Empirical Software Engineering, 5, 269–285 (2000)

Early Lifecycle Work: Influence of Individual Characteristics, Methodological Constraints, and Interface Constraints

ANDREW BROOKS andy@acm.org
Department of Computer Science, University of Strathclyde, Glasgow, G1 1XH, Scotland, UK

FREDRIK UTBULT
Centre for Advanced Empirical Software Engineering Research (CAESAR), University of New South Wales, Sydney 2052, Australia

CATHERINE MULLIGAN
Centre for Advanced Empirical Software Engineering Research (CAESAR), University of New South Wales, Sydney 2052, Australia

ROSS JEFFERY R.Jeffery@unsw.edu.au
Centre for Advanced Empirical Software Engineering Research (CAESAR), University of New South Wales, Sydney 2052, Australia

Abstract. This paper reports the results of an experiment undertaken for the CADPRO (Constraints And the Decision PROject) project. Subjects with varied experience produced data flow diagrams (DFDs) using a DFD tool generated by CASEMaker, a meta-CASE tool. Half the subjects received routine notice of instances of internal (as opposed to hierarchical) methodological constraint violations via an unobtrusive window whilst the other half did not. The DFD tool automatically recorded subjects' delivery and constraint profiles. Video records, observer notes, and subject debriefings were also used to yield other performance data. While evidence was found in support of the research model underpinning the CADPRO project, the model needs to be revised to take into account the affects of human-computer interface constraints and the different speeds with which people work. We learnt an important lesson about subject randomisation, which is not to assume that all subjects can be treated alike if they share the minimum necessary experience thought required of the problem. We believe it is important for every subject-based experiment to consider and understand the performance of individuals. Because of the complexity of constraint environments in CASE tools we also conclude that studies comparing extreme programming approaches with conventional CASE tool approaches are needed to help determine if the struggle to understand the constraint environment at a high level of abstraction is worthwhile or not. Further experiments, possibly replication variants of this one, are needed to help validate our interpretations.

Keywords: CASE tools, human-computer interface, methodological support

1. Introduction

Methodological constraints are often embodied in computerised tools to help guide activities like analysis and design. For example, a data flow diagramming (DFD) tool may automatically prevent direct links between data stores and a class diagramming tool may automatically prevent cyclic inheritance relationships. Too much enforced guidance, however, can hinder rather than help the software engineer during creative problem solving. Earlier survey work (Day, 1995, 1996) 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. (1997) 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 (1997) outline the CADPRO (Constraints And the Decision PROcess) project which seeks to confirm such effects in a controlled experimental setting. The ultimate goal 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 human-computer interfaces for any computerised tool.

The original plan in the CADPRO project (Day et al., 1997) 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 (Scott, 1997). CASEMaker is sufficiently flexible to allow configuration of the methodological constraint environment. Productivity and quality metrics were to be applied to artifacts created under different configurations of the constraint environment, thus allowing a test of the research model. Pilot studies (Brooks et al., 1998; Takada et al., 1998; Brooks et al., 1999), however, revealed that traditional metrics were not applicable to early lifecycle work. Instead, it was believed analysis should focus on performance profiles i.e. delivery and constraint profiles which chart with time how the work product grows and how instances of methodological constraint violations come and go and persist to the end of a work session.

This paper reports the results of an experiment that involved the automatic capture of such profiles by a DFD tool generated by CASEMaker. Video records, observer notes, subject debriefings, and subjects' DFD solutions were also used to yield other performance data, which aided the interpretation of the performance profiles.

Section 2 provides more background and describes the research model of Day et al. Section 3 describes the experiment and methods used. Section 4 describes the results and the delivery and constraint profiles. The final section draws conclusions. Appendix A lists the problem description and the instructions given to subjects. Appendix B describes the methodological constraint environment of the CASEMaker-generated DFD tool used by subjects. Appendix C provides a result table of the main measures.

2. Research Model

How best to configure constraint environments in computerised tools is far from clear. Even for document production it is far from obvious how best to constrain users so they avoid making spelling and grammatical mistakes. Silver (1988) discusses several objectives favouring greater restrictiveness and several other objectives favouring lesser restrictiveness in decision support tools. All these objectives are broadly applicable to any computerised tool. For CASE, a prescription objective could be the wish to restrict users so that they comply with methodological constraints, a good thing from a software quality viewpoint. To support conceptual design, as opposed to design capture, unconstrained modelling environments may be desirable (Haddley and Sommerville, 1990; Reeves et al., 1995; Jarzabek and Huang, 1998). Placing the constraint environment under user control may be an answer to the configuration problem. Work by Day et al. (1997) suggests that providing alternative

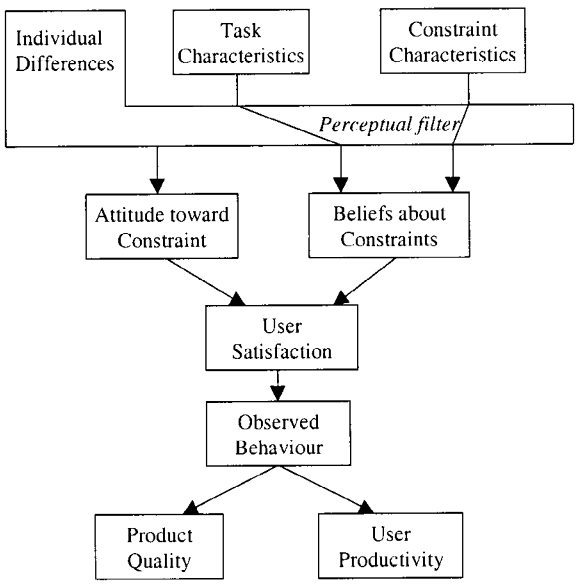

Figure 1. Research model by Day et al.

environments is not sufficient in itself to guarantee success: users' behaviour will also be governed by their perceptions of the constraint environment and also their attitudes toward constraint. Their survey-derived research model is shown in Figure 1 below.

An experimental study (Brooks and Scott, 2000) of individual DFD constraints found laboratory evidence in support of this survey-derived model. For example, having been taught a top-down approach to design, two-fifths of subjects using two CASE tools could not find out how to do designs bottom-up: this would clearly affect behaviour (limiting the design approach) and productivity (if a top-down design requires revision) in a work situation if the faulty belief persisted that doing designs bottom-up was impossible. Evidence from this study suggests that beliefs about constraints can directly impact observed behaviour: the intermediate role of user satisfaction might need to be revised in the research model.

3. Experiment

3.1. Introduction

Features of a usability laboratory were simulated by having each subject perform in one room whilst a video link allowed the investigators to observe what the subject did in a room nearby. Investigators made notes of any difficulty subjects experienced and any periods where the subject re-arranged diagram elements or were deemed to be re-reading the problem description. These observer notes formed the basis of subject debriefings that took place following completion of the problem task. 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, and there was a time limit of 1.5 hours. Use of pen and paper was disallowed: we are interested in studying tool-based constraint environments with the long term goal of building design environments in which users never need to reach out for a pen and some paper even during periods of highly creative design work. Subjects were required to use a DFD tool generated by CASEMaker. Half the subjects received notice of instances of methodological constraint violations via an unobtrusive window whilst the other half did not. The experiment was exploratory: there was no formal null hypothesis to test. The investigators had no preconceived ideas about the shapes of the delivery and constraint profiles though they expected differences in profiles because half the subjects received routine feedback about methodological constraint violation and so might develop different beliefs about, and reactions to, the constraint environment.

3.2. Subjects

Subjects comprised staff and students of the School of Information Systems, Technology and Management at the University of New South Wales. All were volunteers and no incentives were given. Subjects' experience levels with DFD modelling varied considerably from 0.5 a year (or less) up to 15 years. Minimum experience levels equated to having participated in a semester long modelling class whilst subjects with many years of experience typically had extensive commercial experience as well. Whilst the subject pool was heterogeneous, there was an expectation by the investigators that every subject had the minimum necessary experience to be able to develop reasonable solutions. Subjects participated one at a time and subjects were asked not to discuss the problem question with anyone until the experiment was completed.

Subjects were randomly assigned to one of two conditions: half the subjects received notice of a set of internal (as opposed to hierarchical) methodological constraint violations via an unobtrusive window whilst the others did not. At the outset there was an expectation of large numbers volunteering to act as subjects but as the experiment progressed, work pressures meant that many potential subjects withdrew. Data was obtained for only 14 subjects and results for 2 of these 14 subjects were later discarded due to too much incompleteness caused by failures of the CASEMaker-generated DFD tool. This left data only for 12 subjects.[1] Retrospectively, many relatively experienced subjects were found to have been randomly assigned to the condition in which routine notices were given of instances of methodological constraint violations whilst many relatively inexperienced subjects had been assigned to the condition in which no notices were given.

3.3. Problem and Solution Scoring

Subjects were required to perform DFD modelling for a Garden Centre System (see Appendix A) which was regarded as having sufficient breadth and depth so that subjects would produce a variety of solutions. Subjects were given a maximum of 1.5 hours to solve the problem but could stop when they felt they had done enough. (The fastest subject required only 34 minutes.) The Garden Centre System problem was taken from a web site of the University of London and appears to have been used as a 2nd year undergraduate assignment in information systems. Instructions given to subjects are reproduced in Appendix A. Subjects were advised that the quality of their work was not being assessed and that their data would be treated confidentially. The intention was to invoke as natural a response as possible under the artificial condition of being videoed. Alleviating subjects' concerns when under close surveillance is, however, not without penalty: their responses may not be typical of the responses they would give in a commercial setting with a rigorous quality control program.

A sample solution for the problem was drawn up and used as a baseline for scoring the completeness of subjects' solutions. In scoring, a mark of '1' was awarded if an item in the sample solution such as a named process or named data-flow could be matched to the sample solution. Sometimes a half mark was awarded where a name was missing but where the context suggested the subject was indeed modelling processes or flows correctly. Scoring was performed by two of the investigators working together to agree marks. The maximum mark obtainable was 45 based on a sample solution revised and worked on for over 3 hours (i.e. at least twice the length of time given to subjects). The sample DFD solution had a depth of 2 (Context Diagram, Level 0 Diagram, and one Level 1 Diagram) but a solution of depth 1 would have been just as acceptable. Though we had considered a more detailed approach to scoring such as that given in Rehder et al. (1997), we deemed our subject's work products to be 'first-cuts' and that a simple approach to scoring was sufficient. No credit was awarded or taken away based on the nature and degree of process decomposition.

3.4. Training

Prior to using the CASEMaker-generated DFD tool, subjects were given a one-page training sheet containing instructions on how to use the tool and were allowed 10 minutes to familiarise themselves with these instructions and the tool.

3.5. CASEMaker-Generated DFD Tool

The second pilot study (Takada et al., 1998) recommended numerous fixes to the CASEMaker tool. Resources permitted only a few of these fixes to be made. General debugging and testing did, however, result in improvements in screen refreshing and overall reliability. The methodological constraint environment of the CASEMaker-generated DFD tool is described in Appendix B. Using this tool, subjects were free to violate 21 methodological constraint rules with 4 other rules being automatically complied with. Two versions of the tool were produced: one routinely issued notice of instances of 9 types of internal (as opposed to hierarchical) methodological constraint violation via an unobtrusive window whilst the other did not. Note that even as investigators we found considerable effort was required to understand the constraint environment of the generated tool.

The CASEMaker-generated DFD tool recorded each subject's delivery and constraint profiles which chart how the work product grows and how instances of methodological constraint violations come and go and persist to the end of a work session. The number of persisting instances of methodological constraint violations at the end of a work session is one measure of quality. Note that the generated tool did not support (or check for) the balancing of data-flows between levels nor did it incorporate an undo mechanism: lack of these two features can be viewed as human-computer interface constraints as opposed to methodological constraints.

4. Results

4.1. Introduction

In addition to the delivery and constraint profiles obtained for each subject, thirty-four other measures were used to characterise each subject's performance: these measures were derived from video records, observer notes, subject debriefings, and subjects' solutions. A result table of the main nine measures is given in Appendix C.

The investigators' had originally held the belief that all the subjects had the minimum necessary experience to be able to produce reasonable solutions. The results, however, showed considerable variation in key measures such as the solution scores (minimum 5.5, maximum 27), the number of instances of persisting methodological constraints violations recorded by the CASEMaker-generated DFD tool (minimum 0, maximum 25), and the natural time taken by the subjects (minimum 34 minutes, maximum 90 minutes). Multiple scatter plot comparisons of the data gave us cause to consider the majority, if not all, the subjects, each as outliers in their own way. For example: Subject 3 had the highest score and achieved this in the shortest time and Subject 7 worked only on the Context Diagram and did not decompose any process. (It was later found out that this subject's experience of DFD modelling was far from recent.)

Only one scatter plot revealed a pattern: see Figure 2.

A straightforward interpretation of this pattern could have been that experience was the dominant factor regarding the number of instances of persistent methodological constraint violations recorded by the CASEMaker-generated DFD tool. Unfortunately, as Figure 2 shows, the majority of experienced subjects received routine notice of instances of methodological constraint violations whilst the majority of inexperienced subjects did not. This was an unfortunate side effect of the subject randomisation process that assumed all subjects had the minimum necessary experience to be able to produce reasonable solutions. Two experienced subjects who did not received routine notices of violations committed fewer persistent violations than any of the inexperienced subjects: so it is not unreasonable to say that depth of experience has had an influence. But low violation rates for the other experienced subjects may be due to their experience, the fact they received notices of the violations as they occurred via an unobtrusive window, or a combination of these factors. Note that, apart from Subject 8, all the subjects who received routine notice of violations stated that they had acted upon the advice (see the result table in Appendix C). This included Subject 7, but as this subject worked solely using a Context Diagram, 25 violations were recorded at

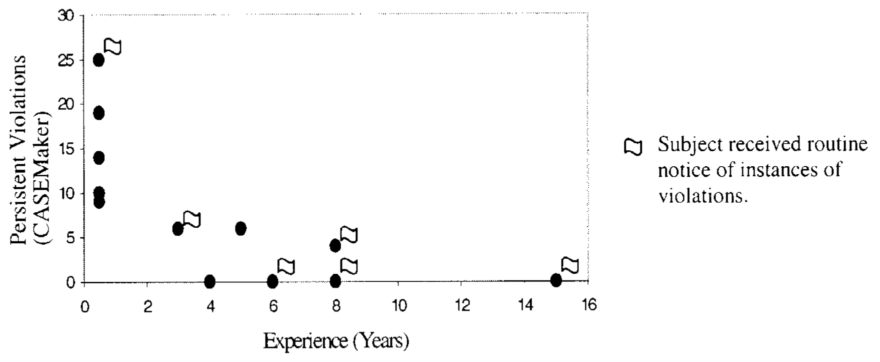

Figure 2. Persistent methodological constraint violations (CASEMaker) versus subject experience.

the end of the session (see Figure 2). Some advice was clearly not acted on by this subject. As noted earlier, this subject's experience of DFD modelling was far from recent.

Aside from Subject 7 who worked solely using a Context Diagram, all the subjects (experienced and inexperienced) committed 2 or more violations of methodological constraints concerned with balancing between levels. (See the result table in Appendix C.) As noted earlier, balancing was not supported by the human-computer interface of the CASEMaker-generated DFD tool: a human-computer interface constraint can be seen to have been responsible for lack of quality in subjects' solutions in this respect.

Despite the variability in this data and the lack of any unambiguous pattern, as will now be seen, consideration of subjects' delivery and constraint profiles led to an understanding of individual behavior. This understanding, in turn, is related to the research model.

4.2. Delivery Profiles

Delivery profiles chart how the work product grows with time. Creations and deletions of nodes (externals, processes, data-stores) and edges (data-flows) were recorded, each creation counting as an increase of one unit of work. Renaming actions were also counted as an increase of one unit of work. Most delivery profiles were regular and could be approximated by straight lines of different gradients. Profiles for Subjects 6, 9, and 11 were not so regular, however, and are shown in Figure 3 contrasted to the delivery profile for Subject 3 (who took the least time and who also had the best score).

The mid-section of the profile for Subject 6 is relatively shallow. It is punctuated by two small dips that were the result of the subject deleting unwanted child graphs. Observer notes taken at the time suggest that this subject took a very casual approach to tackling the problem. Also, about 10% of the time taken by this subject was estimated to have been spent simply rearranging diagram elements. A human-computer interface constraint can be seen to have contributed to the loss of this subject's productivity: the tool had no automatic layout feature and the onus was on subjects to keep their diagrams tidy.

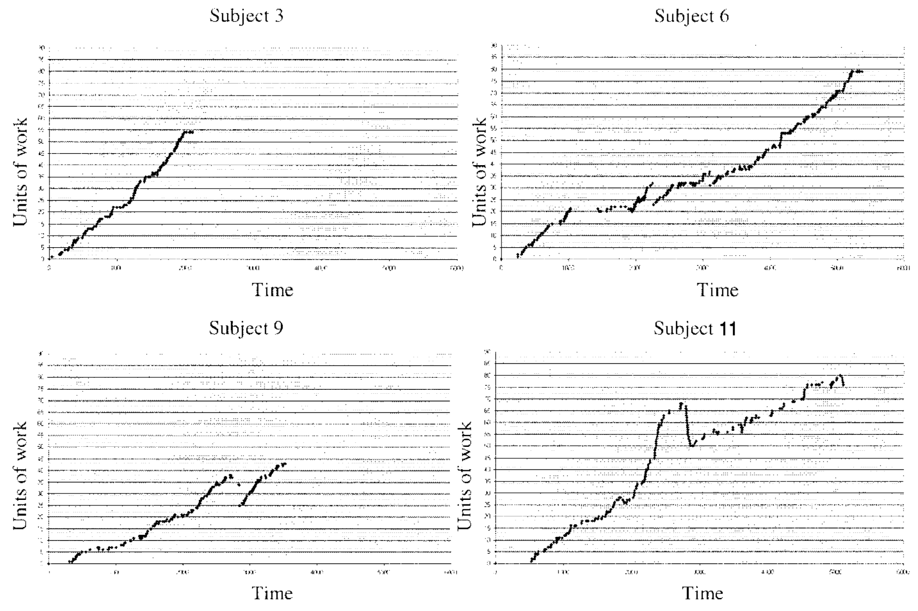

Figure 3. Delivery profiles for Subjects 3, 6, 9, and 11.

The profile for Subject 9 has a large dip: this was caused by accidental deletion of a parent process that resulted in a child graph being deleted. The subject accidentally went ahead with the deletion despite an appropriate warning being given by the DFD tool. As the DFD tool had no undo the subject lost some work product. A human-computer interface constraint can be seen to have been responsible for a loss of this subject's productivity.

The profile for Subject 11, in contrast, has a large hump: this was caused by accidental creation of data-flows when the tool was slow to update the screen. (The problem of screen updating had not been entirely eliminated.) When the unnecessary data-flows finally appeared on the screen, the subject systematically deleted them. Again, a human-computer interface constraint can be seen to have been responsible for a loss of this subject's productivity.

As in the second pilot study (Takada et al., 1998; Brooks et al., 1999), the delivery profiles have revealed the influence of human-computer interface constraints on productivity. The research model needs to be revised to show the contribution of such constraints. The respondents to the survey which was used as a basis for the research model may have based their views as much on the failings of the human-computer interfaces as on the methodological constraint environments: it is not possible to know for certain.

Where delivery profiles were regular, differences occurred in gradient, the rate of growth. So productivity is also determined by speed of working which may relate to inherent differences of abilities between subjects. The research model needs to be revised to show the contribution of this factor.

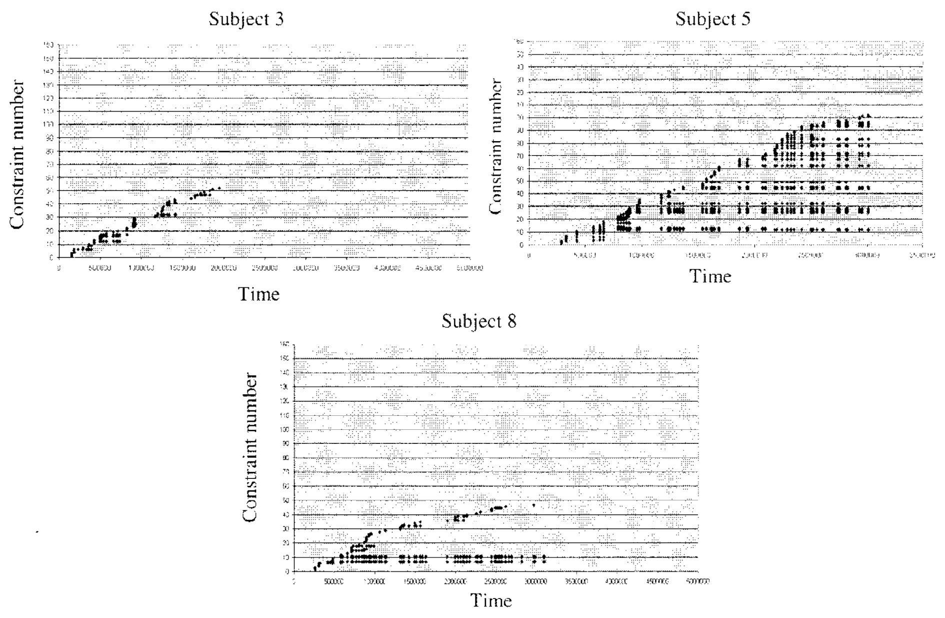

Figure 4. Constraint profiles for Subjects 3, 5, and 8.

4.3. Constraint Profiles

Constraint profiles chart with time how instances of methodological constraint violations come and go, including those violations that persist to the end of a work session. (Note that the numbering on the X-axis is unique to the profile: the same numbers across profiles relate to different violations.) The constraint profiles of the most experienced subjects typically revealed that instances of methodological constraint violations were not allowed to persist for very long.

The constraint profile for Subject 3 is given as a representative example in Figure 4. The profile for Subject 8 was an exception where some violations were allowed to persist until the very end of the session and this too is shown in Figure 4. The constraint profile of Subject 5, a subject who had many violations persisting to the end of the session, is shown for contrast in Figure 4.

Those subjects who had broadly complied with the routine notices about constraint violations can be viewed as having an accepting attitude toward being constrained: they acted on warnings about violations and this meant that the quality of their solution was better (in terms of the methodological rules checked for). This is in keeping with the research model, save perhaps for the intermediate role of user satisfaction.

Subject 8, in complete contrast, however, stated during debriefing that he ignored the routine notices about violations: this subject believed it was acceptable to place data stores on a Context Diagram and so these data-store violations persisted to the end of the session.

The origination and persistence of these violations stand out clearly in the constraint profile for this subject. It would be wrong, however, to attribute this behavior to a difference in attitude toward constraint: in further debriefing it became clear that this subject held the belief that placing data-stores on Context Diagrams was acceptable and that the warning notices issued by the tool were simply wrong. It is possible that this subject may have been exposed to Yourdon's ideas on data-flow diagramming methodology: Yourdon (1989) found it acceptable to have data stores on Context Diagrams indicating the sharing of data between system and terminators. So it was more this subject's belief rather than his attitude that determined his behaviour. This is in keeping with the research model where individual differences can contribute towards beliefs about constraints. In the context of an industrial project where a software engineer might be obliged to rework an analysis then there will be productivity loss where methodological violations take place at high levels of abstraction.

5. Conclusions

We have learnt an important lesson about the randomisation of subjects to separate experimental conditions: don't assume that all subjects can be treated alike if they share the minimum necessary experience thought required of the problem. Recency of experience and depth of experience are contributing determinants of performance. Given the differences in behaviour found, we believe it is important for every subject-based experiment to consider and understand the performance of individuals rather than rely on statistical testing of average trends of a small number of variables.

While evidence has been found in support of the research model, the model needs to be revised to take into account the affects of human-computer interface constraints and the different speeds with which people work. It is perhaps no great surprise to find that the human-computer interface and individuals' characteristics are important determinants of product quality and user productivity. Our experiment, however, has shown how it is possible to measure both the individual (by gradient of delivery profile and constraint profile) and the impact of an individual constraint (by features such as dips in the delivery profile). A Pareto analysis of the impact of individual constraints on productivity and quality could lead to an understanding of how best to design the human-computer interface and the methodological constraint environment of any computerised tool. Further experiments, possibly replication variants of this one, are needed to help validate our interpretations and determine how frequently behaviour like that of Subject 8 occurs.

We found that even as investigators, considerable effort was required to understand the constraint environment of the generated tool—a relatively unsophisticated DFD tool. Productivity loss can occur in the period immediately following the introduction of any computerised tool and we suspect that learning about the constraint environment is a major contributory factor. Object-oriented CASE tools supporting the Unified Modelling Language (UML) present much richer constraint environments with many inter-related diagram types and notations. A question that must be answered is this: is it better to design at a high level of abstraction but struggle to understand the constraint environment or is it better to design by coding and focus on understanding the constraint environment at the level of the code. Studies comparing extreme programming approaches (Beck, 1999) with conventional

CASE tool approaches are needed. Any comparison work must, however, reflect on the user acceptance problems syntax-directed editors had because of the restrictions they imposed on program development (Zelkowitz, 1990; Khawaja and Urban, 1993): the constraint environments of any program editors used must be an integral part of any experimental designs.

Given the lessons learnt about developing and using a research tool solely within an academic environment, and the need to understand more about the constraint environments of commercial tools, plans to develop an object-oriented CASEMaker meta-CASE tool were dropped. Work has since progressed on instrumenting a professional object-oriented CASE tool[2] supporting UML. It is hoped to report on some experiments with this tool in the near future.

For those wishing to attempt a replication study, further details of the experiment and analysis reported in this paper are available as a technical report (Brooks et al., 2000).

Appendix A: Problem Description and Instructions

Problem Description

Dexter Garden Centre

The Dexter family run a small garden centre, selling plants, flowers and gardening supplies to customers in their district. Much of what they sell is grown by them in their greenhouse complex, but such things as gardening implements and special weed killers, fertilisers, etc. are brought in from a selection of different suppliers. (Often also, the seeds from which the stock is to be grown may be purchased in.)

The most difficult task in running the garden centre is to plan and decided what is to be put on sale at what time of the year. Ideally, plants should be sold when they are either just coming into bloom, or are about to commence their growing cycle, and there are a number of standard patterns and schedule guides to help the Dexter's in making these decisions. However the sudden changes in weather conditions and the seasonal swings in temperature make such forecasting very difficult.

Once seeds have been planted, they must be very carefully nurtured and controlled so that whatever the weather, the plants into which they are to grow will be ready at the right moment for sale. Again, this maintenance of growing stock (feeding and watering, adjusting greenhouse temperature, etc.) is done to a pattern based on a standard manual of guidelines built from the experience of many years.

Customers are encouraged to browse through the garden area where the plants are displayed and through the shopping arcade where the shelves are stocked with tools and materials, assembling their order on a trolley then paying for everything at the counter by the exit.

The high-risk nature of the business means that the Dexters' have to manage their finances extremely carefully, balancing expected income against the expenditure, and basing purchasing decisions on the availability of funds generated from recent sales. The need to pay salaries to the employees, and to cope with the tax demands from various government sources both add to the complexity, and therefore to the difficulty, of achieving the required balance.

Instructions Provided to the Subjects

The purpose of this exercise is to produce a DFD that closely follows the standard guidelines for creating a DFD.

The design problem will be collected at the end of this experiment. Please refrain from discussing the contents of this problem with other members of the department in order to maintain internal validity.

You have up to 90 minutes to complete this DFD.

We are not assessing the quality of the design product in this experiment.[3]

All results will be confidential and the subjects will remain anonymous, being represented by a subject number, not their names.

Appendix B

Methodological Constraint Environment of CASEMaker-Generated DFD Tool

Methodological Constraint Description	Jankowski	Free?	Checked?
Internal consistency			
DFD			
At least one process	J1	Free	1
no more than seven processes	J2	Free	0
More than one process in Level 0 and lower diagrams	B1	Free	1
Context			
must exist	J3	Auto	−1
must contain only one process	J4	Free	0
One input from an external entity and one output to an external	J5	Free	1
No datastore on context diagram	B2	Free	1
No data flows between processes	B3	Free	1
Process			
The process must be numbered 0	J6	Auto	−1
At least one input dataflow and one output dataflow	J7	Free	1
Must be connected to a datastore, process or external	J8	Free	CE
Labelled	J9	Free	0
Unique Label	B4	Auto	−1
External Entity			
Must only appear on the context diagram	J10	NA	−1
must be connected to a process	J11	Free	1
must be labelled	J12	Free	0
DATAFLOW			
Must be an interface between a process and either a second process, datastore, or an external entity	J13	Free	CE
A dataflow from a datastore must have a composition that is a subset of the datastores composition	J14	NA	−1
Must be labelled	J15	Free	0
Unique Label	B5	Free	1
Dataflows should not be used to connect to a datastore on a context diagram	B6	Free	1

(*continued*)

Methodological Constraint Description	Jankowski	Free?	Checked?
Datastore			
Must be an interface between two processes	J16	Free	0
Must be labelled	J17	Free	0
Hierarchical consistency			
DFD			
A parent diagram must exist unless it is a context diagram	J18	Free	0
process			
Must decompose to either another diagram or a process specification	J19	NA	−1
Must be numbered with respect to its parent	J20	Auto	−1
DATAFLOW			
An input/output dataflow on a parent diagram must appear on the child diagram as input/output	J21	Free	0
An input/output dataflow on a child diagram must appear on the parent diagram as input/output	J22	Free	0
A set of dataflows on a child diagram that were split from a dataflow on a parent diagram must match the parent	J23	NA	−1
A dataflow must decompose to either a record definition or an element definition	J24	NA	−1
DATASTORE			
A datastore must decompose to either a file definition or a record definition	J25	NA	−1
Process Specification			
All inputs and outputs must match those of the corresponding primitive process on the dataflow diagram	J26	NA	−1
Must be labelled with the same identifier as the corresponding primitive process on the dataflow diagram	J27	NA	−1

Explanations
(i) Rule numbers (J1–J27) are taken from Jankowski (1997): rule numbers (B1–B6) are assigned for completeness of reference.
(ii) ‘Free’ means a subject was free to violate a methodological rule. ‘NA’ means the rule was not applicable in the context of the experimental study. ‘Auto’ means the DFD tool enforced compliance.
(iii) ‘1’ means the DFD tool checked for violation of the rule (and so could warn a subject). ‘0’ means the DFD tool did not check for violation of the rule. ‘−1’ means the DFD tool either automatically enforced the rule or the rule was not applicable in the context of the experimental study. ‘CE’ means the rule was considered to have already been checked elsewhere by another rule or rules.

Appendix C

Result Table

Subject	Experience	Notice	Score	Time	Depth	Persistent Constraint Violations			Use of
	(years)		(max. 45)	(seconds)		Balancing	All Other	CASEMaker	Notice
1	0.5	0	23	2964	2	12	16	9	na
2	8	1	18.5	3004	1	8	9	0	1
3	6	1	27	2041	1	7	8	0	1

(*continued*)

Subject	Experience	Notice	Score	Time	Depth	Persistent Constraint Violations			Use of
	(years)		(max. 45)	(seconds)		Balancing	All Other	CASEMaker	Notice
4	0.5	0	15.5	3294	1	4	10	10	na
5	0.5	0	5.5	3079	2	6	49	19	na
6	3	1	11	5363	2	15	21	6	1
7	0.5	1	13	2542	0	0	45	25	1
8	8	1	18	3238	1	2	4	4	0
9	5	0	26	3525	1	3	5	6	na
11	4	0	17	5102	1	4	45	0	na
12	0.5	0	11.5	2973	1	4	5	14	na
14	15	1	22	4464	2	15	15	0	2

Explanations
(i) The ‘Notice’ column indicates which subjects received routine notice of instances of methodological constraint violations (1) and those that did not (0).
(ii) The ‘Balancing’ column indicates the number of instances of persisting methodological constraint violations for balancing rules J21 and J22 (determined manually).
(iii) The ‘All Other’ column indicates the number of instances of persisting methodological constraint violations for all other rules not automatically recorded by the CASEMaker-generated DFD tool (determined manually).
(iv) The ‘CASEMaker’ column indicates the number of instances of persisting methodological constraint violations for 9 internal rules automatically recorded by the CASEMaker-generated DFD tool.
(v) The ‘Use of Notice’ column indicates what subjects stated during debriefing about how they had made use of the screen window which issued routine notice of instances of methodological constraint violations:

na	not applicable
0	ignored
1	used intermittently
2	acted on frequently

Acknowledgments

We thank the referees and conference attendees for their constructive comments. 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, Macquarie University, and the University of Strathclyde. Thanks to Graham Low for assistance in setting up the experiment.

Notes

1. With only six subjects in each condition, retrospective hypothesis testing is also inappropriate.
2. Simply Objects from Adaptive Arts
3. This was stated to alleviate subjects’ concerns of being assessed: the investigators were interested in quality as measured by the number of persisting methodological constraint violations at the end of each session.

References

Beck, K. 1999. Embracing change with extreme programming. *Computer* 32(10): 70–77.

Brooks, A. and Scott, L. 2000. User perceptions of constraints in CASE tools. Submitted to the *International Journal of Human-Computer Studies*.

Brooks, A., Scott, L. and Takada, S. 1998. 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.

Brooks, A., Takada, S. and Scott, L. 1999. Strongly formative pilot studies on constraints in early life-cycle work. *Proceedings of the Sixth Asia-Pacific Software Engineering Conference (APSEC '99)*, 614–621. IEEE Computer Society.

Brooks, A., Utbult, F., Mulligan, C. and Jeffery, R. 2000. Early lifecycle work: influence of individual characteristics, methodological constraints, and interface constraints. Centre for Advanced Empirical Software Research (CAESAR) Technical Report 00/9. School of Information Systems, Technology and Management, University of New South Wales. (http://www.caesar.unsw.EDU.AU/CAESAR.html)

Day, D. 1995. User responses to constraints in computerized design tools. Ph.D. thesis, School of Information Studies, Syracuse University. UMI Order Number 95-44905.

Day, D. 1996. User responses to constraints in computerised design tools: An extended abstract. *ACM Software Engineering Notes* 21(5): 47–50.

Day, D., Ahuja, M. and Scott, L. 1997. Constraints in design engineering: a report of research in progress. 8th *Australian Conference on Information Systems*, 509–516.

Haddley, N. and Sommerville, I. 1990. Integrated support for systems design. *Software Engineering Journal* 5(6): 331–338.

Jankowski, D. J. 1997. Computer-aided systems engineering methodology support and its effect on the output of structured analysis. *Empirical Software Engineering* 2(1):11–38.

Jarzabek, S. and Huang, R. 1998. The case for user-centred CASE tools. *Communications of the ACM* 39(10): 94–103.

Khwaja, A. A. and Jurban, J. E. 1993. Syntax-directed editing environments: issues and features. *Proc. 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*. Indianapolis, USA, 230–237.

Rehder, B., Pennington, N., and Lee, A. Y. 1997. Scoring the completeness of software designs. *J. Systems Software* 36: 33–68.

Reeves, A., Marashi, M. and Budgen, D. 1995. A software design framework or how to support real designers. *Software Engineering Journal* 10(4): 141–155.

Scott, L. 1997. Hypernode model support for software design methodology modelling. JRCASE Technical Report 97/2, School of MPCE, Macquarie University.

Silver, M. S. 1988. On the restrictiveness of decision support systems. In *Organizational Decision Support Systems*. Elsevier Science, 259–270.

Takada, S., Scott, L. and Brooks, A. 1998. Evaluating data flow diagrams and class diagrams in usability laboratory experiments (CADPRO Pilot #2). JRCASE Research Report No. 98/15, School of MPCE, Macquarie University.

Yourdon, E. 1989. *Modern Structured Analysis*. Prentice-Hall International Inc., 339.

Zelkowitz, M. 1990. Evolution towards specifications environment: experiences with syntax editors. *Information and Software Technology* 32(3): 191–198.

Andrew Brooks received the BSc and MPhil degrees in Astrophysics and Astronomy from the University of Edinburgh in 1978 and 1983, and the PhD degree in Computer Science from the University of Strathclyde in 1991. His current research interests include software engineering and cognitive science. He is a member of the ACM, IEEE, and the British Computer Society. He has held a number of visiting researcher positions at Macquarie University and the University of New South Wales. He is currently a lecturer in Computer Science at the University of Strathclyde but will shortly be taking up a position as Senior Lecturer in Computer Science at the University of Auckland.

F Utbult receieved a BSc degree in Information Systems from the University of New South Wales. He then went on to work as a researcher at Centre for Advanced Empirical Software Research (CAESAR) at University of New South Wales for 2 years. He is currently an Application Programmer/Project Engineer at NDC Netzler & Dahlgren CO in Gothenburg, Sweden.

Catherine Mulligan received a BSc degree in Business Information Technology from the University of New South Wales. She then went on to develop telecommunications software at Ericsson. She is currently working with wireless technologies.

Ross Jeffery is Professor of Software Engineering in the School of Computer Science and Engineering and Director of the Centre for Advanced Software Engineering Research (CAESAR) at The University of New South Wales. Professor Jeffery was the first Head of the School of Information Systems at UNSW from 1989 to 1994. He was the founding Chairman of the Australian Software Metrics Association and also was instrumental in creating the Australian Conference on Information Systems for which he was General Chair for the first two meetings. He served on the editorial board of the IEEE Transactions on Software Engineering for many years, and is an Associate Editor of the Journal of Empirical Software Engineering. He has served on the steering committee of the International Conference on Software Engineering, and is also on the editorial board of the Wiley International Series in Information Systems. He was honoured by the Australian Computer Society for his contribution to research in software engineering. His current research interests are in Software engineering process and product modeling and improvement, software metrics, software technical and management reviews, and software resource modeling. His research has involved over fifty government and industry organizations over a period of 15 years. He has also held positions at The University of Queensland, University of Maryland, and University of Minnesota. He has authored/co-authored four books and over seventy research papers.