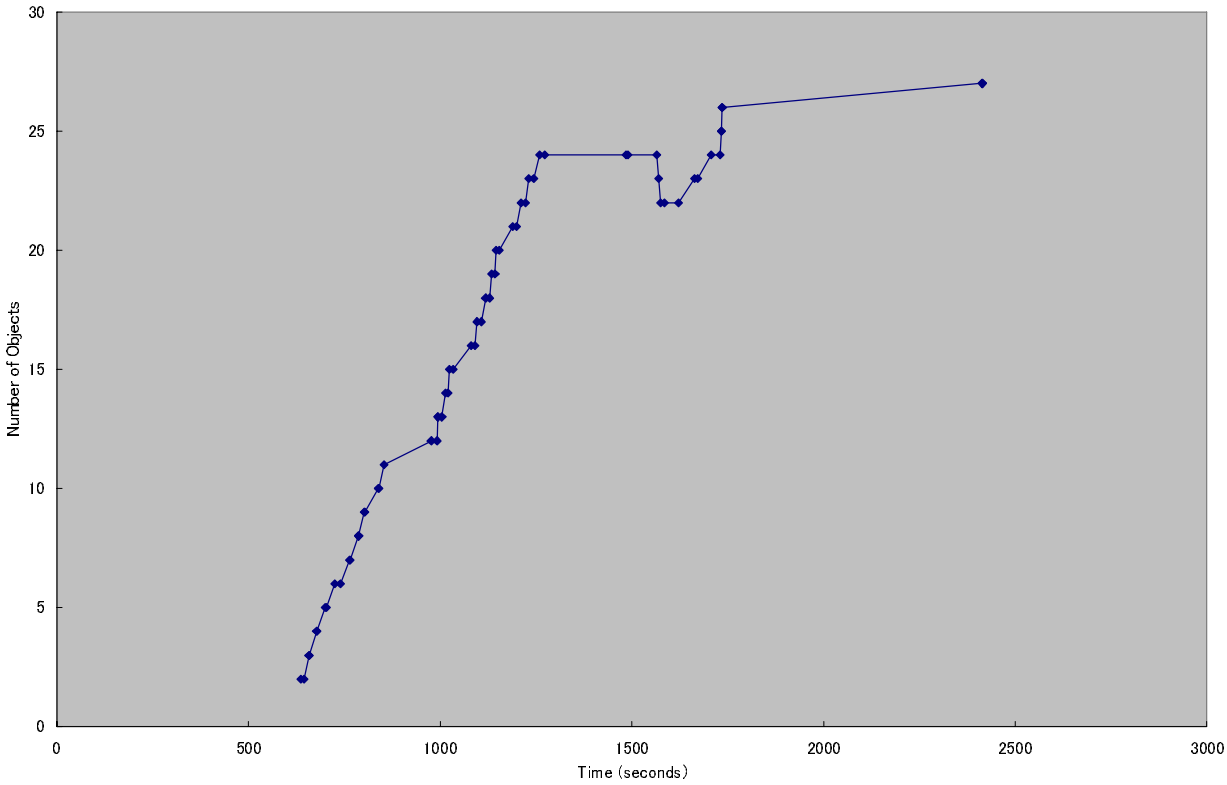


Figure 7. Delivery Profile for Subject C (post-crash)

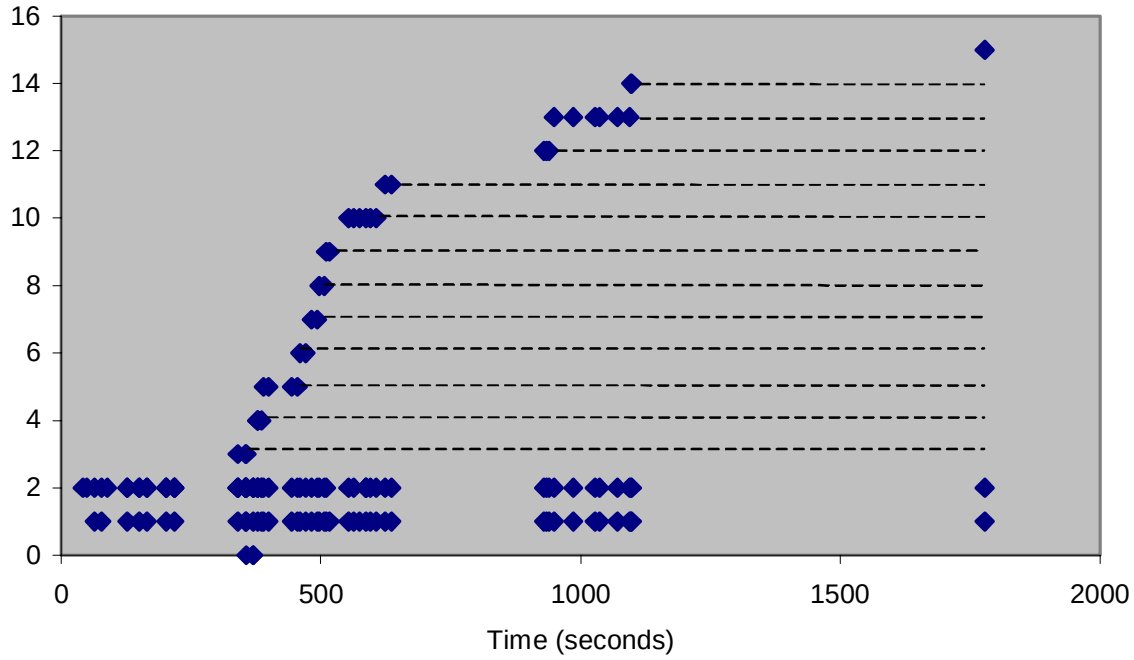


Figure 8. Constraint profile for Subject C (post-crash)

6. CASEMaker Evaluation

Problems were encountered with the installation of CASEMaker on the PC at Nara Institute of Science and Technology. These problems were due to reliance of CASEMaker on Symantec proprietary classes which were not installed on the PC. After trying unsuccessfully to install the required classes, the decision was made to use the tool running on the JRCASE laptop that had all the required packages installed.

Configuration of the CASE tool is given in Appendix F. In general the reliability and usability of the tool were considered good. The tool crashed once for each session. Because the tool was programmed to save activity and constraint profiles every five actions, little data was lost. Some data was lost once because files were not renamed after the crash.

Many functional and user interface fixes were recorded for the tool. Some of the fixes were prompted by observing the subjects using the tool, others by comments the subjects made about the tool in the debriefing session and others by analysing the constraint and activity data that was collected. All of the required fixes are listed in Appendix G. Fixes include modifications to the user interface, adding or modifying functionality, and fixing bugs.

7. Summary and Recommendations

Various influences contributed to a lack of quality in subjects' solutions:

- lack of experience with the modelling technique,
- lack of domain experience,
- misunderstandings caused by the process of translating the English versions of the problem descriptions into Japanese[3],
- subjects considering issues of relevance later in the software life cycle,

[3] Also note that though it did not affect his work, Subject B had commented that one of the sentences in the Japanese was difficult to understand. This was due to one Kanji character being missing.

- a difficulty with English, and
- a lack of a context where analysts are free to ask questions and so clarify system boundaries and problem elements.

Several matters could not be resolved from a quality standpoint given the early stage of analysis. Without video and audio records and subject debriefings incorrect interpretations might have been drawn concerning the quality of subjects' solutions. It is also clear that it is very difficult, if not impossible, to constrain problem descriptions in such a way as to prevent subjects' making unexpected assumptions and to constrain subjects into considering issues purely concerned with analysis. A metric approach to the evaluation of quality is inappropriate for small artifacts created early on in analysis and in the absence of an operational context where analysts are free to ask questions to resolve problem boundaries, etc.

The product delivery and constraint activation profiles hold great promise for the evaluation of productivity under different configurations of the constraint environment. A number of dips in delivery profiles were interpretable as lost time due to interface constraints present in the CASEMaker tool. The influence of interface constraints cannot be ignored. Again, without video and audio records and subject debriefings, incorrect interpretations might have been drawn. It is our judgement that subjects met few, if any, constraints when class modelling.

On the basis of the analyses presented and our own experiences of being involved with Pilot #2, we make the following recommendations.

1. It is unlikely subjects will meet many constraints for class modelling alone. The Automatic Map Labelling problem should be expanded to include the modelling of additional diagram types such as interaction or state diagrams: subjects should meet, and be influenced by, inter-diagram constraints. This will, of course, require subjects to know how to model with the various diagram types employed.
2. In addition to some self-exploration time, subjects' training in any new tool should be more formalized by requiring example diagrams to be entered.
3. The Mail Order System problem should be expanded to incorporate the entry of data dictionary details to resolve any ambiguity in subjects' intentions and interpretations of process, data-flow, and data-store names.
4. An English-Japanese dictionary should be provided to Japanese subjects. Although the original English versions of problem descriptions were handed to subjects, some of the subjects mentioned a difficulty in understanding what a certain Japanese word in the Japanese version corresponded to in the English version. An English-Japanese dictionary should help such situations.
5. It would be useful to manually determine delivery profiles from the videos taken of subjects solving the Automatic Map Labelling problem (tedious, but less so than doing it manually for the DFD problem) to be certain that subjects did not meet any constraints (interface or methodological) when class modelling.
6. Expanded problems might require 3-hour sessions in which event subjects should take breaks to avoid fatigue.
7. There appears to be little point in calculating conventional metrics for either productivity or quality: the artifacts created are small, the assumption problem is likely to persist, and some quality issues simply cannot be judged on with artifacts produced in an early stage of analysis. Instead, emphasis should be placed on interpreting product delivery and constraint profiles. It will be useful to have an object-oriented tool produced using CASEMaker to automate the recording and display of such profiles for object-oriented modelling.
8. Consideration should be given to the use of a maintenance activity. Supplying an existing model should considerably reduce the assumption problem, create a richer constraint environment (i.e. an environment where subjects are more likely to meet, and be influenced by, constraints), and make judging matters of quality more straightforward.
9. The delivery profiles guided the investigators to particular video sequences that were replayed and carefully analysed. Full off-line analysis of all the video records may yield further insights into subjects' behaviour.
10. Delivery profiles should be supplemented with rearrangement profiles i.e. profiles which show the activity of simply moving nodes and edges about the screen in a rearrangement activity. Such activity can be viewed as lost time from a productivity point of view and may reveal the need to provide the user with functionality for automatic layout of diagram elements. (There may be research work here that can be drawn on from the human-computer interaction literature.)
11. Tools generated by CASEMaker should be extensively prototyped to minimise the impact of interface constraints.
12. Instrumenting professional tools for the capture of delivery profiles would allow the capture of real data from real projects albeit within a fixed methodological constraint environment.

8. Conclusions

This second pilot study has shown how difficult it is to prevent subjects from making unexpected assumptions. Many factors were found to influence the quality of subjects' solutions (including misunderstandings caused by the translation to Japanese), and some quality issues could not be judged on with artifacts produced in an early stage of analysis. Video and audio records and subject debriefings were vital: incorrect assessments could have been made about solution quality in their absence. By examining delivery profiles and video records, interface constraints were found to have reduced subjects' productivity on a number of occasions. Analyses of delivery and constraint activation profiles holds great promise for determining the influence of constraints (interface and methodological) on productivity and perhaps quality as well - dips in delivery profiles may represent rework undertaken to improve quality. Consideration has to be given to further changes in the experimental design and the task. A demanding maintenance task taking 2-3 hours making use of a supplied model consisting of a number of diagrams of different types might provide a richer constraint environment that is less exposed to the assumption problem. If the current experimental design were to be repeated, then the instructions and problem descriptions should be revised to try and eliminate any ambiguities.

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 (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.

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. A. Brooks, L. Scott, S. Takada. Evaluating the application of software metrics to data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #1). CAESAR Technical Report 98/9, School of Information Systems, University of New South Wales, 1998.
7. L. Scott. Hypernode model support for software design methodology modelling. JRCASE Technical Report 97/2, School of MPCE, Macquarie University, 1997.
8. M. Page-Jones. The Practical Guide to Structured Systems Design, 2nd Edition. Prentice-Hall, 1988.
9. J Liberty. Beginning Object-Oriented Analysis and Design with C++. Wrox Press, 1998.
10. J. P. Bansler and K Bodker. A Reappraisal of Structured Analysis: Design in an Organizational Context. ACM Transactions on Information Systems, Vol. 11, No. 2, April 1993. pp165-193.
11. J. Dvorak. Conceptual Entropy and Its Effect on Class Hierarchies. IEEE Computer, Vol. 27, No. 6, June 1994, pp59-63

APPENDIX A Problem Descriptions

Mail Order System (DFD1)

INSTRUCTIONS

This exercise involves preparing a logical model of the current system. Do not be concerned about the physical nature of elements or what the logical and physical design should be in future in order, for example, to run the business through the Internet.

Only give names to external entities, processes, data stores, and data flows that you create. Do not write descriptions or specifications for any of these elements. Do not create a data dictionary. Do not model error flows.

Model the system as it is described below - do not assume the existence of any additional external entities or data stores.

Work to produce the best quality analysis that you can.

PROBLEM DESCRIPTION

A mail order company receives orders, by mail or by telephone, from business customers. The mail order company handles invoices, payments, cancellations, and any customer inquiries. Products that have been ordered are sent either directly to the business customer or to the regional offices of the mail order 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 the next few years the mail order company expects to expand and computerize much of its business. You have been called in as an analyst and your first task is to prepare a complete set of logical data-flow diagrams (DFDs) for the current system as described above.

PROBLEM DESCRIPTION IN JAPANESE

通信販売社は、郵便または電話で顧客から注文を受ける。通信販売社は、請求書や支払いの取り扱い、商品のキャンセル、問い合わせに対応する必要がある。発注された商品は顧客に直接送られるか、通信販売社の地域の支店に送られる。支店に送られた場合、その支店が実際に客への配送を行う。通信販売社は、三つの基本的なデータファイルを持っており、それぞれ次の事柄を保持する:客の住所などの情報、商品の在庫情報、請求書番号に基づいた請求書の情報。

次の数年間の間に、通信販売社は拡大し、仕事のほとんどをオンライン化するつもりである。そこであなたはアナリストとして雇われた。最初の仕事は、上記で記されている現在のシステムに対する完全な論理的データフロー図を作成することである。

Mail Order System (DFD2)

INSTRUCTIONS

This exercise involves preparing a logical model of the current system. Do not be concerned about the physical nature of elements or what the logical and physical design should be in future in order, for example, to run the business through the Internet.

Only give names to external entities, processes, data stores, and data flows that you create. Do not write descriptions or specifications for any of these elements. Do not create a data dictionary. Do not model error flows.

Model the system as it is described below - do not assume the existence of any additional external entities.

Work to produce the best quality analysis that you can.

PROBLEM DESCRIPTION

(Same as DFD1 for both English and Japanese)

AML System (Class1 – English)

INSTRUCTIONS

This exercise involves preparing a class diagram of the current system showing association, aggregation, inheritance and relationship cardinality.

Name roles and relationships only when this aids understanding. Attributes and operations should be shown but do not show the basic access operations of set and get. Do not show or edit any default visibilities of attributes and operations. Do not model the user-interface or long term storage and retrieval of maps. Do not show message passing on your class diagram.

Model the system as it is described below. Do not assume the existence of additional classes - do not model low level classes such as those used to represent strings or numbers.

Work to produce the best quality analysis that you can.

PROBLEM DESCRIPTION

The AML system automatically places labels on maps. Maps have two types of features: circle features and rectangular features. Each feature has only one label. Once the cartographer has specified the coordinates and text descriptions of all the features of a map, AML generates default labels depending on feature type. For example, the display text can be an abbreviated form of the text description of a feature. The program then detects overlaps and resolves any resulting conflicts to produce an optimal set of label positions. The user chooses one of three levels of optimisation - the higher the level, the longer it takes AML to produce a labelling solution. Cartographers have complete control over the font, size, colour, background, and display text of a label. For example, the display text can be an abbreviated form of the default text label. After any change, the cartographer can re-run the optimisation process.

During the next year the software company which maintains AML intends a major rewrite from the existing structured language to a leading object-oriented language. You have been called in as an analyst and your first task is to prepare a class diagram for the current system as described above.

PROBLEM DESCRIPTION IN JAPANESE

AML システムは自動的に地図の上にラベルを置く。地図には丸(circle)と長方形(rectangular)の二種類の特徴(feature)がある。各特徴にはラベルがただ一つだけ割り当てられる。地図作成者が地図上のすべての特徴の座標とテキスト記述を指定すると、AMLは特徴の種類により、デフォルトのラベルを作成する。例えば、表示テキストは特徴のテキスト記述の省略形であるかもしれない。それからプログラムは重なり合っているラベルを検出し、それらを解決することにより、ラベルの最適な配置を作り出す。最適化については、ユーザは三つのレベルから選ぶことができる。レベルを高くすればするほど、ラベルの配置を作り出すのに時間がかかる。フォント、大きさ、色、バックグラウンド、ラベルの表示テキストについては、地図作成者が決めることができる。例えば、表示テキストは、デフォルトのテキストラベルの省略にすることができる。地図作成者が何か変化を施した場合、最適化のプロセスを再度行うことができる。

これから一年間、AML の保守を行っているソフトウェア会社は現在の構造化言語からオブジェクト指向言語に書き換えるつもりである。そこで、あなたはアナリストとして雇われ、最初の仕事は上記に記されているシステムのためのクラス図を作成することがである。

AML System (Class2 – English)

INSTRUCTIONS

This exercise involves preparing a class diagram of the current system showing association, aggregation, inheritance and relationship cardinality.

Name roles and relationships only when this aids understanding. Attributes and operations should be shown but do not show the basic operations of set and get or create and destroy. Do not show or edit any default visibilities of attributes and operations. Do not model the user-interface; for example, no display of maps, features or labels. Do not model the long term storage and retrieval of maps. Do not show message passing on your class diagram.

Model the system as it is described below. Do not assume the existence of additional classes - do not model low level classes such as those used to represent strings or numbers.

Work to produce the best quality analysis that you can.

PROBLEM DESCRIPTION

(Same as Class1 for English)

PROBLEM DESCRIPTION IN JAPANESE

AML システムは一つ以上の地図に対して自動的にラベルを配置する。地図には丸(circle)と長方形(rectangular)の二種類の特徴(feature)がある。各特徴にはラベルがただ一つだけ割り当てられる。地図作成者が地図上のすべての特徴の座標とテキスト記述を指定すると、AML は特徴の種類により、デフォルトのラベルを作成する。例えば、表示テキストは特徴のテキスト記述の省略形であるかもしれない。それからプログラムは重なり合っているラベルを検出し、それらを解決することにより、ラベルの最適な配置を作り出す。最適化については、ユーザは三つのレベルから選ぶことができる。レベルを高くすればするほど、ラベルの配置を作り出すのに時間がかかる。フォント、大きさ、色、バックグラウンド、ラベルの表示テキストについては、地図作成者が決めることができる。例えば、表示テキストは、デフォルトのテキストラベルの省略にすることができる。地図作成者が何か変化を施した場合、最適化のプロセスを再度行うことができる。

これから一年間、AML の保守を行っているソフトウェア会社は現在の構造化言語からオブジェクト指向言語に書き換えるつもりである。そこで、あなたはアナリストとして雇われ、最初の仕事は上記に記されているシステムのためのクラス図を作成することである。

APPENDIX B Solutions (Diagrams)

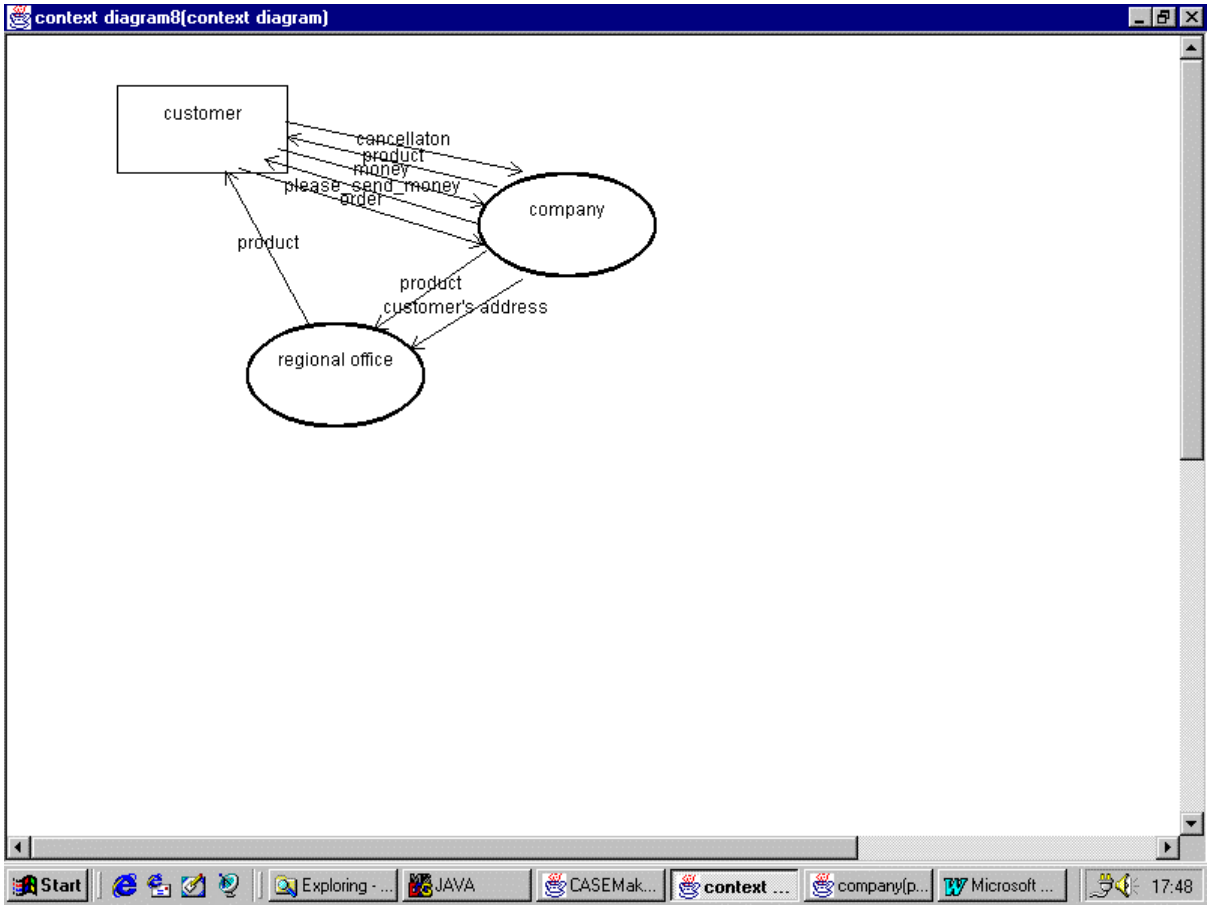


Figure 9. Subject A's DFD context diagram

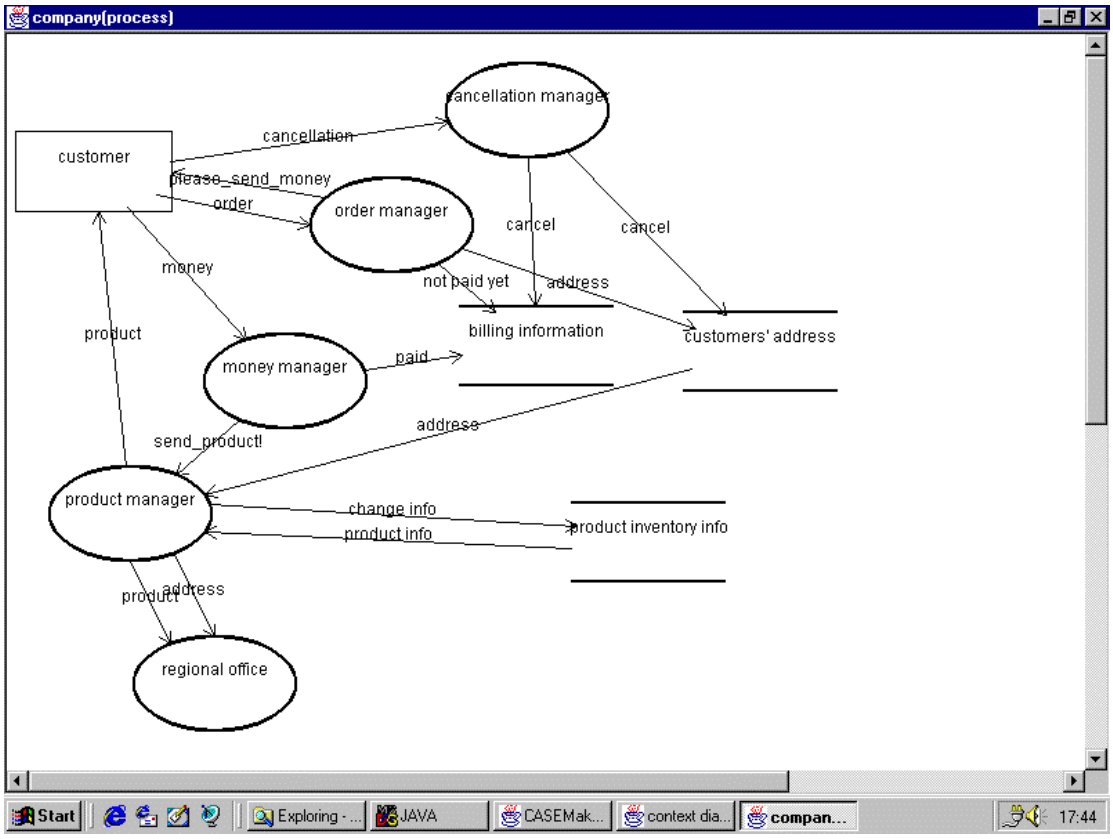


Figure 10. Subject A's DFD Level 0 diagram

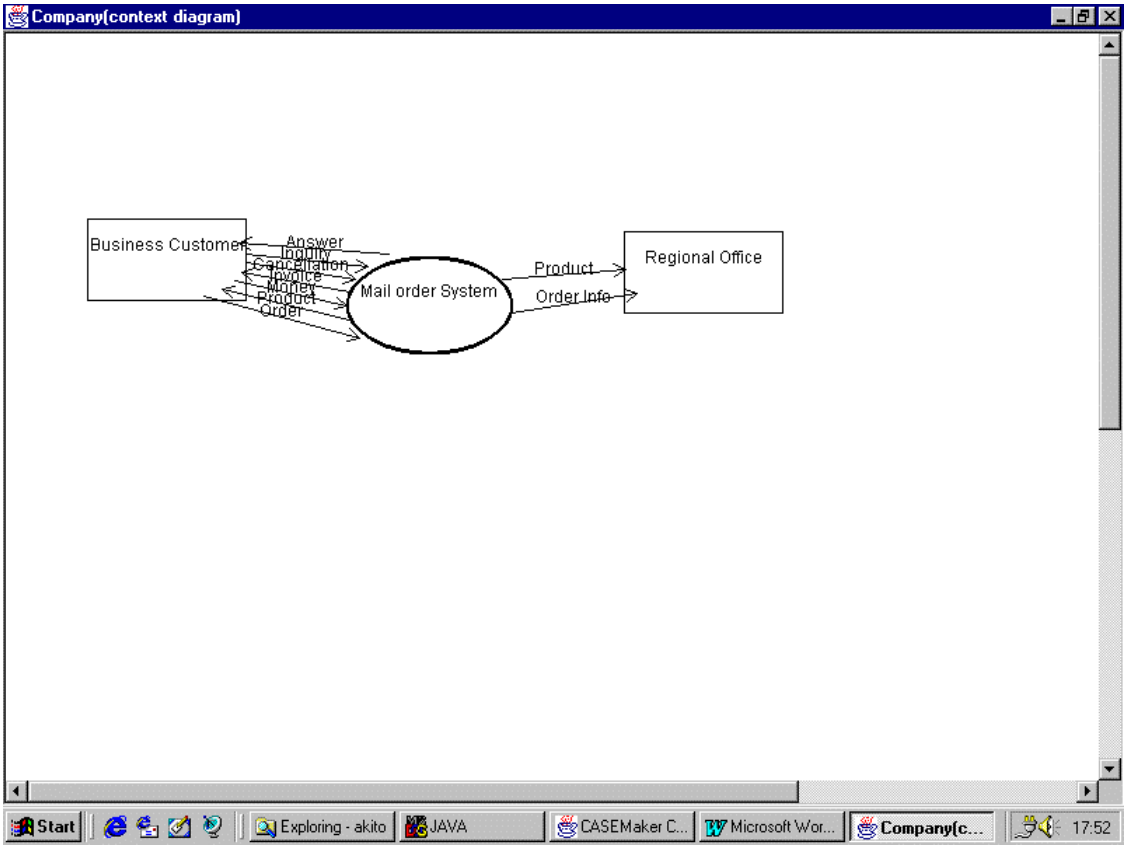


Figure 11. Subject B's context diagram

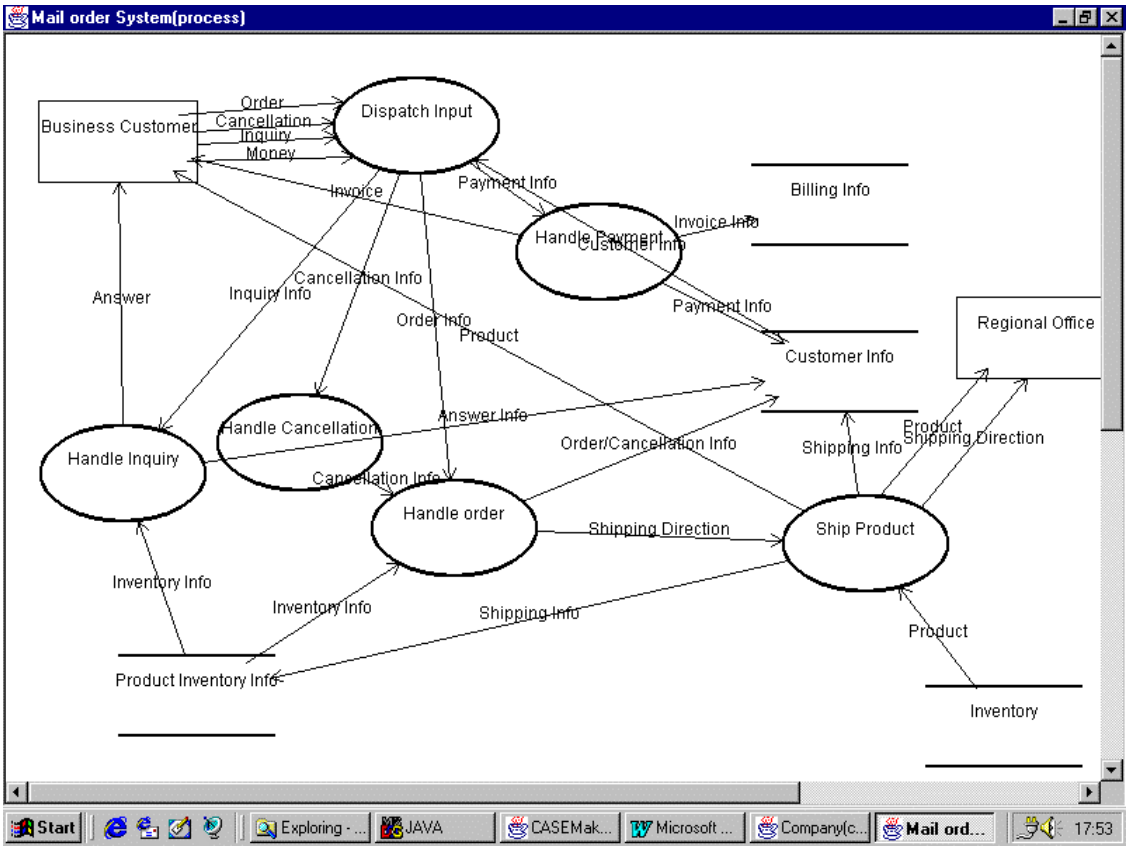


Figure 12. Subject B's Level 0 diagram

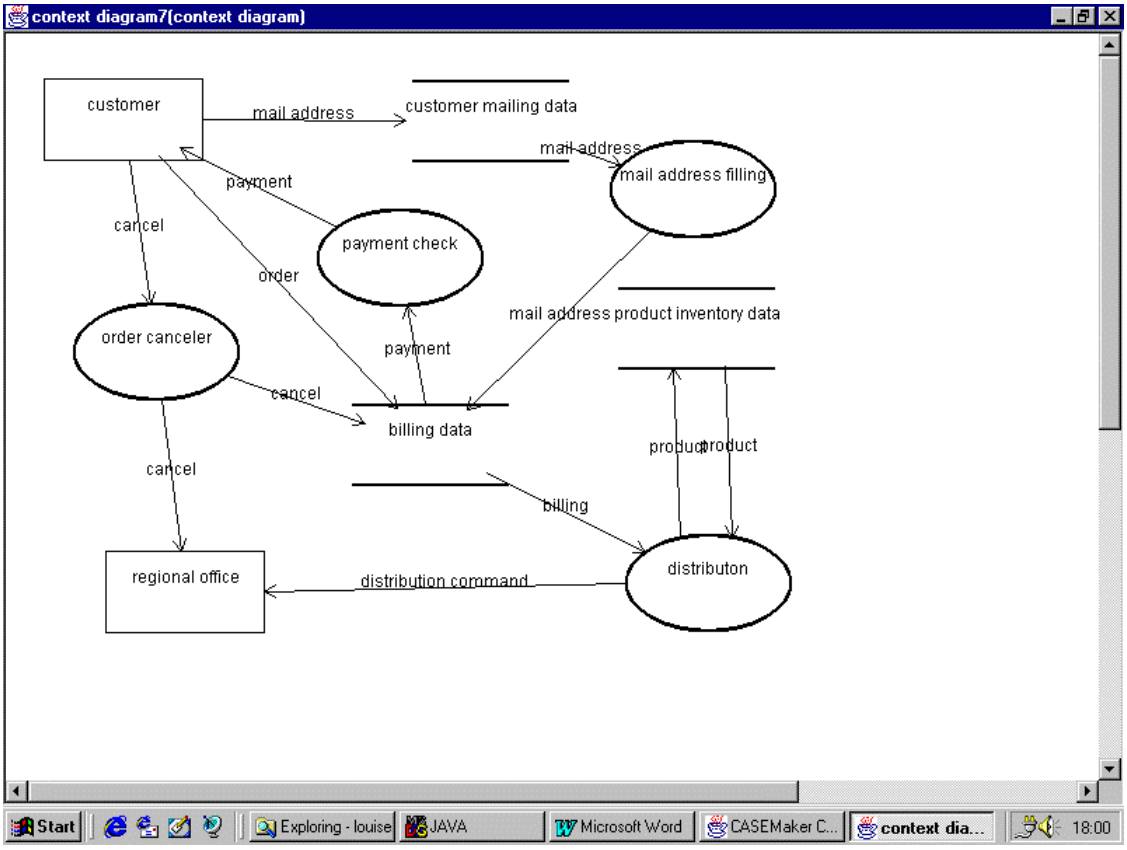


Figure 13. Subject C's Context diagram

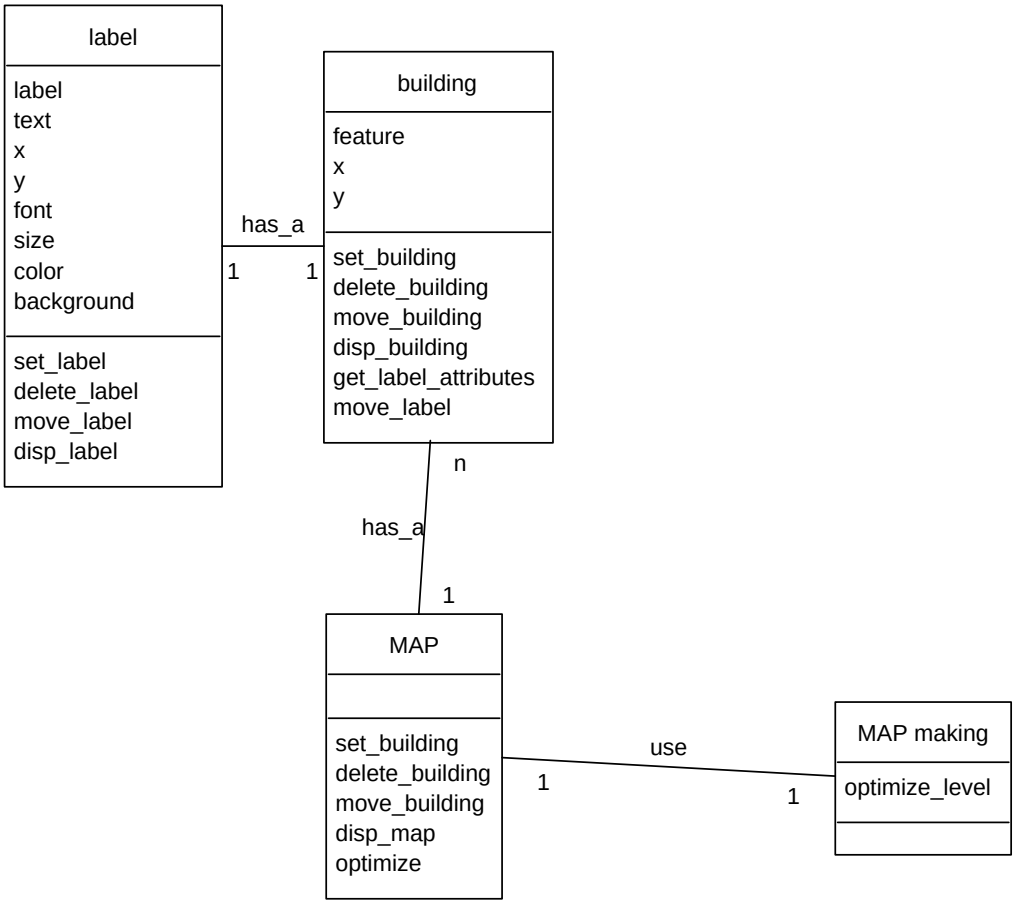


Figure 14. Subject A's Class diagram

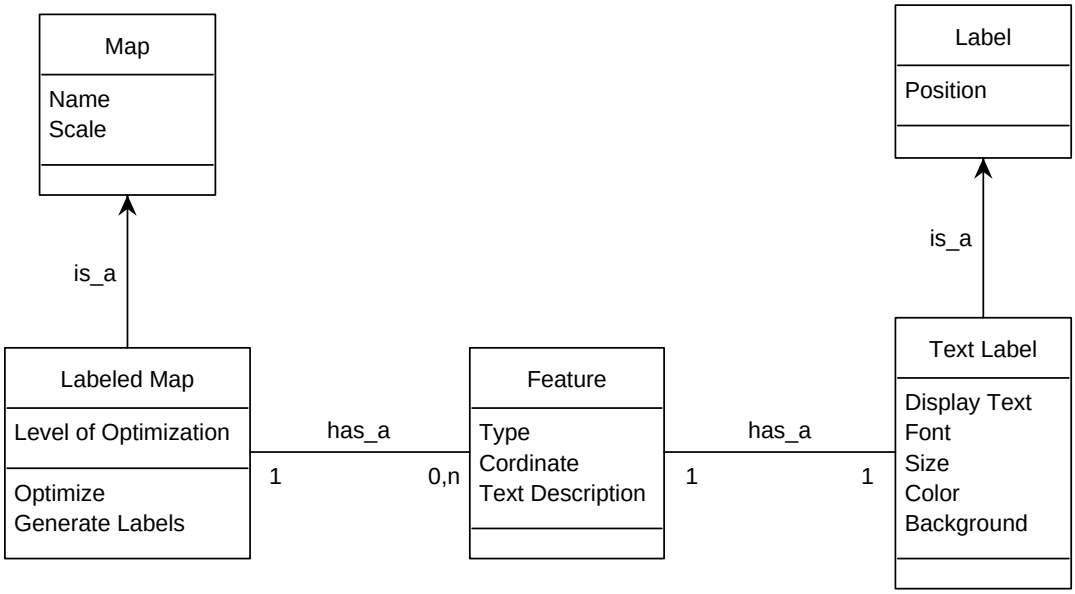


Figure 15. Subject B's Class diagram

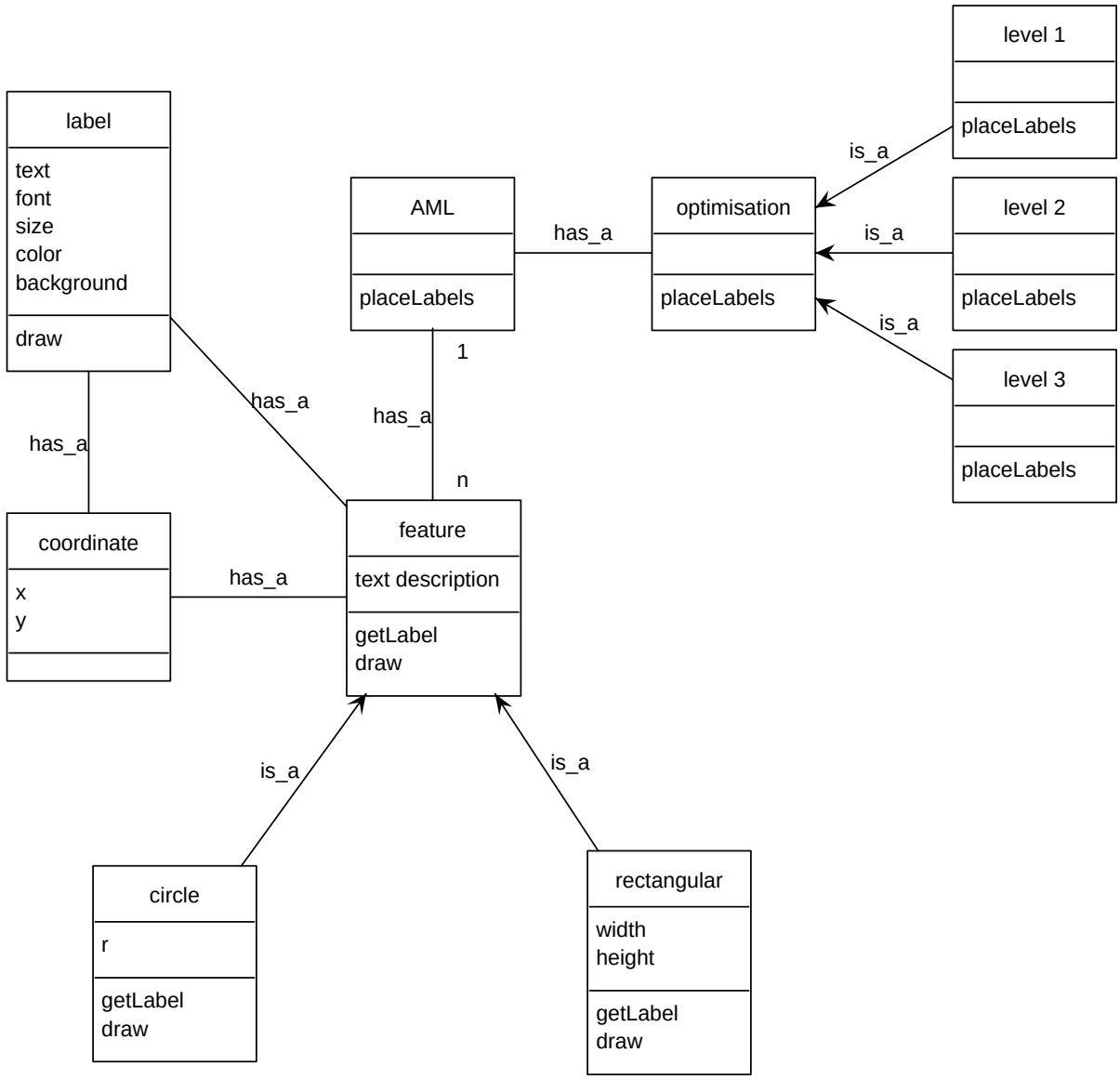


Figure 16. Subject C's Class diagram

APPENDIX C Assumptions

List of encountered assumptions for the Mail Order System

- The customer can place an order using an ID number.
- The regional offices also keep products and the product inventory store in the main office knows inventory information about the regional offices.
- The company is required to keep track of all answers given in response to a customer's inquiries. (Note: this assumption was determined from the video/audio playback.)
- All information concerning the customer is stored along with the mailing information. (Note: this assumption arose because of ambiguity in the Japanese translation.)

List of encountered assumptions for the Automatic Map Labelling System

- Labels can exist on maps by themselves. (The problem description needs modified to read 'labels **on features** on maps'.)
- Maps usually have a name and a scale associated with them so these attributes are required.
- There may be different types of label in future (therefore create a super-class).
- There may be different optimization algorithms in future (therefore create a separate optimization class).

- The levels of optimization correspond to the number of passes through the optimizer (rather than the "neighbourhood tolerance" within which a label is placed).
- Only 'buildings' are labeled. Roads, mountains, rivers, etc are not labeled.

Appendix D Interface Problems

List of interface problems encountered for the Mail Order System problem (CASEMaker)

- Screen refresh was slow and then had to be forced in CASEMaker.
- In CASEMaker, flows could not been dragged about.
- Subjects experienced mode problems i.e. they wanted to move an element but were in create mode. (Possible solution: left mouse button always for move operations?)

List of interface problems encountered for the Automatic Map Labelling System problem (TCM)

- Screen refresh had to be performed manually in TCM after a diagram element had been moved or when a class needed to be resized when several attributes or operations had been entered.
- For TCM, key deletes required used of the left arrow and then <Delete>. <BackSpace> did not work.
- With a two-button mouse, for TCM, both buttons had to be depressed for a dragging operation.
- Subjects experienced mode problems i.e. they wanted to move an element but were in create mode. (Possible solution: left mouse button always for move operations?)

Appendix E CASEMaker training sheet

Using the CASEMaker DFD Tool

To Create a New Diagram
- Select "New Diagram" from the "File" menu
- Select the type of diagram you wish to create
- Select the "Open" button

To Rename a Diagram
- Select "Save Diagram" from the "File" menu
- Type in the name of the diagram and click "OK"

To Create a Design Object
- Select the object on the drawing pallete
- Click once on the drawing canvas

To Create a Design Relationship
- Select the design relationship on the drawing pallete
- Depress the left mouse key over the source object of the relationship
- Drag the mouse until it is positioned over the target object
- Release the left mouse key

To Rename a Design Object or Relationship
- Click once on the object or the relationship with the left mouse button
- Type the name of the object or relationship and click "OK"

OR
- Click once on the object or relationship with the right mouse button to activate the popup menu
- Select "Properties" from the popup menu
- Type the name of the object in the "Name" field of the Properties Box and click "OK"

To Move a Design Object
- Select the "Select Mode" button (the one with the arrow on it)
- Depress the left mouse button over the obect to be moved
- Drag the mouse to the desired position of the object
- Release the left mouse button

To Cut or Copy a Design Object

- Click once on the object with the right mouse button to activate the popup menu
- Select "Cut" or "Copy" from the popup menu
NOTE: Design relationships cannot be cut, copied or pasted

To Paste a Design Object
- Click once on the drawing canvas where the object is to be pasted to activate the popup menu
- Select "Paste" from the popup menu
NOTE: if "Paste" is not highlighted it is because no object has been previously "Cut" or "Copied"

To Delete a Design Object or Relationship
- Click once on the object or relationship with the right mouse button to activate the popup menu
- Select "Delete" from the popup menu

To Set the Properties of a Design Object or Relationship
- Click once on the object or relationship with the right mouse button to activate the popup menu
- Select "Properties" from the popup menu
- Modify the object/relationship name and description as required

To Assign a Number to a "Process" Design Object
- Click once on the object or relationship with the right mouse button to activate the popup menu
- Select "Properties" from the popup menu
- Type the object's number in the "Number" field

To Create a Child "Process" Design Object
- Click once on the object with the right mouse button to activate the popup menu
- Select "Explode" from the popup menu

To Display the Design Checker
- Select "Display" from the "Checker" menu

To Save Your Work
- Select "Save" from the "File" menu

Appendix F CASEMaker Configuration

Two graphs were used to configure the tool for Data Flow Modelling. The graphs are shown in Figure 17.

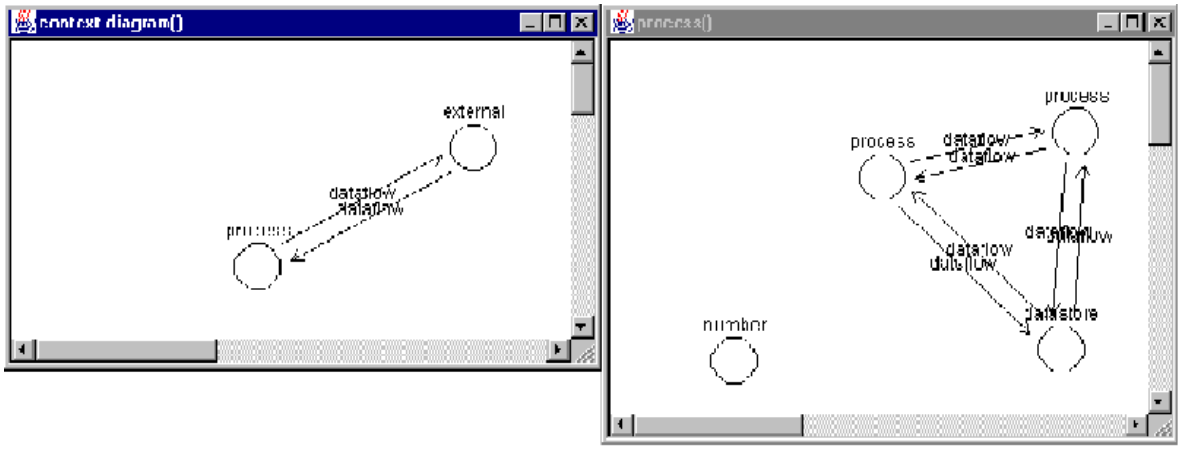


Figure 17. The two graphs used to configure CASEMaker for DFD Modelling

The following settings for notation, UI and constraints were used
context diagram
- no symbol defined
- is a diagram
- must contain "at most one" process

process
- symbol drawer "ProcessDrawer"
- symbol icon "processicon.gif"
- symbol name "process"
- is a diagram
- properties box class "ShowProcessProperties"
- each data store must contain "at least one" dataflow going in (this didn't work)

external
- symbol file "external.gif"
- symbol icon "externalicon.gif"
- symbol name "external"

dataflow
- line drawer "DataFlowDrawer"
- line icon "dataflowicon.gif"
- symbol name "dataflow"

data store
- symbol file "datastorebig.gif"
- symbol icon "datastore.gif"
- symbol name "data store"

number
- symbol file - no symbol
- symbol icon - none
- symbol name "number"

Editor configuration: process, external, data store and dataflow buttons

Ran on Dwarf - pentium 90, 24Mb RAM

Appendix G Fixes required for CASEMaker

Fixes to CASEMaker itself

- disallow accidental edge creation when source and target are the same node (not just named the same but rendered in the same spot) - creates problems with product delivery and constraint activation profiling
- remove the sensitivity of line selection
- provide an “undo” function
- remove the display of initial autonames for nodes and edges (and diagrams)
- improve refresh speed for large diagrams (will improve scrolling capabilities as well)
- provide error messages in Japanese
- improve understandability of constraint violation messages
- reformat constraint violation messages so they can be read by the S-plus statistical package
- provide unique naming of activity and constraint data files
- move dataflows
- copy, cut, paste dataflows
- change “Edge Properties” (the title on the edge properties box) to “*edge type* Properties”
- record modifications made through properties box (recode action interceptions into the "interface" class)
- provide highlighted displays for selected objects
- rethink mode operations
- close windows of deleted graphs
- fix the typechecking bugs
- doesn't detect all edge violations
- didn't detect lack of incoming dataflows to data stores
- fix the data recording
- record finish time as well as start time
- “create diagram” time stamp uses different format
- do differential time stamping as well as absolute
- some constraint data is missing actions (must miss some actions which invoke typechecking)

- don't record accidental edge creation with the same source and target node
- record decomposition actions
- record constraint checker invocation
- remove non-typed graphs from “Open Diagram” under metaCASE functionality
- remove reliance on Symantec classes
- remove “New” tool functionality under metaCASE
- do not display types when editing the node name
- add a UI function to change node types

Fixes to training

- emphasis that to select an edge the mouse must be clicked on the line - not on the name of the edge
- report that it is not possible to move edges
- report that it is not possible to cut, copy or paste edges
- report that there is no “undo” function
- consider providing an example design for them to capture during training

APPENDIX H Sample solutions

Mail Order System

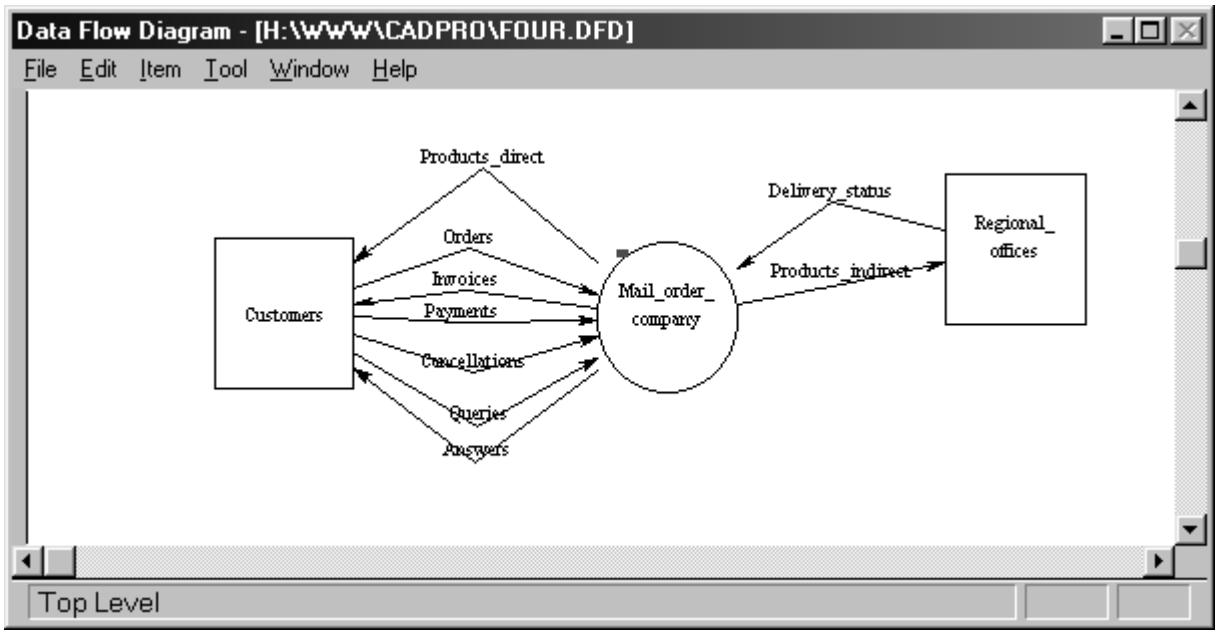


Figure 18. Context Diagram

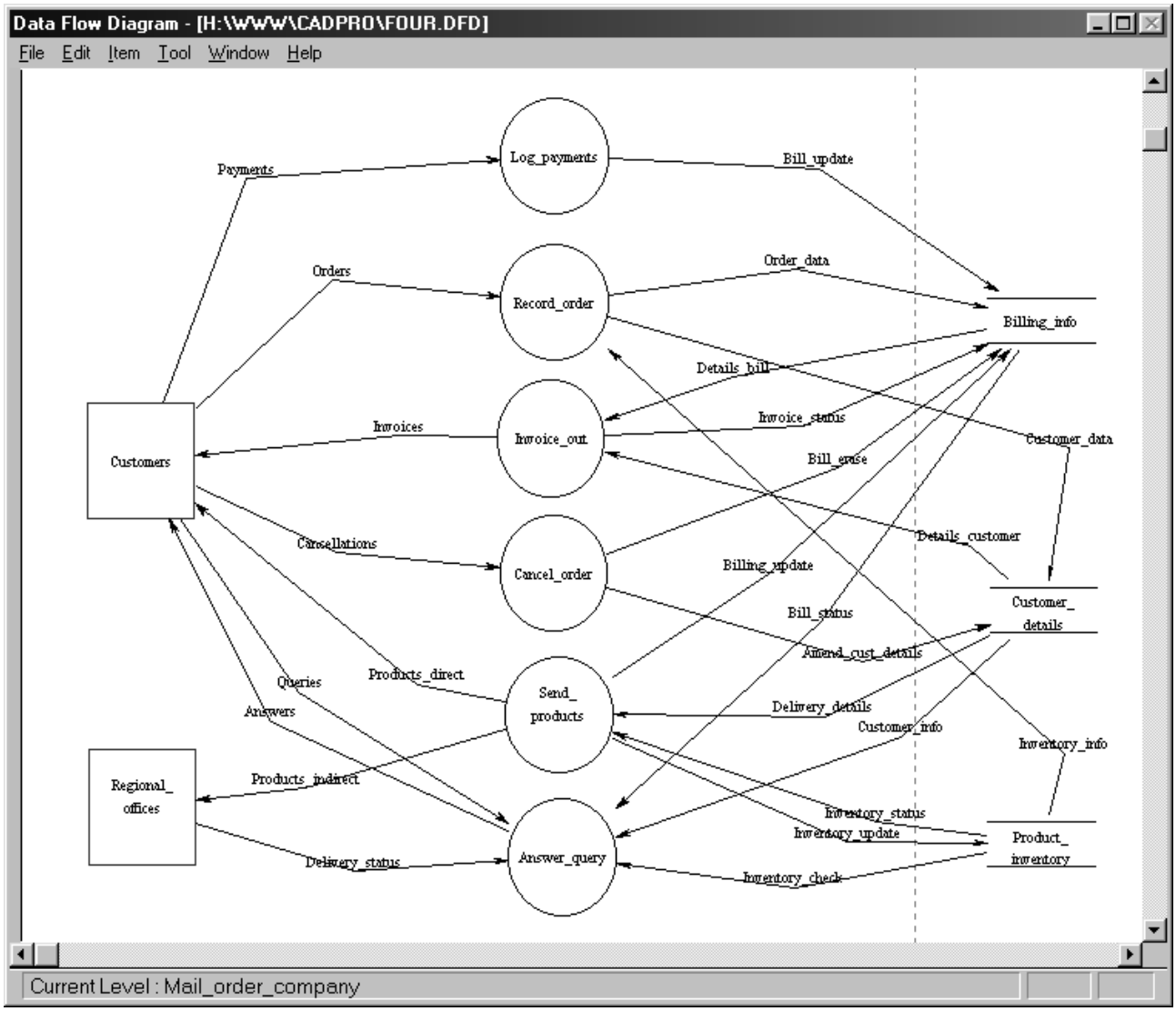


Figure 19. Level 0 Diagram

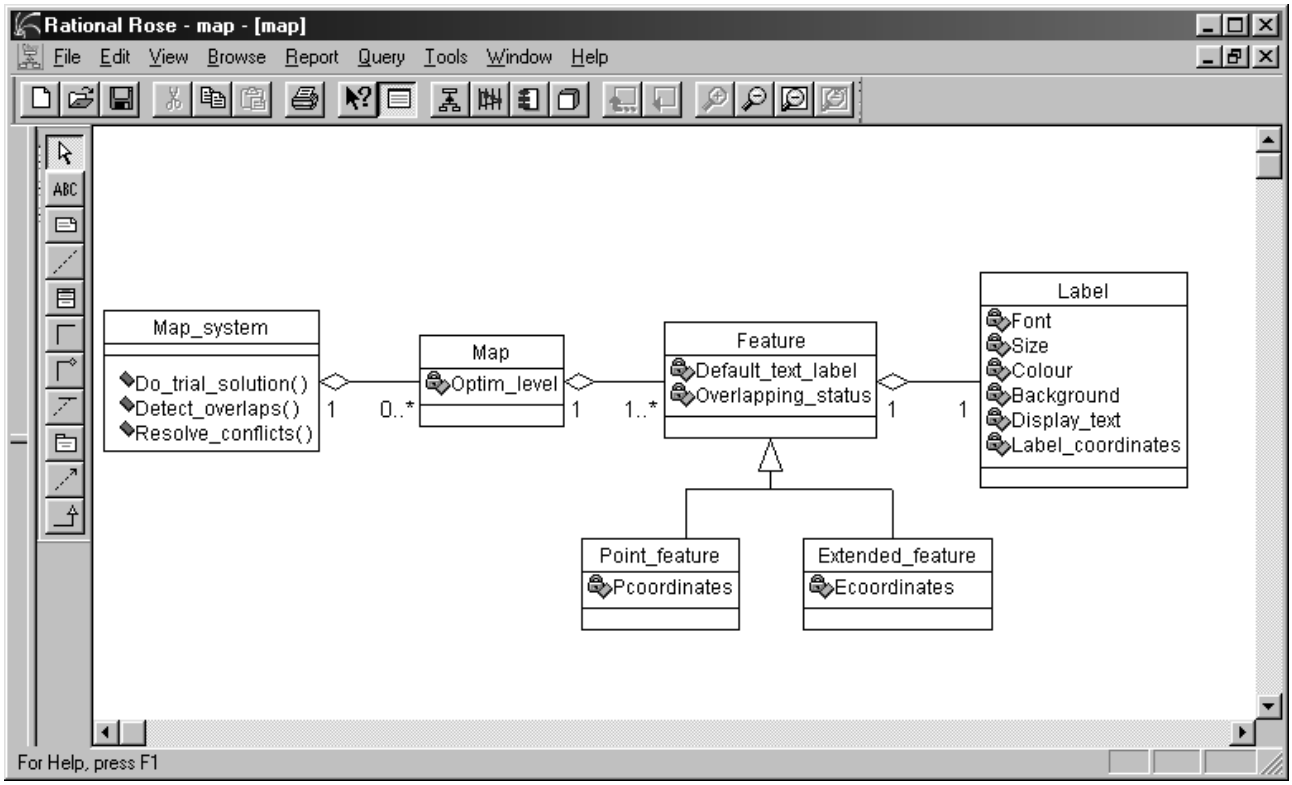


Figure 20. Class diagram