Computer Science at Kent

Cause for alarm?: A multi-national, multi-institutional study of student-generated software designs

Technical Report No. 16-04
September 2004

Copyright © 2004 University of Kent
Published by the Computing Laboratory,
University of Kent, Canterbury, Kent CT2 7NF, UK

Cause for alarm?: A multi-national, multi-institutional study of student-generated software designs

Sally Fincher[1], Marian Petre[2], Josh Tenenberg[3], Ken Blaha[4], Dennis Bouvier[21], Tzu-Yi Chen[5], Donald Chinn[3], Stephen Cooper[6], Anna Eckerdal[7], Hubert Johnson[8], Robert McCartney[9], Alvaro Monge[10], Jan Erik Moström[11], Kris Powers[12], Mark Ratcliffe[13], Anthony Robins[14], Dean Sanders[15], Leslie Schwartzman[16], Beth Simon[17], Carol Stoker[18], Allison Elliott Tew[19], Tammy VanDeGrift[20]

1 Computing Laboratory, University of Kent, UK, 2 Dept. of Maths and Computing, Open University, UK, 3 Computing and Software Systems, Institute of Technology, University of Washington, Tacoma, USA, 4 Department of Computer Science, Pacific Lutheran University, USA, 5 Department of Maths and Computer Science, Pomona College, USA, 6 Department of Mathematics and Computer Science, Saint Joseph's University, USA, 7 Uppsala University, Sweden, 8 Computer Science Department, Montclair State University, USA, 9 Computer Science and Engineering, University of Connecticut, USA, 10 Computer Engineering and Computer Science, California State University Long Beach, USA, 11 Dept. of Computing Science, Umeå University, Sweden, 12 Department of Computer Science, Tufts University, USA, 13 Department of Computer Science, University of Wales Aberystwyth, UK, 14 Computer Science Department, University of Otago, New Zealand, 15 Department of Computer Science/Information Systems, Northwest Missouri State University, USA, 16 School of Computer Science and Telecommunication, Roosevelt University, USA, 17 Mathematics and Computer Science Department, University of San Diego, USA, 18 Department of Computer Science, Azusa Pacific University, USA, 19 College of Computing, Georgia Institute of Technology, USA, 20 Computer Science and Engineering Department, University of Washington, Seattle, USA, 21 Parks College of Engineering and Aviation, Saint Louis University, USA

Abstract

This paper reports a multi-national, multi-institutional study to investigate Computer Science students' understanding of software design and software design criteria. Students were recruited at two levels: those termed 'first competency' programmers, and those completing their Bachelor degrees. The study, including participants from 21 institutions over the academic year 2003/4, aimed to examine students' ability to generate software designs, to elicit students' understanding and valuation of key design activities, and to examine whether students at different stages in their undergraduate education display different understanding of software design. Differences were found in particpants' recognition of ambiguity in requirements; in their use of formal (and semi-formal) design representation and in their prioritisation of design criteria.

1. Background

Software design requires a variety of skills and knowledge: within the domain, in software and computing (Soloway & Ehrlich, 1984), and in design (McCracken, 2004). It requires the ability to map both between problems and solutions, and between domain and software/computation.

Software design is difficult: dealing with ill-defined and ill-structured problems; having complex and often conflicting constraints; producing large, complex, dynamic, intangible artefacts; and being deeply embedded in a domain (cf. Goel and Pirolli's characteristics

of the design task, (Goel & Pirolli, 1992)). Marshalling resources, applying knowledge, prioritising sub-tasks, managing constraints, evaluating proposed solutions, and managing the design process are constituent and interacting skills—and potential sources of breakdown even in professional design behaviour (Guindon, Krasner, & Curtis, 1987) (Curtis, 1990). These characteristics make software design elusive to characterise and difficult to teach.

Models of software design involve decomposition and management of the design process (Détienne, 2001). This management includes tracking the relationships among sub-problems, and integrating sub-problems into a coherent structure. Jeffries et al. noted that novices differ from experts in their ability to decompose a problem effectively, to solve sub-problems, and to integrate solutions (e.g., (Jeffries, Turner, Polson, & Atwood 1981)). Experts organise information differently from novices, producing different and larger "chunks" (summarised in (Kaplan, Gruppen, Levanthal, & Board, 1986)). Examining how software designers decompose problems has the potential to provide insight into the cognitive process of "chunking" and hence into how software designers structure their solutions.

2. Research Questions

Many of the studies cited above have examined the processes that software designers, whether novice or expert, employ in producing software. Few studies, however, have examined and characterised design artefacts, especially those of novices. Many open questions remain: Do novice designs exhibit characteristics associated with software design expertise, such as loose coupling and tight cohesion (Yourdon & Constantine, 1978)? How do novices conceptualise what constitutes a software component and how components interrelate? Will their conceptualisation reflect the initial programming language paradigm to which they have been exposed? Do novices generate software designs using standard design representations such as UML (Rumbaugh, Jacobson, & Booch, 1998), or natural language descriptions, code, and ad-hoc representations?

What criteria do novice designers employ when doing software design? Have novices internalised any heuristics or principles that expert designers have codified ((Mayer, 1997), (Riel, 1996))? Do they view their design criteria as relatively task- and context-invariant, or do they use different criteria in different situations?

The study has an enveloping framework that compares data from first-competency and completing students (Atman, Chimka, Bursic, & Nachtmann, 1999) and from educators. This supports comparative questions. Do students' problem decompositions, and their ability to describe and represent a solution change during the course of their education? Are students' decompositions and descriptions significantly different from educators'? Do students' relative prioritisation of design criteria alter over time, and how closely does their prioritisation approximate educators' prioritisation at each stage?

3. The Study

This study used two tasks to explore students' understanding of the software design process:

- *a decomposition task*, to examine students' ability to analyse a problem and then design an appropriate solution structure, and to elicit students' understanding-in-action of fundamental software design concepts, and

- *a design criteria prioritisation task*, to elicit which criteria students consider most and least important for different design scenarios.

Decomposition Task

Participants were given a design brief, specifying a "super alarm clock" to help students manage their sleep patterns (see the Design Brief in Appendix A). The design brief stated:

> In doing this you should (1) produce an initial solution that someone (not necessarily you) could work from (2) divide your solution into not less than two and not more than ten parts, giving each a name and adding a short description of what it is and what it does – in short, why it is a part. If important to your design, you may indicate an order to the parts, or add some additional detail as to how the parts fit together.

Participants were prompted with generic language (e.g., *part* rather than *object* or *function*), to elicit their concept of what constitutes a part, and how to describe and represent parts. On completion of their designs, participants were asked to "talk through" their design, and to name and describe the function of each part.

Design Criteria Prioritisation Task

After completing the decomposition task, participants were given 16 cards, each describing a single design criterion. Participants were asked to indicate the five most important and the five least important criteria for each of four scenarios:

- for the design they had just completed (current task),
- for the current task, but in a team (task in team),
- for the current task—on their own—but delivering a fully-functional result at the same time tomorrow (extreme time pressure), and
- for the current task, but designing the system as the basis of a product line that would have a 5-year lifespan (longevity).

The sixteen criteria are further described and included in Appendix B.

Participants

Participants recruited from 21 institutions of post-secondary education from the USA, UK, Sweden and New Zealand completed the same tasks. Three types of participant were represented from each institution:

- *First competency students*. (FC) To ensure comparability across institutions, students were selected at the point in their education where they could be expected to program at least one problem from the set proposed by McCracken ((McCracken et al., 2001)). These problems involve the simulation of a simple calculator for arithmetic expressions. The McCracken problem set was used because it references levels of competence, irrespective of curriculum and was devised for use in one of the first multi-national, multi-institutional CS Education Research studies. Not all of the FC participants were Computer Science majors, but all had taken, or were taking, a Computer Science course.
- *Graduating students*. (GS) Graduating students were defined as being those within the last eighth of a Bachelor degree program, or equivalent (e.g., the last

6 months of a four-year, full-time program) in Computer Science or a software-intensive degree.

- *Educators*. (E) Educators were defined to be those holding faculty positions, and teaching in the undergraduate program.

The total cohort consisted of 314 participants from 21 institutions representing 28 educators, 136 first-competency and 150 graduating students.

For each participant the following material was collected: their representation of the design (i.e. the marks they made on paper), the time they took to make it, and a record of their prioritisation of the design criteria. Full transcriptions of verbalisation during the task were made for a proportion of the students; researcher notes were made for all. For analysis purposes, student participants were allocated to one of five "performance buckets". In the same way as the study employed the McCracken problem set as a discriminator, allowing selection of participants with similar aptitude from different institutions, "performance buckets" were devised to help group students meaningfully across institutions without having to consider, for example, whether an "A" grade in the US means the same thing, for example, as a "90%" in the UK. The five categories appear in table 1.

Identifier	Descriptor
1. Picasso	Exceptional students. You would only expect to see one or two per year
2. Top	Excellent students.
3. High Side	Students who have done everything they should, but who did not, perhaps, do it as well as they could have.
4. Low Side	These students have passed the course, but on minimum criteria
5. Fail	Below the pass mark.

Table 1: Identification and Description of "performance buckets"

The categories were accompanied by a protocol, determining allocation of students. On a four point (4.0) scale (used in the USA), the divisions would occur at: 4, 3.7, 2.7, 1.7 and 0 where these numbers represent the bottom of the category in question. A *Picasso* has a 4.0; a 3.7 or higher falls into the *Top* category. In terms of percentages, these would mean: 100%, 93%, 67%, 42% and 0. In terms of letter grades, this roughly maps to: A+, A, B-, C- and F. Here, a "D" was considered to be synonymous with failure (although this is considered a pass in some contexts).

Excluding educators and six FC students for whom no grade data was available, the cohort divided between the performance buckets are shown in table 2.

	1 (Picasso)	2 (Top)	3 (High Side)	4 (Low Side)	5 (Fail)
FC	15	24	56	26	9
GS	9	34	87	19	1

Table 2: Division of the cohort between "performance buckets"

4. Results

Analysis was conducted on three aspects of the data: participants' recognition of ambiguity in requirements, how the design artefacts are characterised in terms of such things as the grouping structures and interactions between parts, and how subjects value different design criteria in different design contexts.

4.1 Recognizing Ambiguity in Requirements

4.1.1. Question

This analysis was conducted to investigate participants' recognition of ambiguous aspects of the design brief requirements. A related interest was a comparison of participants who recognize ambiguity to those participants who don't with respect to how fully the requirements were addressed. Recognition of ambiguity was defined by these observable events within the study: asking a question about ambiguities or omissions in the specification, or making explicit assumptions in writing or in speech.

4.1.2. Motivation

In comparing the design processes of freshman and senior engineering students, Atman *et al*. found that seniors made more requests for additional information and made more than three times as many assumptions (Atman et al., 1999). It is suggested in *Are Your Lights On?* that a designer's "lights aren't on" if he fails to raise several questions when understanding a problem specification (Gause & Weinberg, 1982). Christiaans and Dorst (Christiaans & Dorst, 1992) found that novice industrial design engineers tend to scope out a problem less and seek less information than experienced designers. Similarly, Bogusch *et al.* also found that freshmen engineers tend to define problems narrowly while more experienced seniors tend to define problems more broadly (Bogush, Turns, & Atman, 2000). Rowland found that novices made few requests for clarifications relative to a design problem (Rowland, 1992).

Recognizing and addressing ambiguity is important because ambiguities in requirements can propagate to errors in the design solution. It is cheaper to recognize and resolve ambiguities early, rather than after the design is completed (Boehm, 1981). Thus, recognizing ambiguity in the design phase is less costly in terms of time completion and number of bugs. Observed differences between participant groups and patterns among other participant characteristics can give insight into the maturity of software designers. Understanding patterns of these subpopulations with regard to ambiguity could provide

clues for ways to enhance the education of future software designers. For example, if ambiguity is not commonly part of homework assignment specifications, then students may not have practice in recognizing it.

4.1.3. Data Analysis

To study the question regarding participants' recognition of ambiguity and the level to which they address requirements the following questions are asked of the corpus:

1. Did the participant ask questions about ambiguities and omissions in the specification (as distinct from questions about word meanings or procedural questions)? (Yes or No)
2. Did the participant make explicit assumptions in the description, representation or other recorded responses about ambiguities and/or omissions in the specification)? (Yes or No)
3. Did the subject attempt to address the requirements of the specification? (Yes, Partially, Hardly, No)

A list of nine requirements was generated to assist in answering question 3. They are detailed in Appendix C.

Each participant was then categorized (see Table 3) on whether they addressed each of these requirements *at all,* with no valuation of the quality of the result.

Categories	Number of Requirements Addressed
Yes	9 (100%)
Partially	5-8 (>= 50%)
Hardly	1-4 (< 50%)
No	0 (0%)

Table 3: Categorisation of participants' addressing of requirements

In this survey of the corpus there is data from all study institutions and the counts of each participant grouping are as follows: FC = 136, GS= 150 and E=28. The data for the separate analyses excludes participants for which the question associated with the analysis could not be answered based on observable behaviour.

Recognizers and Non-Recognizers of Ambiguity

A participant *recognizes ambiguity* if they are observed doing one or both of the following: asking a question or making an assumption. The observed question or assumption could be made in the written representation or verbally during the decomposition task or interview. The participants who recognize ambiguity define a new subpopulation termed **recognizers**. Those who did not recognize ambiguity are termed **non-recognizers**.

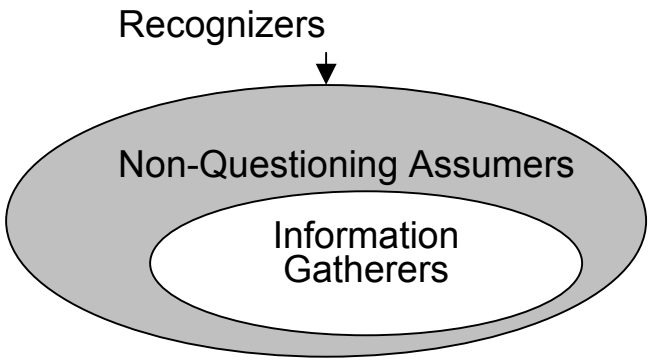
The recognizers and non-recognizers were analyzed with respect to participant group, performance bucket, time taken to perform the decomposition task, and institution.

Additionally, the prioritization task data of recognizers and non-recognizers are analyzed to determine if any particular design criteria is more often identified as important to either subpopulation.

Information Gatherers

As noted above, information gathering about an under-specified task is a characteristic of more experienced designers. The **information gatherer** subpopulation represents a subset of the recognizers. This subpopulation asks questions, and may or may not make observable assumptions.

Analysis of information gatherers includes looking for patterns among participant groups and comparing those to similar patterns for recognizers. Additional comparisons between information gatherers and recognizers are made with regards to performance bucket, time to complete the decomposition task, and institution.



Studying Thoroughness of Addressing Requirements

Additionally, data regarding how thoroughly each participant addressed the requirements listed in the specification is used for analyzing differences between recognizers and non-recognizers. An analysis to compare design criteria chosen during the prioritization task between recognizers and non-recognizers is also performed.

Introduction of Bias

One introduction of bias occurs in the creation of definitions for recognizer, non-recognizer, information gatherer, and non information gatherer. These definitions are based on observable behavior (asking questions and making explicit assumptions). Participants could have made assumptions about the specifications without externalizing these assumptions in written or verbal form. These participants are coded as not making assumptions in our analysis. The number of participants making assumptions could be larger than the number created in our analysis.

A second source of bias is introduced when answering the questions about participants asking questions about ambiguities/omissions in the specification, making explicit assumptions, and addressing requirements. In a few cases, it is difficult to tell if a participant makes an explicit assumption. For these cases, the participant is coded as "Don't Know" in terms of making assumptions. The answer to the question involving the level to which the requirements are addressed was operationalized into four categories: Yes, Partially, Hardly, No. The small number of answer choices helps to increase consistency across coders and reliability within coders.

4.1.4. Results

How many participants recognized ambiguity?

Figure 1 indicates the percentage of participants who recognized ambiguity versus those who did not recognize ambiguity with respect to participant group. The figure shows that of the first competency students, 63% (82 of 131) recognized ambiguity. 76% (110 of 145) of graduating seniors and 89% (24 of 27) of educators recognized ambiguity. 216 participants are recognizers and 87 are non-recognizers. 11 participants could not be classified as recognizers or not recognizers, for a total of 303 participants included in the analysis.

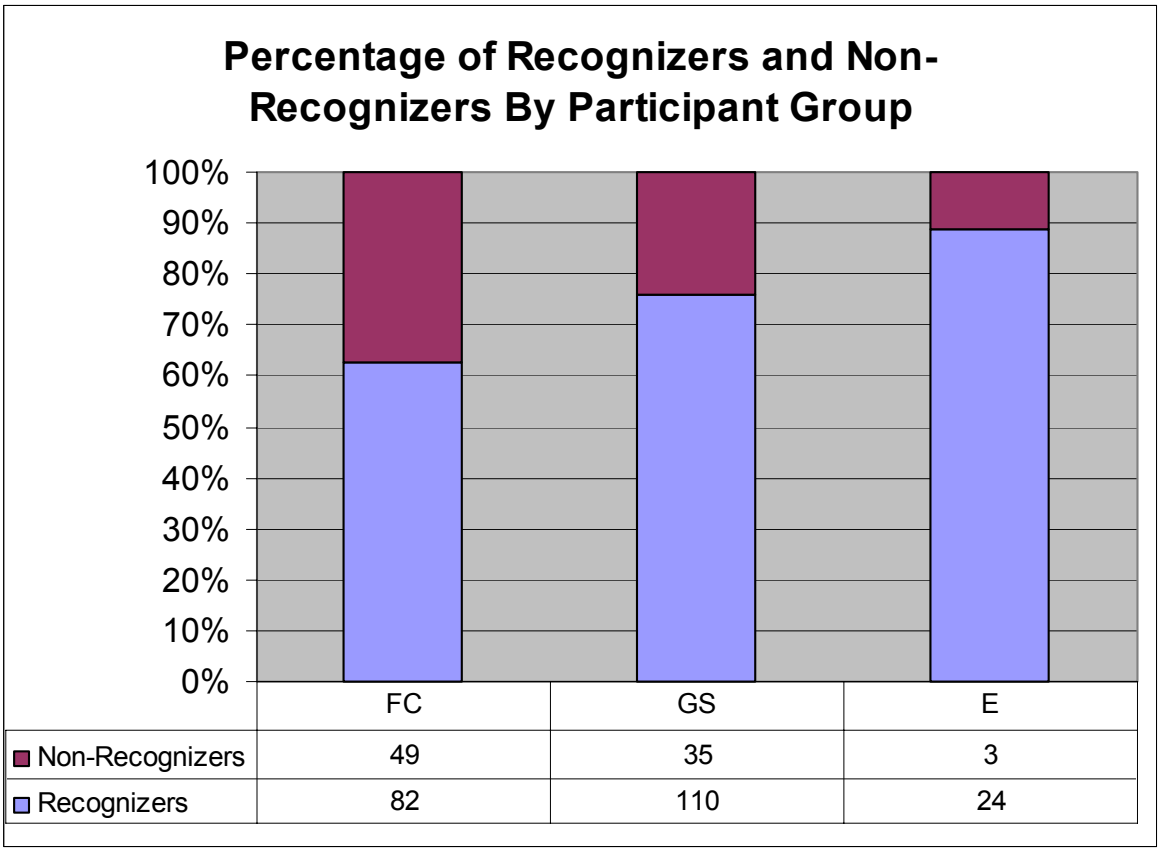


Figure 1

How many participants gathered information?

Figure 2 shows the breakdown of information gatherers by participant group. 33% (43 of 131) of first competency students, 50% (73 of 145) of graduating seniors, and 81% (22 of 27) of educators gathered information during the decomposition task. Of the participants recognizing ambiguity, 48% (39 of 82) of first competency students, 34% (34 of 110) of graduating seniors, and 8% (2 of 24) of educators made assumptions without requesting information. There are 138 information gatherers and 165 non information gatherers, with no data for 11 participants.

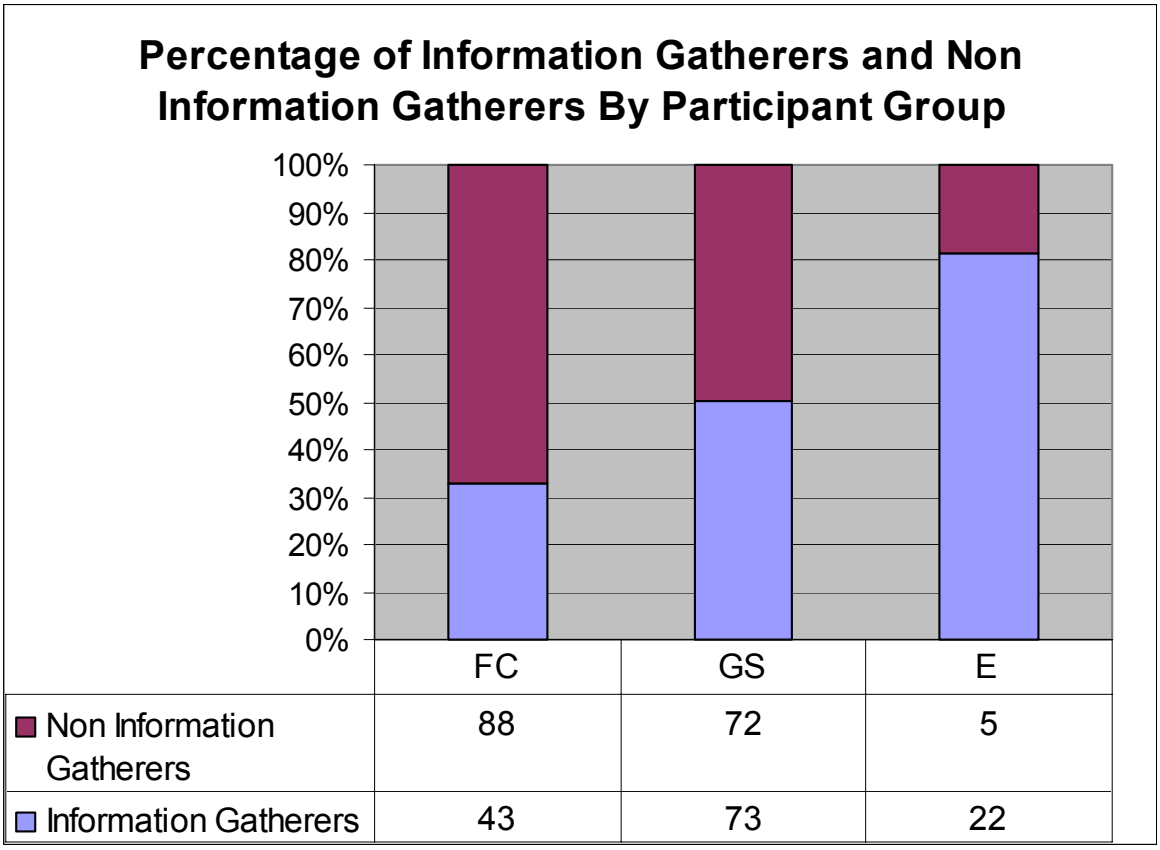


Figure 2

What is the breakdown of recognizers versus non-recognizers with respect to how many requirements were addressed?

Figures 3 and 4 show the relationship between recognizers/non-recognizers and requirements groups. The requirement groups include *Yes*, *Partially*, *Hardly*, and *No*. The general trend is that as the number of requirements addressed decreases, the percentage of recognizers decreases. *Yes* includes 120 participants, *Partially* has 151 participants, *Hardly* has 26, and *No* has 4. Of the participants classified as recognizers and non-recognizers, 2 did not have data pertaining to the requirements addressed.

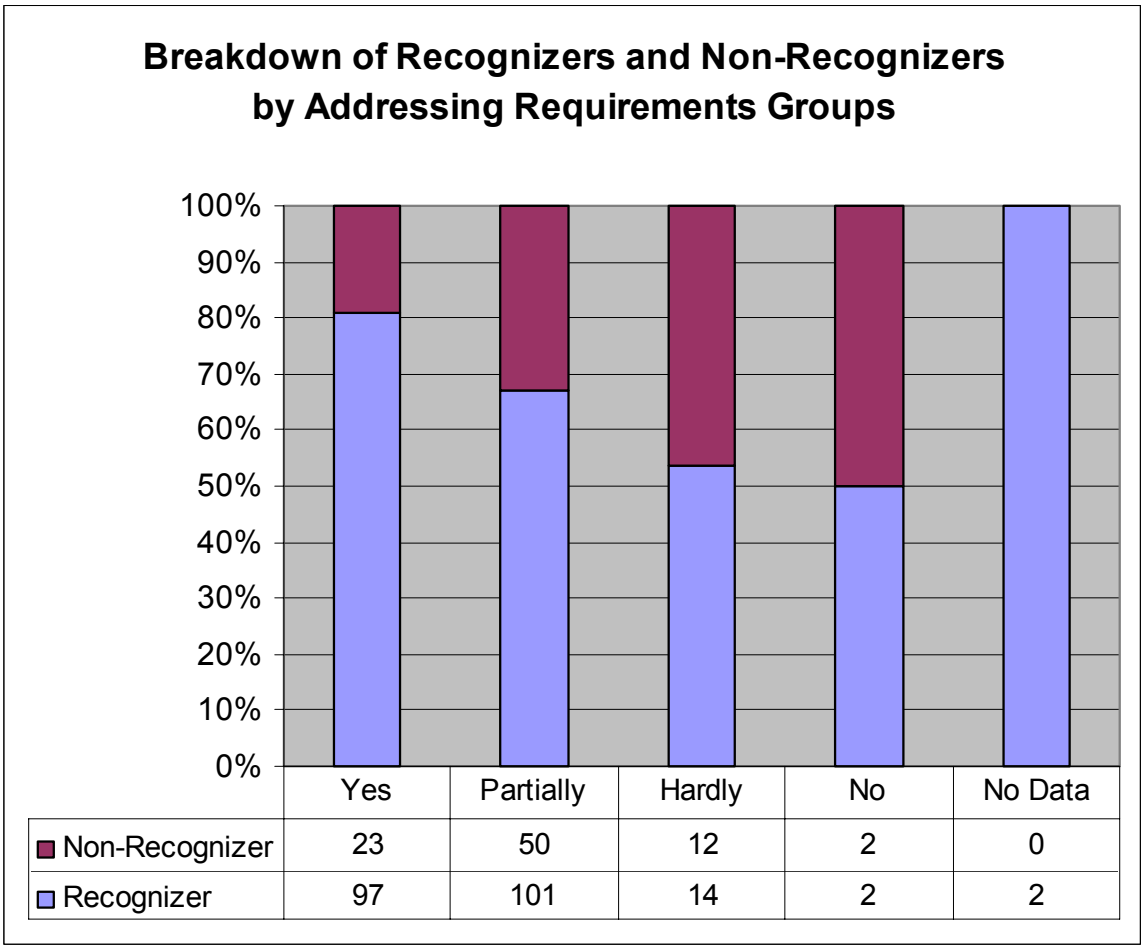

Breakdown of Recognizers and Non-Recognizers
by Addressing Requirements Groups
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Yes
Partially
Hardly
No
No Data
Non-Recognizer
23
50
12
2
0
Recognizer
97
101
14
2
2

Figure 3

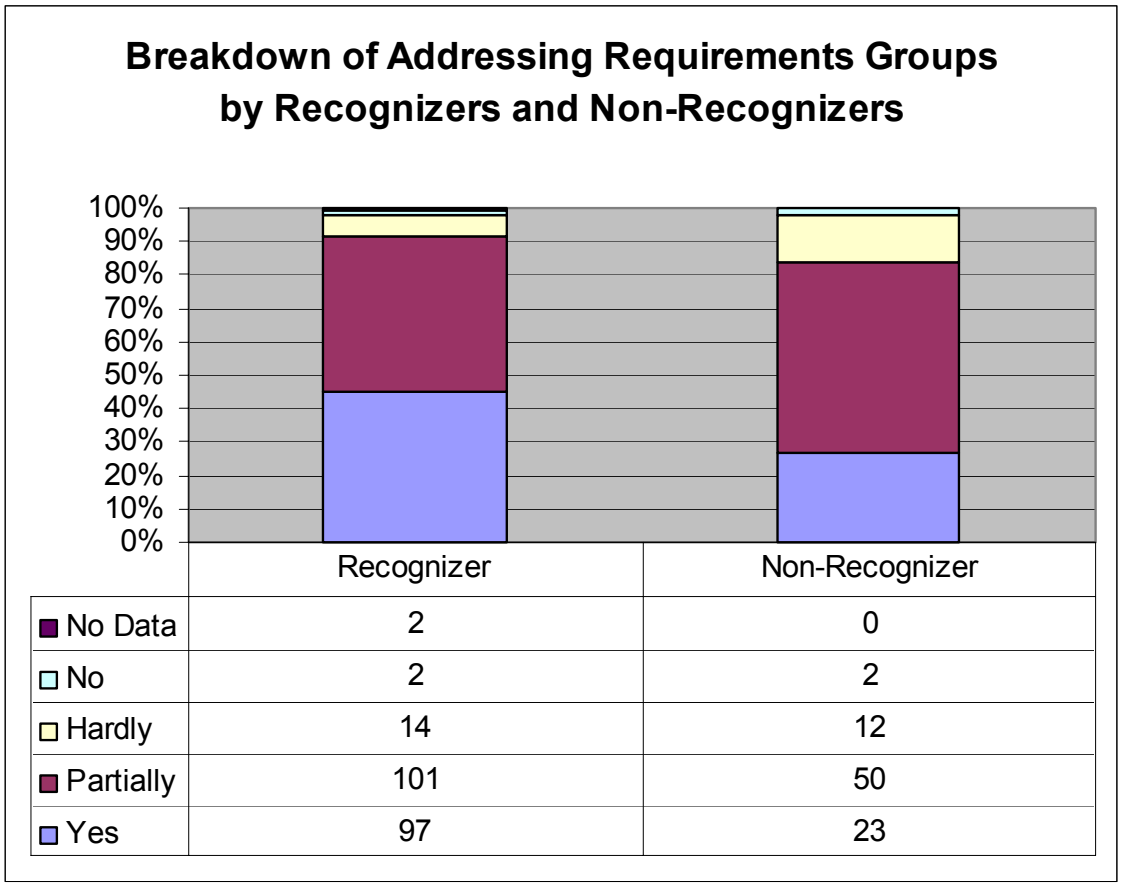


Figure 4

What is the breakdown of information gatherers versus non information gatherers with respect to how many requirements were addressed?

Figures 5 and 6 show the relationship between information gatherers/non information gatherers and requirements groups. The general trend is that as the requirement groups go from Yes to No, the percentage of information gatherers decreases.

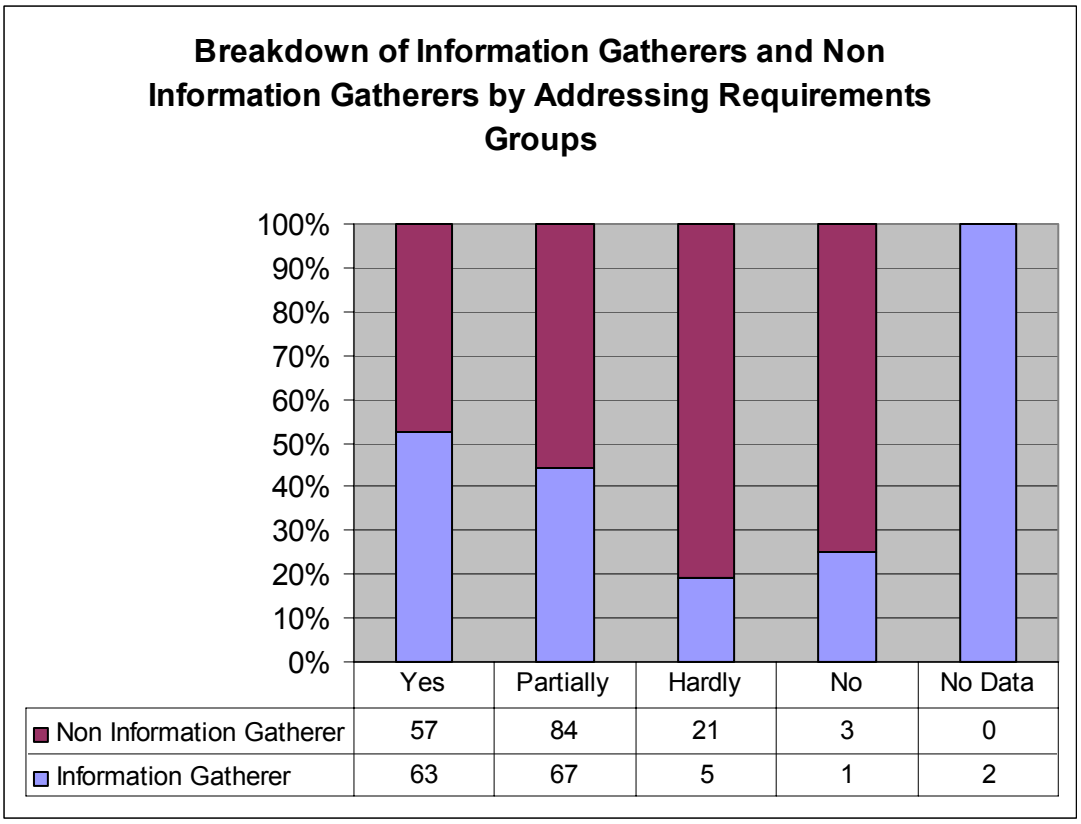
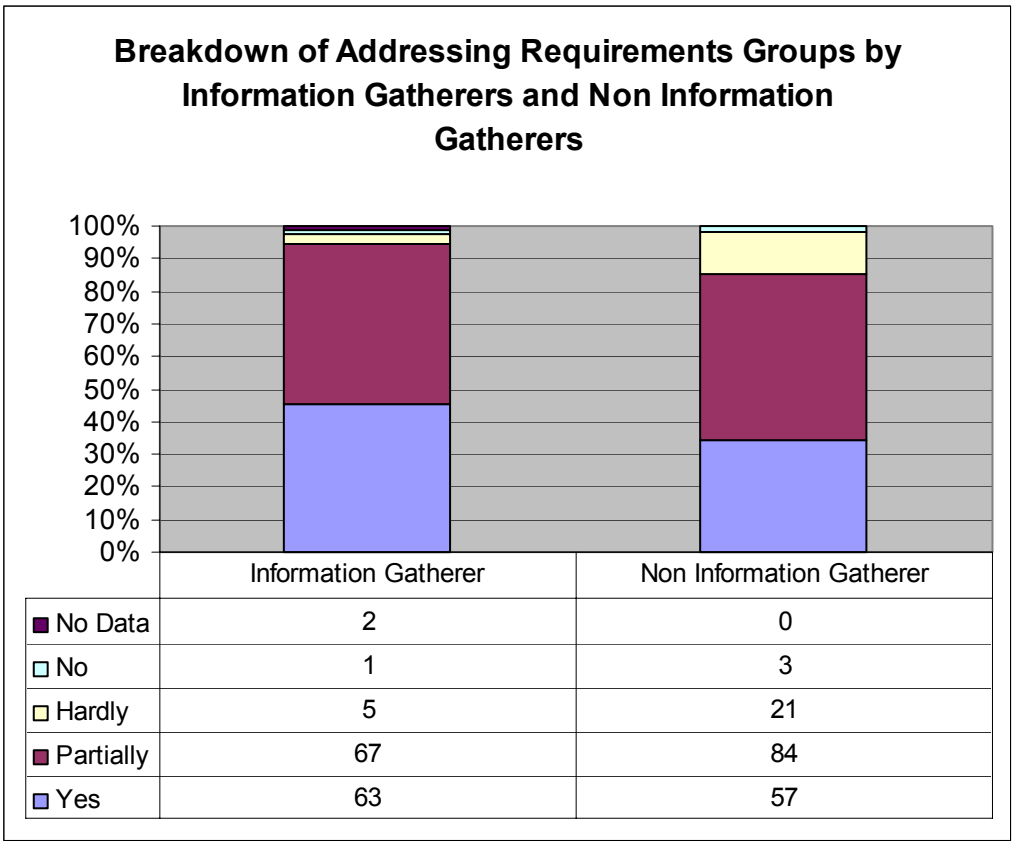

Breakdown of Information Gatherers and Non Information Gatherers by Addressing Requirements Groups
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Yes
Partially
Hardly
No
No Data
Non Information Gatherer
57
84
21
3
0
Information Gatherer
63
67
5
1
2

Figure 5



	Information Gatherer	Non Information Gatherer
No Data	2	0
No	1	3
Hardly	5	21
Partially	67	84
Yes	63	57

Figure 6

What is the relationship between recognizers and non-recognizers and information gatherers and non information gatherers with respect to performance bucket?

Figures 7 and 8 show the FC and GS student populations and performance buckets with respect to recognition of ambiguity and information gathering behaviour, respectively. Of the FC recognizers and non-recognizers, 15 were in bucket 1, 24 were in bucket 2, 54 were in bucket 3, 24 were in bucket 4, 8 were in bucket 5, and 6 had no performance bucket data. Of the GS recognizers and non-recognizers, 9 were in bucket 1, 34 were in bucket 2, 85 were in bucket 3, 17 were in bucket 4, and 1 was in bucket 5. One might expect that participants in bucket 1 would tend to recognize ambiguity. The trend indicates that in the GS group, as performance bucket decreases, the percentage of recognizers decreases and then increases. For the FC group, as performance bucket decreases, the percentage of recognizers increases and then decreases. About half of the GS participants in buckets 1 through 4 gathered information. None of the GS participants in bucket 5 gathered information. The highest percentage of information gatherers in the FC group were in bucket 2, at 46%.

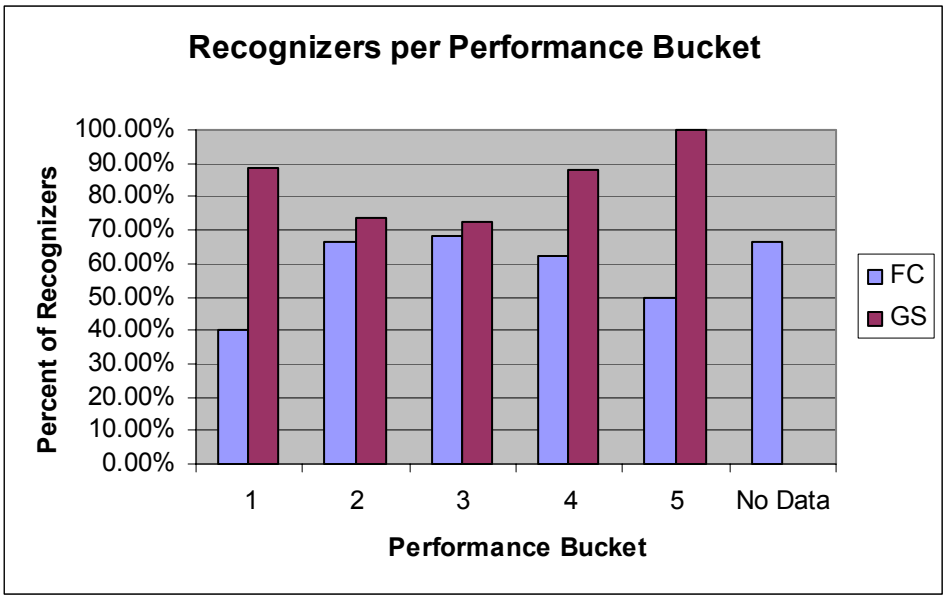


Figure 7

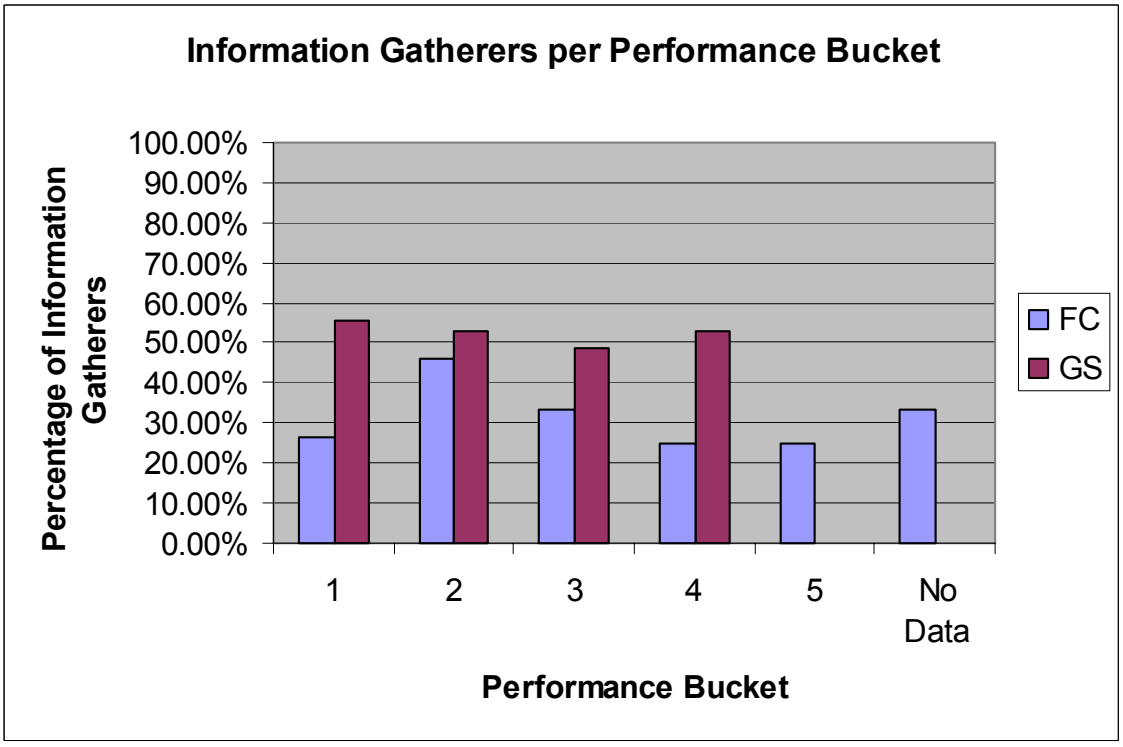


Figure 8

How long did recognizers/non-recognizers and information gatherers/non information gatherers take for the decomposition task?

The average amount of time taken for the recognizers is 41.2 minutes and 31.7 minutes for the non-recognizers. The trend for information gatherers is similar with information gatherers taking an average of 44.0 minutes and non information gatherers taking an average of 33.8 minutes. Figures 9A and 9B show the distributions of these groups according to distinct 10-minute time intervals.

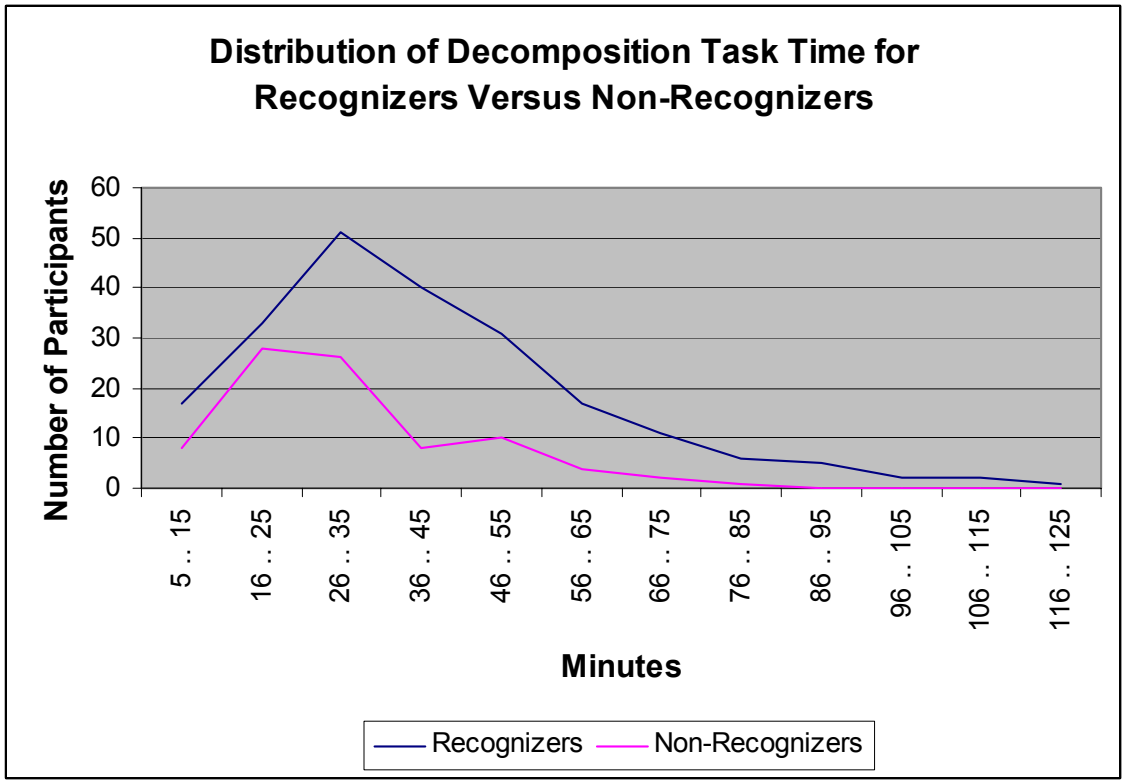


Figure 9A

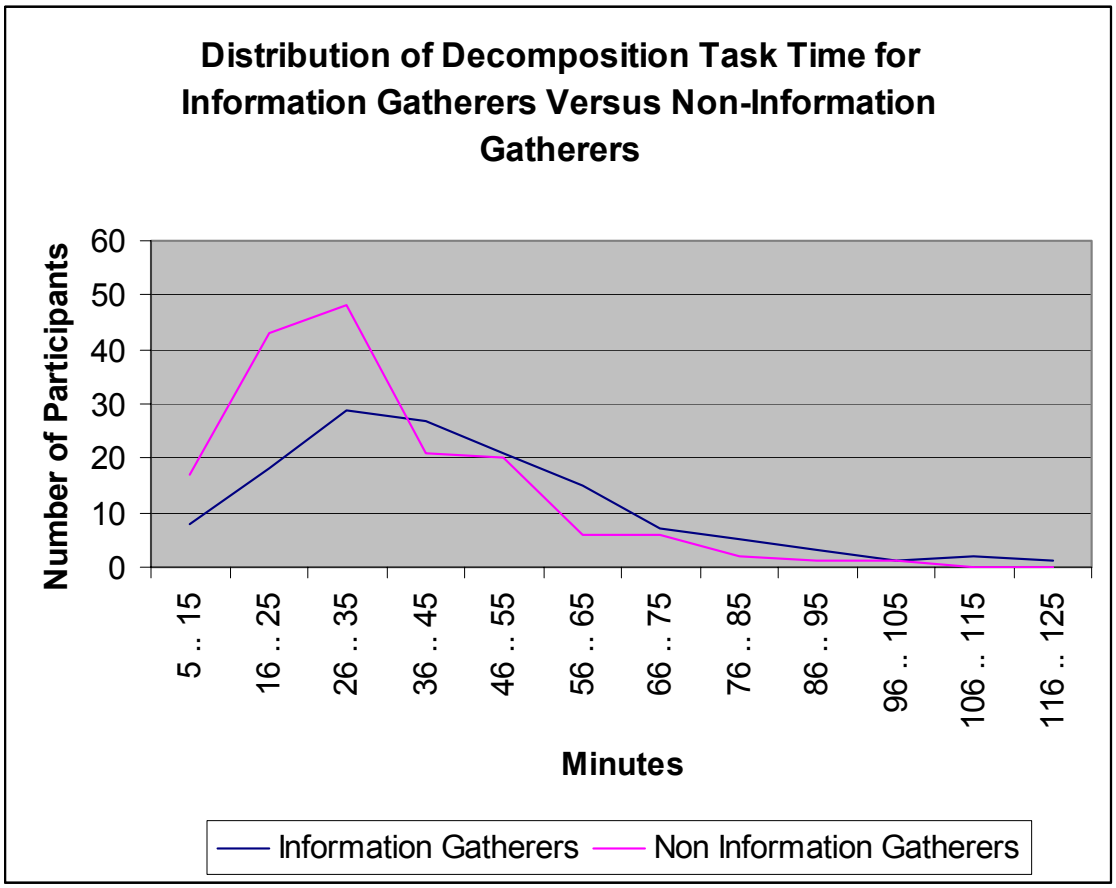


Figure 9B

What are the percentages of recognizers and information gatherers per institution?

Figure 10 shows percentages of recognizers and information gatherers per institution. L, S, T, and V had over 60% of study participants gathering information. C, I, N, V, and Y were institutions in which all participants recognized ambiguity.

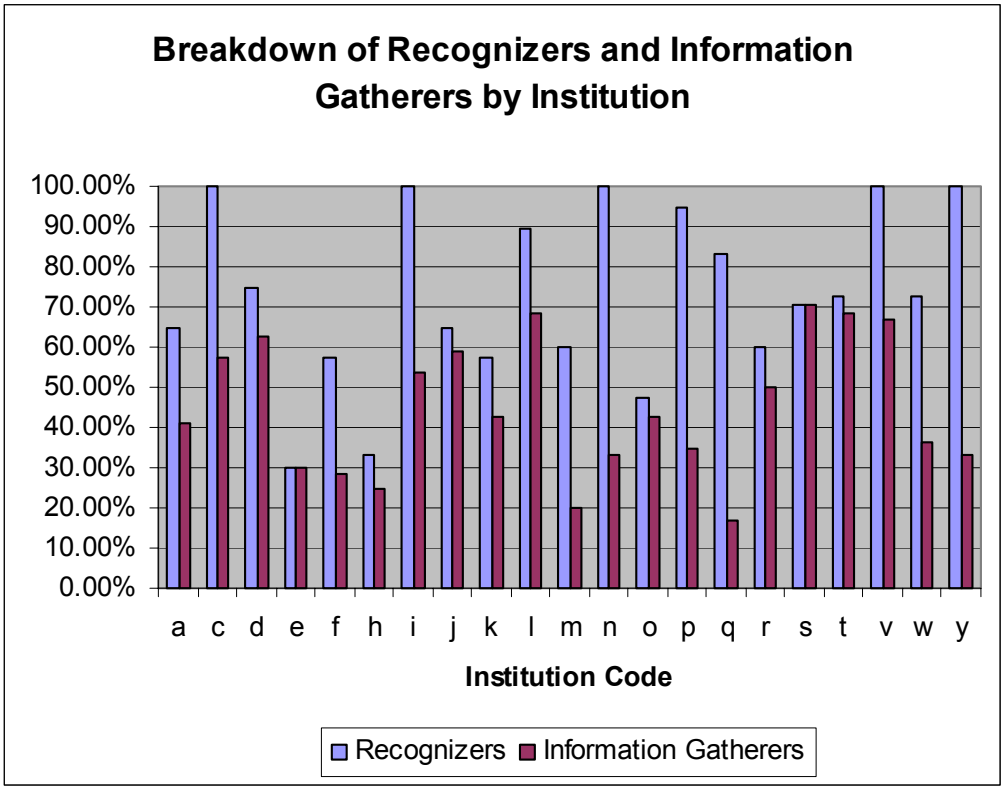


Figure 10

How do institutions compare with respect to the breakdown of participant groups into requirements groups?

Figures 11, 12, and 13 show the breakdowns by institution of the FC, GS, and all participants, respectively, according to the requirements groups for the complete set of 314 participants.

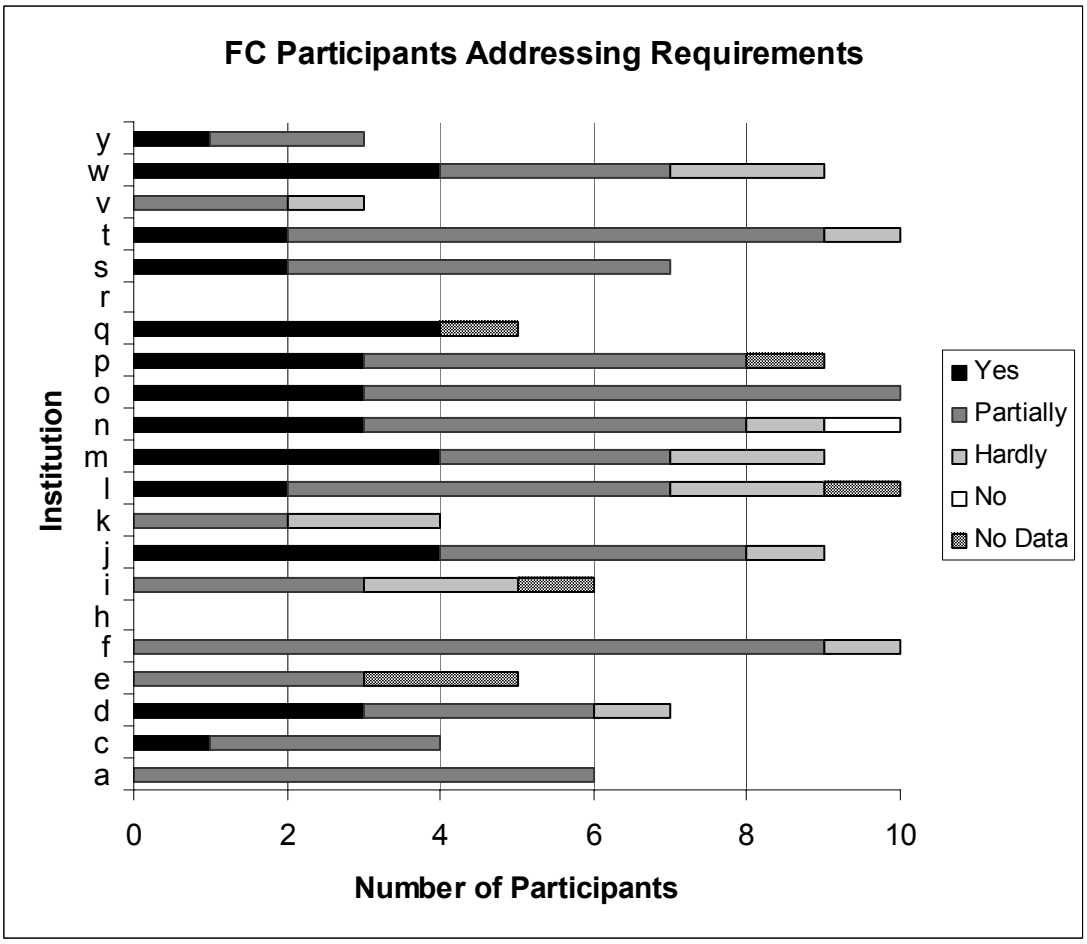

FC Participants Addressing Requirements
Institution
y
w
v
t
s
r
q
p
o
n
m
l
k
j
i
h
f
e
d
c
a
0
2
4
6
8
10
Number of Participants
Yes
Partially
Hardly
No
No Data

Figure 11

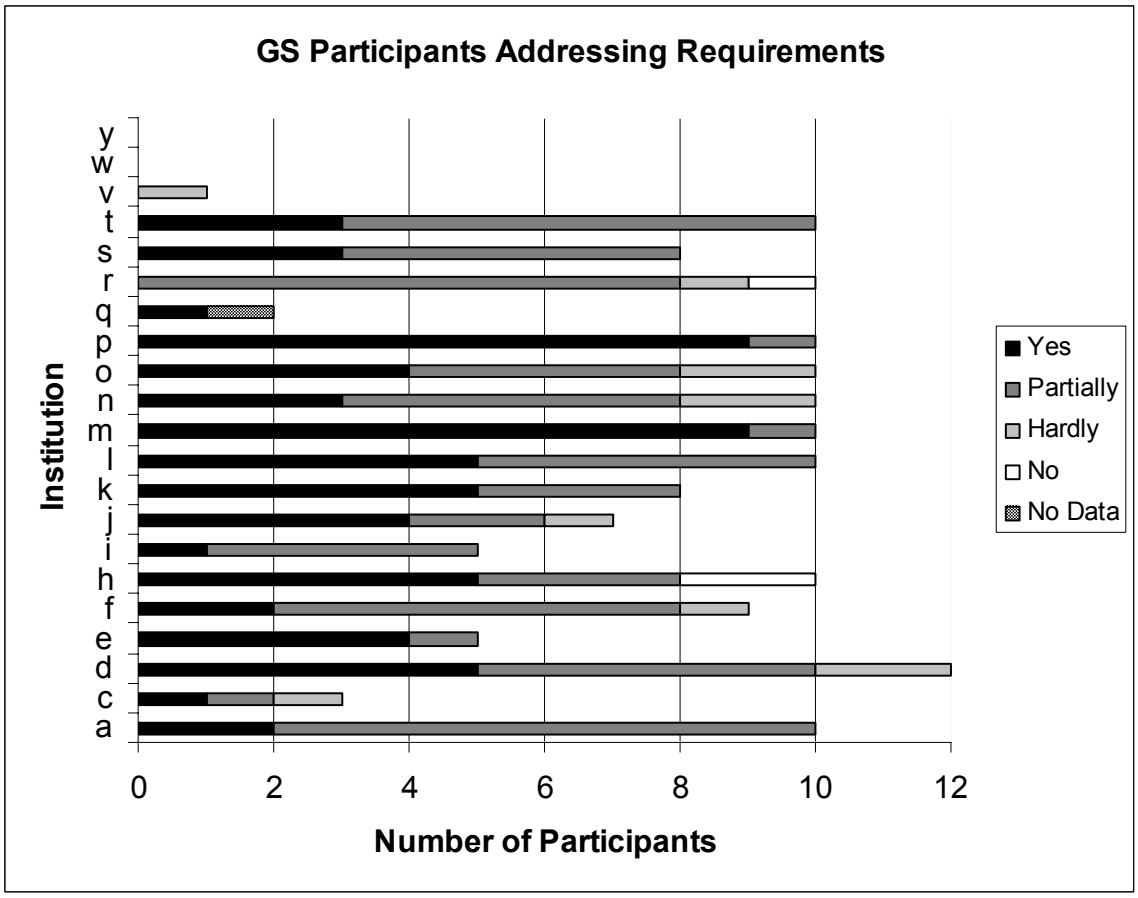

GS Participants Addressing Requirements
Institution
y
w
v
t
s
r
q
p
o
n
m
l
k
j
i
h
f
e
d
c
a
0
2
4
6
8
10
12
Number of Participants
Yes
Partially
Hardly
No
No Data

Figure 12

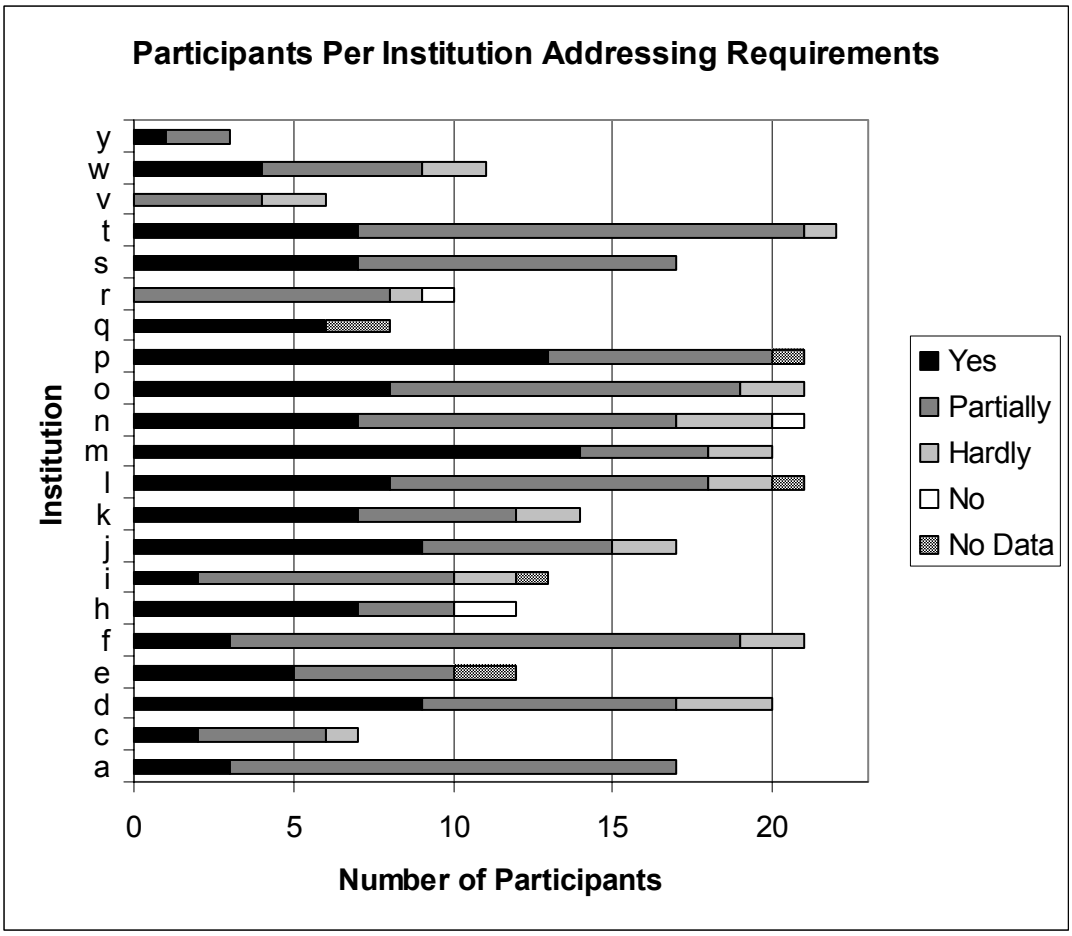


Figure 13

Are there any differences between recognizers and non-recognizers with respect to design criteria selected for the current task?

No. The percentages weighted by the size of each subpopulation for each design criterion selected as most important for the design criteria prioritization task of current task varied from 46% to 57%. There is little difference between the design criteria selected as important by recognizers versus the design criteria selected as important by non-recognizers. There is also no difference between the information gatherers and non information gatherers.

4.1.5 Discussion

The percentage of recognizers increases from first competency students to graduating seniors to educators. Likewise, the percentage of information gatherers increases with respect to the same order. Information gatherers are a subset of recognizers, so the trend is not surprising. Proportionally more seniors recognize ambiguity and seek information than first competency students. This result is consistent with Atman *et al*. in that more seniors request information than freshmen (Atman et al., 1999). Figure 2 indicates that

almost half of the first competency group made assumptions without requesting additional information while this figure is only 8% for educators. Making assumptions without gathering information is a characteristic of less experienced designers.

With respect to addressing requirements, the percentage of recognizers decreases as the requirements groups move from Yes to No. This indicates that those who recognized ambiguity had a higher success rate in addressing all requirements than those who did not recognize ambiguity. Participants who gathered information also had a higher success rate in addressing all requirements than those who did not gather information. This trend indicates that there is an association between recognizing ambiguity and successfully fulfilling requirements.

Recognizers took more time, on average, than the non-recognizers for the decomposition task. Similarly, information gatherers took more time than non information gatherers for the decomposition task. This is not surprising since it takes time to ask questions during the task.

Recognizers and non-recognizers of ambiguity did not differ with respect to selecting the five most important design criteria for the current task.

4.1.6 Conclusions/Implications/Limitations

The results from analyzing participants' information-gathering, assumption-making, and requirements-addressing indicate that as students go from first-competency to graduating seniors, they tend to recognize ambiguities in under-specified problems. Additionally, participants who recognized ambiguity (and the subset who gathered information) had a higher success rate in addressing all requirements. These results imply that with experience, students will become more aware of ambiguous specifications and by realizing that ambiguities exist, they can design software that meets requirements. This may imply that educators should be more explicit in teaching students how to recognize ambiguities in specifications.

4.1.7. Future Work

The data will be analyzed further to classify participants' detailed questions and assumptions. Questions 3 and 4 below will provide information as to how broadly computer science students define design problems. A classification scheme similar to that in (Bogush et al., 2000)could be used.

1. Did participants add features or enhancements beyond the requirements?
2. How many requirements (number of the total) do the students address in their design?
3. Did participants ask questions about the user population (students, sleep researchers, mothers)?
4. Did participants mention ethical or social impacts of the alarm clock system?

4.2 Design Characteristics in Software Design

4.2.1. Motivation

Studies have compared expert and novice behaviour in many domains (e.g. chess (Chase & Simon, 1973), physics ((Maloney, 1994)) or programming ((Wiedenbeck, 1985)). The current study generated a large and rich population of case studies of software design that

is relevant to this literature. This section presents a preliminary analysis of our FC, GS and E designs, which correspond to various points within the novice – expert continuum.

No attempt was made to assess the overall "quality" of the participant's software designs, but they can be compared on a number of more specific dimensions. These dimensions include obvious indicators such as the times taken to produce each design and the number of parts in each design. The use of formal notation, the amount of grouping and interaction, the communication between parts and the level of hierarchical organisation are all potential indicators of design richness. These latter factors were chosen based on a set of potential criteria elicited from a variety of international CS educators (see Appendix B).

4.2.2 Question

This section of the study investigated a number of different areas that might highlight differences in the software design between the different participant groups. Where appropriate, the investigation considered differences between participating institutions. The particular questions addressed were to what extent did the participant groups vary in their:

- number of parts identified in their software design?
- utilisation of formalized software design representation techniques?
- time required to produce a software design?
- use of hierarchical, nested or grouping structures?
- indication of interaction between its parts?

4.2.3 Data Analysis

Number of Parts

The design task protocol was specified so as to capture the number of parts for each design, and the name of each part. One of the most basic ways to characterize the designs is to examine the number of parts. Table 5 shows a breakdown of this information (each row of the table represents the participants at a single institution by institution letter code). From Table 5 we can derive, for any specified subpopulation, information about the distribution of the number of parts. For one subject (F01 at institution Q) it was not possible to identify the number of parts in their software design: they were subsequently excluded from all calculations. Table 4 shows such an analysis for the participant subpopulations FC, GS and E, and for the participants from each institution. The columns of the table are N (the number of participants in the subpopulation) and the minimum, maximum, mean, median, mode and standard deviation of the number of parts.

A one way analysis of variance shows that the variation in the mean number of parts for FC (5.1), GS (4.6) and E (6.2) participant groups were highly significant ($p = 0.001$), although there is no clear trend over the educational level of these subgroups (note that GS have the lowest mean). Variation in mean number of parts by institution was also significant ($p < 0.01$), but variation by performance bucket (over FC and GS) was not.

SUBPOPULATION	*N*	MIN	MAX	MEAN	MEDN	MODE	SD
Educators	28	3	14	6.2	5.5	4	2.9
First competency	135	2	11	5.1	5	3	2.2
Graduating students	150	2	12	4.6	4	4	1.9
Institution A	17	2	9	4.529	4	2	2.09
Institution C	7	3	12	6	5	3	3.251
Institution D	20	2	10	4.8	4	3	2.227
Institution E	12	2	7	4.667	4	4	1.434
Institution F	21	2	10	4.238	4	3	1.875
Institution H	12	3	9	4.333	4	4	1.546
Institution I	13	2	14	5.615	5	5	2.975
Institution J	17	3	9	4.765	4	3	1.699
Institution K	14	2	10	5.143	5	5	2.1
Institution L	21	2	10	4.952	6	6	2.214
Institution M	20	2	10	5.65	5.5	7	2.056
Institution N	21	3	10	5.667	5	4	1.984
Institution O	21	3	9	4.619	4	4	1.704
Institution P	21	2	9	4.333	4	4	1.643
Institution Q	7	3	10	6.857	6	6,10	2.416
Institution R	10	2	6	3.6	3.5	4	1.2
Institution S	17	2	10	5.647	5	5	1.969
Institution T	22	2	7	4.364	4	3	1.639
Institution V	6	3	13	7.167	6.5	5	3.287
Institution W	11	3	11	4.545	4	3	2.463
Institution Y	3	7	9	7.667	7	7	0.943

Table 4: Distributions of numbers of parts, by Institution

	E01	E02	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10	G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G12
A	6		4	8	3	2	3	5					5	4	6	5	7	2	4	9	2	2		
C			9	5	7	3							3	3	12									
D		8	3	10	3	7			3	4	7		3	7	6	2	3	4	7	3	2	3	5	6
E	3	7	4	6	2	4	4						6	6	4	6	4							
F	3	4	4	3	3	6	2	4	10	6	6	3	3	2	4	5	6	3	2	6	4			
H	4	4											5	5	4	4	4	3	9	3	4	3		
I	8	14	5	6	8	4	5	6					5	2	3	3	4							
J	7			4	3	6	9	3	3	4	5	6	5	3	5	3	4	4	7					
K	4	3	2	7	8	7							5	3	5	5	4	4	10	5				
L	7		6	4	6	3	2	8	3	6	6	4	4	6	2	2	3	2	6	7	10	7		
M	3		7	4	10	6	4	7	5	5		2	9	7	7	6	3	7	8	5	4	4		
N	8		9	4	4	3	5	6	10	10	5	4	6	5	5	5	4	6	4	5	4	7		
O	5		4	5	4	4	9	5	5	3	4	7	3	4	4	3	4	5	3	4	9	3		
P	9	4	6	6	3	5	3		3	7	3	5	5	5	2	4	4	3	4	4	4	2		
Q	5		X	10	3	10	8						6		6									
R													5	6	2	4	3	2	3	4	4	3		
S	10	7	3	5	7	5	5	3	9				5	5	7	2	6	5	6	6				
T	5	7	3	3	7	3	7	5	3	5	3	4	7	2	4	2	6	3	4	6	4	3		
V	8	13	3	5	5								9											
W	3	4	3	4	8	3	11	3	3	4	4													
Y			7	7	9																			

Note: X = not possible to identify

Table 5: Number of parts in software design for each participant

Design Representations

All designs were analysed irrespective of their size and ordering to determine the richness of the software designs. Each was categorised according to their visual predominance in one of the following five different groupings, namely:

1. Standard Graphical Representation (S): This was used to include recognised notations of software design. Ten different types were represented in the corpus: Architecture Diagram, Class Diagram, Class-Responsibility-Collaborator (CRC) Cards, Data Flow Diagram (DFD), Entity-Relationship Diagram (ER), Flow Chart, Graphical User Interface (GUI), Sequence Diagram, State Transition Diagram (STD) and Use Case Diagram.
2. Graphical (G): This category included diagrams of any form that were not recognised as standard notations of software design. Large sections of text were accepted in this category providing that they were considered refinements of items identified in the diagram. In some cases it was difficult to differentiate between *ad hoc* diagrams and standardized graphical representations. In order to characterise the latter category, detailed syntax was ignored and benchmarks defined. For example in order to be recognized as a Class Diagram, it was agreed that a representation should consist of a box conceptualising both data and functions.
3. Code or pseudo-code (C): This was used for any software design that included code segments such as assignments, iteration and selection.
4. Predominantly textual (T): This category was used for free text descriptions but allowed an occasional diagram if used for illustration: for example, graphical interface or report layout.
5. Mixed (M): This was used when there was no clear dominance between different styles. For example a participant might start with a textual description then proceed with a Class Diagram. If there was no connection between the descriptions and the identified classes then the category was Mixed (Text and Class Diagrams).

To ensure consistency the designs were all categorised by three of the authors and required consensus.

4.2.4 Results

Figure 14 and Table 6 show the results of this analysis; table 7 displays the data by institution. (Figure 14 and Table 6 include data from all 314 participants. However, although we were able to make a determination that the representation participant 01 from institution Q used was textual, that participant did not complete the design. Consequently, we were unable to answer any of the other questions under consideration: number of parts, hierarchy between parts, and interaction among parts. All tables relating to that data include only 313 participants).

In general, there is a progression in the formalisation of techniques that are applied by the participant groupings. First competency students tend to rely much more on textual representations (47%) compared with more formalised techniques (15%). At the other end of the scale, diagrammatic representation (50%) was the most frequent technique employed by educators. Graduating students were more evenly distributed among textual, graphical and standard representations. Only 11% of all participants chose to represent their software designs using code. The categorisations of software designs produced by

subject groups shows considerable variability across institutions. However small sample sizes at some of the institutions precludes meaningful statistical analysis.

Using the characterization data, two null hypotheses concerning the three participant groups FC, GS, and E were tested. Firstly, that there is no association between the three participant groups and their design representation. Using the chi-square test, the data supported the rejection of this null hypothesis at the alpha = .001 significance level.

A single combination student group (S) consisting of all student subjects (i.e., the union of FC and GS) was also considered. The second null hypothesis was that there was no association between groups S and E in their design representation. Using the chi-square test, the data supported the rejection of this null hypothesis at the alpha = .025 significance level.

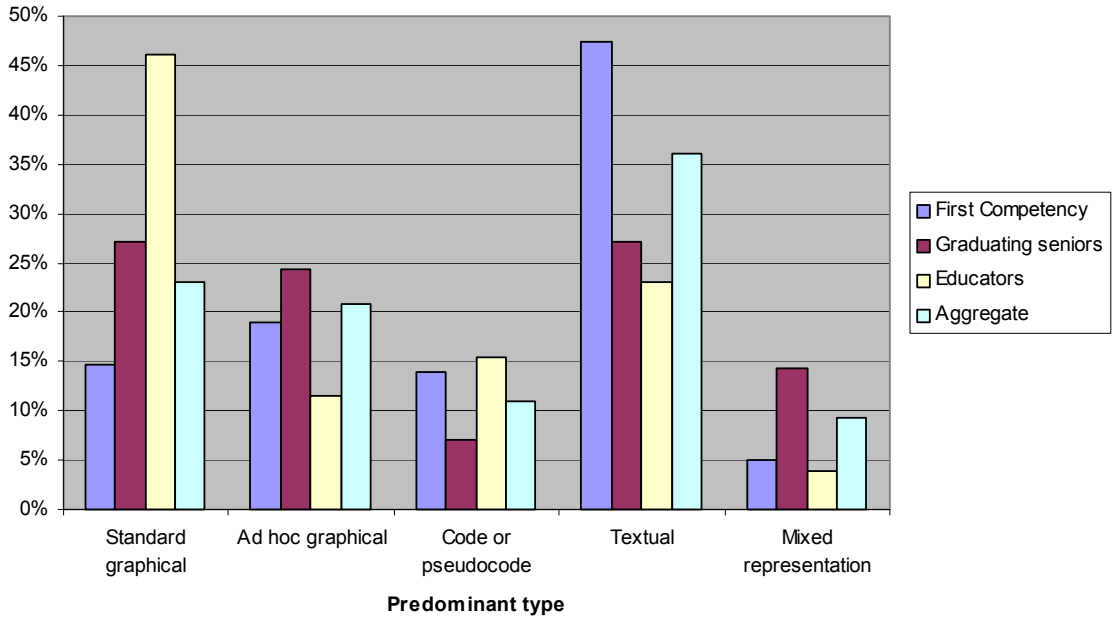


Figure 14: Design Representations According to Design

	Design Categorisation									
Participant Group	Standard		Graphical		Code		Textual		Mixed	
	Count	Percent	Count	Percent	Count	Percent	Count	Percent	Count	Percent
FC	20	14.71%	26	19.12%	19	13.97%	64	47.06%	7	5.15%
GS	43	28.67%	34	22.67%	11	7.33%	42	28.00%	20	13.33%
E	14	50.00%	3	10.71%	4	14.29%	6	21.43%	1	3.57%
Total	77	24.52%	63	20.06%	34	10.83%	112	35.67%	28	8.92%

Table 6: Design Representations

	Design Categorisation									
Institution	Standard		Graphical		Code		Textual		Mixed	
	Count	Percent	Count	Percent	Count	Percent	Count	Percent	Count	Percent
A	1	5.88%	2	11.76%	5	29.41%	5	29.41%	4	23.53%
C	1	14.29%	1	14.29%	0	0.00%	5	71.43%	0	0.00%
D	10	50.00%	2	10.00%	2	10.00%	4	20.00%	2	10.00%
E	2	16.67%	5	41.67%	2	16.67%	2	16.67%	1	8.33%
F	1	4.76%	4	19.05%	5	23.81%	11	52.38%	0	0.00%
H	7	58.33%	0	0.00%	1	8.33%	4	33.33%	0	0.00%
I	3	23.08%	4	30.77%	2	15.38%	4	30.77%	0	0.00%
J	4	23.53%	4	23.53%	3	17.65%	5	29.41%	1	5.88%
K	7	50.00%	2	14.29%	1	7.14%	4	28.57%	0	0.00%
L	9	42.86%	5	23.81%	1	4.76%	3	14.29%	3	14.29%
M	0	0.00%	3	15.00%	3	15.00%	11	55.00%	3	15.00%
N	6	28.57%	8	38.10%	1	4.76%	5	23.81%	1	4.76%
O	4	19.05%	4	19.05%	0	0.00%	9	42.86%	4	19.05%
P	6	28.57%	4	19.05%	1	4.76%	6	28.57%	4	19.05%
Q	0	0.00%	1	12.50%	0	0.00%	5	62.50%	2	25.00%
R	2	20.00%	3	30.00%	0	0.00%	5	50.00%	0	0.00%
S	9	52.94%	1	5.88%	0	0.00%	5	29.41%	2	11.76%
T	4	18.18%	6	27.27%	5	22.73%	7	31.82%	0	0.00%
V	1	16.67%	1	16.67%	0	0.00%	4	66.67%	0	0.00%
W	0	0.00%	2	18.18%	2	18.18%	7	63.64%	0	0.00%
Y	0	0.00%	1	33.33%	0	0.00%	1	33.33%	1	33.33%
Total	77	24.52%	63	20.06%	34	10.83%	112	35.67%	28	8.92%

Table 7: Design Representation by Institution

The use of hierarchical, nested, or grouping structures among parts, or even components in the design that were not labeled specifically as parts, was examined. Each researcher analysed the designs from their own institution in terms of grouping by answering the question "Did the design include any hierarchical, nested, or grouping structure of any kind: Yes (Y), No (N), Don't Know (D)?" For example, a diagram with boxes labeled "Pocket PC," "Alarm Handler," and "User Interface," all of which are labeled collectively as "User/Front End" would count as grouping. Table 8 shows that in the E group 46% of the participants included grouping in their designs, while 24% of the FC group and 27% of the GS group included grouping in their designs.

	Hierarchical Representation?					
Participant Group	Yes		No		Don't Know	
	Count	Percent	Count	Percent	Count	Percent
FC	32	23.70%	100	74.07%	3	2.22%
GS	41	27.33%	108	72.00%	1	0.67%
E	13	46.43%	15	53.57%	0	0.00%
Total	86	27.48%	223	71.25%	4	1.28%

Table 8: Inclusion of Hierarchy in the Design by Participant Groups

As shown in Table 9, there is a considerable difference among institutions; it ranges from a low of 5% of the participants of institution F who included grouping in their designs, to 86% of the participants of institution Q who did so.

	Hierarchical Representation?					
Institution	Yes		No		Don't Know	
	Count	Percent	Count	Percent	Count	Percent
A	2	11.76%	15	88.24%	0	0.00%
C	1	14.29%	5	71.43%	1	14.29%
D	7	35.00%	13	65.00%	0	0.00%
E	5	41.67%	7	58.33%	0	0.00%
F	1	4.76%	20	95.24%	0	0.00%
H	1	8.33%	11	91.67%	0	0.00%
I	5	38.46%	7	53.85%	1	7.69%
J	9	52.94%	8	47.06%	0	0.00%
K	4	28.57%	10	71.43%	0	0.00%
L	5	23.81%	15	71.43%	1	4.76%
M	5	25.00%	15	75.00%	0	0.00%
N	4	19.05%	17	80.95%	0	0.00%
O	7	33.33%	14	66.67%	0	0.00%
P	4	19.05%	17	80.95%	0	0.00%
Q	6	85.71%	0	0.00%	1	14.29%
R	1	10.00%	9	90.00%	0	0.00%
S	8	47.06%	9	52.94%	0	0.00%
T	7	31.82%	15	68.18%	0	0.00%
V	2	33.33%	4	66.67%	0	0.00%
W	1	9.09%	10	90.91%	0	0.00%
Y	1	33.33%	2	66.67%	0	0.00%

Table 9: Inclusion of Hierarchy in the Design by Institution

Two null hypothesis were considered. First that there is no association between the three participant groups and the use of grouping in their designs. Using the chi-square test, there is no significant difference among the participant groups and their use of grouping in their designs. Second, considering a single combination student group (S), there is no association between groups S and E in their use of grouping in their designs. Using the chi-square test, the data supported the rejection of this null hypothesis at the alpha = .025 significance level.

Also examined was whether the design contained an indication of interaction among the parts. In a similar way mentioned above, indication of interaction was analysed by each researcher by answering the question, "Are interaction between ay of the parts indicated, Yes(Y), No (N), Don't Know (D)?" For example, a diagram with two boxes, an arrow linking the two boxes, and an explanation that one box is providing information to the other box would count as interaction. Table 10 shows that in the E group 93% of the participants indicated interaction among the parts in their designs, while only 66% of the FC group and 81% of the GS group did so. Again, there is a noticeable difference among institutions, as shown in Table 11; it ranges from a low of 40% of the participants of institution M who indicated interaction in their designs, to 100% of the participants of institutions C, V, and Y who did so.

The null hypothesis for this analysis was that there is no association between the three participant groups and an indication of interaction in their designs. There is a statistically significant difference among all three participant groups. Using the chi-square test, the data supported the rejection of the null hypothesis at the alpha = .001 significance level. Furthermore considering the combined student group (S), the data supported the rejection, at the alpha - .025 significance level, of the null hypothesis that there is no association between groups S and E and an indication of interaction in their designs.

	Interaction Indicated?					
Participant Group	Yes		No		Don't Know	
	Count	Percent	Count	Percent	Count	Percent
FC	89	65.93%	44	32.59%	2	1.48%
GS	122	81.33%	23	15.33%	5	3.33%
E	26	92.86%	2	7.14%	0	0.00%
Total	237	75.72%	69	22.04%	7	2.24%

Table 10: Inclusion of Interaction in the Design by Participant Groups

	Interaction Indicated?					
Institution	Yes		No		Don't Know	
	Count	Percent	Count	Percent	Count	Percent
A	16	94.12%	1	5.88%	0	0.00%
C	7	100.00%	0	0.00%	0	0.00%
D	17	85.00%	3	15.00%	0	0.00%
E	11	91.67%	1	8.33%	0	0.00%
F	14	66.67%	7	33.33%	0	0.00%
H	10	83.33%	1	8.33%	1	8.33%
I	9	69.23%	4	30.77%	0	0.00%
J	14	82.35%	1	5.88%	2	11.76%
K	12	85.71%	2	14.29%	0	0.00%
L	20	95.24%	0	0.00%	1	4.76%
M	8	40.00%	10	50.00%	2	10.00%
N	14	66.67%	7	33.33%	0	0.00%
O	12	57.14%	9	42.86%	0	0.00%
P	10	47.62%	11	52.38%	0	0.00%
Q	6	85.71%	0	0.00%	1	14.29%
R	7	70.00%	3	30.00%	0	0.00%
S	13	76.47%	4	23.53%	0	0.00%
T	20	90.91%	2	9.09%	0	0.00%
V	6	100.00%	0	0.00%	0	0.00%
W	8	72.73%	3	27.27%	0	0.00%
Y	3	100.00%	0	0.00%	0	0.00%
Total	237	75.72%	69	22.04%	7	2.24%

Table 11: Inclusion of Interaction in the Design by Institution

The result from the study indicates that the ability to use grouping and interaction in software design increases with programming experience and exposure to these concepts in the curriculum. The data also indicates the degree to which these abilities are manifested in student software design.

The final aspect of systematic differences considered the time participants took to produce software designs. The results showed that graduating students on average take longer (38 minutes) to complete their software designs than do first competence students (37 minutes). Whilst the difference between those groups was not significant, educators took on average 48 minutes.

4.3 Design criteria prioritisation

One of this study's focal questions is whether students recognize different criteria within the design process. This was motivated by discussion with educators and papers such as (CMM Correspondence Group, 1997), that suggest that there are particular criteria that should be considered when doing software design. The particular focus here is on the relative importance of different criteria—whether these rankings (prioritizations) are the same for various student groups, whether they are the same as educators, and whether they vary over different tasks and contexts.

By examining these prioritizations across participant groups, it could be possible to see how (or whether) these are learned through the curriculum—do the rankings of first competency students differ from those of graduating seniors, and are the rankings of graduating seniors different from those of educators. These rankings were collected immediately after, and are set in the context of, the decomposition task in Appendix A.

4.3.1. Questions

. Specifically, the following questions were addressed in the data collection and analysis:

- What are the differences, if any, among the prioritisations done by different participant groups?
- How closely do the selection patterns correlate with various distinguishing characteristics within each participant group?
- To what degree do members of each participant group (FC, GS, and E) agree among themselves on how they rank the criteria in each scenario?
- How do each group's criteria rankings vary across scenarios?

4.3.2. Data collection and measurement scales

Data were collected by presenting the subject with a set of 16 cards, each with a short phrase describing a design criterion (Appendix C). They were asked to partition the set of cards into three groups: the five most important, the five least important, and the others. They partitioned the cards in this way four times, one for each of four scenarios:

1. creating the design individually (*current task*),
2. performing the task as part of a team (*task in team*),
3. performing the task by themselves in 24 hours (*time pressure*), and
4. designing software with a long expected lifetime (*longevity*).

These were done in sequence, without knowing what the successive scenarios were going to be.

Each partition is considered as one observation. For example, one observation might be most important ={1,3,6,7,9}, least important ={2,4,5,10,16}, (other={8,11,12,13,14,15} by default). To understand what sort of analyses might be applicable, it is necessary to understand the measurement scales involved. The criteria (card numbers 1-16, or category names) are *nominal* scaled data: the card numbers are only identifiers, and have no other significance. The importance chosen for an individual criterion (most, least, other) is *ordinal* scaled data: *most* is greater than *other*, which is greater than *least*. An observation is simply a partition of the criteria into three sets: 5 *most*, 5 *least*, and 6 *other*.

Participants' questions and comments during the prioritisation task were also collected. These qualitative data were used in interpreting the results of the statistical analysis of numerical data.

4.3.3. Data analysis

The data collected for each prioritisation were collected into frequency counts for each participant group and scenario. These data are shown as a set of bar charts in Appendix D, and show each group's rankings for the different scenarios. These suggest that each group prefers certain criteria over others within a given scenario, and that these preferences are not identical across participant groups, and not identical across scenarios. To examine whether these suggested differences are significant, we ran a series of comparisons, with associated significance tests where appropriate.

Between-group agreement

Two series of tests were done to evaluate whether there were differences between the groups on how criteria were ranked: one series that examined all of the criteria at once, and one that looked at the criteria one at a time.

The first series of tests are Chi-square tests for differences in probabilities (Conover, 1971), where the data (for either most or least important) are arranged into a *g by 16* contingency table, where *g* is the number of participant groups being tested, 16 is the number of criteria, and the value in each cell is the number of times that criterion was picked by that group. The layout of such a table for most important, with marginal annotations, is given in Table 12. The test is based on cell probabilities: the probability of cell i,j is the probability that a randomly chosen observation from group i includes criterion j in the most important group. The null hypothesis is that all probabilities in the same column are equal: that is, each group has the same preference for each criterion.

	Criterion_1	Criterion_2	…	Criterion_{16}	*Totals*
FC	$O_{FC,1}$	$O_{FC,2}$	...	$O_{FC,16}$	n_{FC}
GS	$O_{GS,1}$	$O_{GS,2}$	...	$O_{GS,16}$	n_{GS}
E	$O_{E,1}$	$O_{E,2}$	...	$O_{E,16}$	n_E
Totals	C_1	C_2	...	C_{16}	N

Table 12 Arrangement of data in 3x16 contingency table. $O_{i,j}$ is the number of observations from group *i* that have Criterion_j rated most important, C_i is the sum of cells in column j, n_i is the sum of the cells in row i. Under the null hypothesis, the expected value in $\text{cell}_{i,j}$ is $n_i\, c_j/N$.

These tests were run on five sets of data (each scenario, plus pooled scenarios) for both most and least important criteria, with a alpha = 0.05. We found significant differences among the groups for both the most and least important criteria, for the pooled scenarios and the longevity scenarios (no difference when looking at current, team, and time pressure scenarios individually). As further post hoc analysis, we looked at the residual cell differences. Considering the most important criteria, pooled scenarios, we found significant differences (residual > 2.0) for *Coupling* (chosen less by FC, more by GS and E), *Engineering* (chosen more by GS, less by E), and *Clarity* (chosen less by E). Under the longevity scenario, we saw similar significant residual differences for *Coupling* and *Clarity*. Considering the least important criteria, pooled scenarios, we found significant differences for *Coupling* (chosen less by GS, more by FC), and eight of the criteria for E:

Intelligibility, Explainability, Recognition of structure, Input re-use, and *Clear functionality* were chosen less often, and *Re-usability, Maintainability*, and *Engineering* were chosen more often. Under the longevity scenario, the only significant residuals were for *Maintainability* and *Engineering* (chosen more often by E).

As most of the significant differences were for group E, and our primary interests are differences between groups FC and GS, we ran the same set of tests on FC and GS data only. These results were consistent with the others: we found significant differences for most important under pooled scenarios, least important under pooled scenarios, and most important under longevity; in each of these, the only significant residual difference was found for *Coupling:* GS rated it relatively more important than FC in these three cases.

We also performed an alternative set of tests for each individual criterion, comparing the participant groups under each scenario. For these tests, we used a 3x3 contingency table, with columns corresponding to the classification (most, other, least), and rows corresponding to groups as before. These comparisons were run with an alpha of 0.05 for each test.

The results of these tests showed significant differences in 16 of the 64 comparisons: *Coupling* (for *task in team, time pressure*, and *longevity*), *Input re-use* (for *current task, task in team*, and *time pressure*), *Re-usability* (for *task in team* and *time pressure*), *Recognition of structure* (for *task in team* and *longevity*), *Clarity* (for *current task* and *longevity*), and *Cohesion, Intelligibility, Maintainability,* and *Engineering* (for *longevity*). To try to isolate the differences between the two student groups, we ran these tests on those data alone, and found significant differences for *Coupling* and *Recognition of structure* (for the *task in team* and *longevity* scenarios).

Summarizing the between group test results:

- Between FC and GS, the only differences were found for *Coupling* (both sets of tests) and *Recognition of structure* (using individual criterion tests, *task in team* and *longevity* scenarios).
- Across all groups, E was significantly different from the others for many of the individual criteria under at least one scenario.

Agreement under distinguishing characteristic

One of the hypotheses was that students in different "performance buckets" would have different criteria preferences. We tested this for the GS participant group (where the performance bucket classification should be most reliable). Due to the low numbers of subjects in the top and bottom performance bucket, we divided the pool into buckets 1 and 2 (GPA = 3.7-4.0), bucket 3 (GPA = 2.7-3.7), and buckets 4 and 5 (GPA = 0-2.7), then ran comparisons using a 3x16 contingency table as above (replacing groups by performance class), for pooled scenarios plus individual scenarios. We found no significant differences between these groups.

Within-group agreement

There are a number of possible ways to measure within-group agreement—proportion of the observations that are the same as the most popular answer, proportion of observations that agree on the classification of the most popular (or key) criteria, etc. Table 13 shows the percent of all observed classifications that are in the top five chosen by a group: the number of choices (for most or least important) that were in the top five answers divided by the total number of choices. This number can range from 30.25% (all criteria chosen in equal proportion) to 100% (all observations agree completely).

	Pooled	Current task	Task in team	Time pressure	Longevity
FC	**55**	**48**	**53**	**60**	**57**
GS	**56**	**48**	**54**	**66**	**57**
E	**59**	**56**	**57**	**65**	**56**

Table 13 Percent of all observed classifications that are in the top five chosen, by group and scenario.

These data suggest that while there is a good deal of agreement, it is not complete (although the choices are clearly not random, as the percentages are all well over the minimum possible). If we run a Chi-square test for difference of probabilities on the underlying count data, we reject (at alpha = 0.05) the hypothesis that there are no differences across groups and scenarios, that is, the probability of a chosen criterion being one of the top five chosen is not independent of scenario and participant group. A closer examination of the residuals confirms what the above table suggests: the proportion of choices in the top five are significantly greater than expected under the time pressure scenario, and the proportion of choices in the top five are significantly less than expected in the current scenario for the FC and GS groups.

Individual variation across scenarios (by group)

One of the things we wanted to assess is the degree to which individuals adjust the rankings of their design criteria when faced by different situations. It has been observed (Adams, Turns, & Atman, 2003) that expert designers adapt the way they approach problems to match task constraints; we decided to test whether our observations across groups would also exhibit this behavior. The data consist of prioritizations across four different scenarios, grouped by individual, so individual responses can be examined. The measure used for these responses was calculated as follows:

1. Each observation was expressed as a 16-element vector (positions corresponding to criteria); each element was given the value –1 if the criterion was in *least*, 1 if it was in *most*, and 0 otherwise. This value assignment respects the order of *least* < *other* < *most* (important).
2. The mean of the four scenario vectors is calculated.
3. We calculate the squared Euclidian distances between the mean vector and each observation.
4. We calculate the response as the sum of these 4 distances.

This is equivalent (within a multiplicative constant) to calculating the covariance matrix of the four vectors and summing the diagonal terms, or summing the variances of each criterion values.

These responses were compared among the groups by running a one-way analysis of variance. The result was that they were significantly different across groups at an alpha of 0.05 (calculated significance $p<.0005$). Examining closer, the average responses for the three groups are FC = 23.3, GS = 23.5, and E = 17.9: the significance of the difference here is due to the lower value for E (this result was verified by running t-tests on FC vs. GS (not significant) and E vs. FC and GS (significant)).

The result: there is no difference in the amount that groups FC and GS (the students) changed their prioritisations over these four scenarios, and the amount of change is significantly greater than that observed with group E (the educators).

Applicability of the statistical tests

The statistical tests used assume that certain things are true about the data; some of these assumptions are not strictly met. For the Chi-square tests based on counts, it is assumed that the measurement scale is at least nominal, each observation is counted in one cell, and that the observations are independent. In all of the tests where we use total counts across criteria (where we are counting the number of times a criterion is named as most or least important), we violate the independence assumption: each observation is treated like 5 observations, and since any criterion can only be counted once in that group of 5, they are not independent. If the number of observations is large relative to 5, this should not have much of an effect.

The tests done on individual criteria (counting as *most, least,* or *other*) do not violate the independence assumption. There is a problem, however, in running a series of individual tests for the criteria: while the alpha value (probability of rejecting the null hypothesis if it is true) holds for each criterion, the likelihood of rejecting some null hypothesis that is true goes up with the number of criteria tested.

There are a number of statistical tests available that use ranks of ordinal data. The problem with the data collected here is that the criteria within an observation have only three possible values, so the ranks end up with a lot of ties: 5 tied for *most*, 5 tied for *least*, and 6 tied for *other*; ranked tests generally require that ties be quite rare for the tests to apply.

4.3.5. Conclusions

Analyses were done of students' and educators' selection patterns in the criteria prioritisation. Each group changed its prioritizations in response to changing design contexts. The size of that response, as measured by the individual variation across contexts, showed a significant difference between students and educators. Surprisingly, students were more flexible, and adapted their criteria rankings to the context of the task to a greater degree than did educators.

Few statistically significant differences were found between FC and GS students—the only differences detected were relative to *Coupling* and *Recognition of structure*. No statistically significant difference was found between high- and low-performing GS students.

We note a potential source of bias. The design criteria were each meant to represent a software engineering concept or principle. The way each was phrased may have lent subtle shades of meaning that differ with other phrasing.

4.3.6. Future work

Future work involves the study of the prioritisation data for some subgroups of the FG, GS, and E groups. Specifically, it will be examined whether other factors make a significant difference with respect to the results, factors including gender, age, number of computer science classes taken, time to perform the decomposition task, number of programming languages known, and institution.

Summary

Research on student software design was undertaken over 21 institutions in the academic year 2003/4. The study generated large sets of both qualitative and quantitative data. The corpus was analysed to investigate three interests: did students perceive and respond to

ambiguity in the requirements, were there differences in their description and representation of their solutions and were there differences in the relative priority they gave different design criteria. In all cases, the questions included comparison between sub-populations of "first competency" students, graduating students and educators.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0243242. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The design brief was developed from a classroom exercise of B.J. Fogg.

The Design Criteria Prioritisation Task was informed by Newstetter and McCracken's study on 'novice conceptions of design' (Newstetter & McCracken, 2001) and previous work on the *Bootstrapping* project (Petre et al., 2003)

The Department of Mathematics and Computer Science, Berry College, USA provided participant support.

Thanks to Leigh Waguespack for assistance with statistical analysis, to Janet Rountree for assistance with data gathering and transcription, to Karen Furuya for assistance with transcription, to Gary Marston for assistance with image scanning and to Jennifer Burley for student GPA data-gathering.

References

Adams, R. S., Turns, J., & Atman, C. J. (2003). *What Could Design Learning Look LIke?* Paper presented at the Expertise in Design: Design Thinking Research Symposium 6, Sydney, Australia.

Atman, C. J., Chimka, J. R., Bursic, K. M., & Nachtmann, H. L. (1999). A comparison of freshman and senior engineering design processes. *Design Studies, 20*, 131-152.

Boehm, B. W. (1981). *Software Engineering Economics*: Prentice Hall.

Bogush, L. L., Turns, J., & Atman, C. J. (2000). *Engineering design factors: how broadly do students define problems?* Paper presented at the ASEE/IEEE Frontiers in Education, Kansas City.

Chase, W. G., & Simon, H. A. (1973). Perception in Chess. *Cognitive Psychology, 4*, 55-81.

Christiaans, H. H. C., & Dorst, K. H. (1992). Cognitive models in industrial design engineering. *Design Theory and Methodology, 42*(131-140).

CMM Correspondence Group. (1997). *Software Product Engineering (draft)*: Carnegie Mellon University.

Conover, W. J. (1971). *Practical Nonparametric Statistics*. New York: Wiley.

Curtis, B. (1990). *Empirical Studies of the Software Design Process.* Paper presented at the Human-Computer Interaction - INTERACT '90.

Détienne, F. (2001). *Software Design - Cognitive Aspects* (F. Bott, Trans.): Springer Verlag.

Gause, D., & Weinberg, G. (1982). *Are Your Lights On?* Cambridge, MA: Winthrop Publishers.

Goel, V., & Pirolli, P. (1992). The Structure of Design Problem Spaces. *Cognitive Science, 16*, 395-492.

Guindon, R., Krasner, H., & Curtis, B. (1987). *Breakdowns and processes during the early activities of software design by professionals*. Paper presented at the Empirical Studies of Programmers: Second Workshop.

Jeffries, R., Turner, A. A., Polson, P. G., & Atwood , M. E. (1981). The processes involved in designing software. In J. Anderson (Ed.), *Cognitive Skills and their Acquisition*: Lawrence Erlbaum Associates.

Kaplan, S., Gruppen, L., Levanthal, L. M., & Board, F. (1986). *The Components of Expertise: a Cross-Disciplinary Review*. Ann Arbor: University of Michigan.

Maloney, D. P. (1994). Research in Problem Solving: Physics. In Gabel (Ed.), *Handbook of Research on Software and Teaching* (pp. 327-354). New York: Macmillan.

Mayer, B. (1997). *Object oriented software construction* (2 ed.): Prentice Hall.

McCracken, W. M. (2004). Research on Learning to Design Software. In S. Fincher & M. Petre (Eds.), *Computer Science Education Research* (pp. 155-174). Lisse: Routledge Falmer.

McCracken, W. M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., et al. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *SIGCSE Bulletin, 33*(4), 125-180.

Newstetter, W. C., & McCracken, W. M. (2001). Novice conceptions of design: Implications for the design of learning environments. In C. M. Eastman, W. M. McCracken & W. C. Newstetter (Eds.), *Design Knowing and Learning: Cognition in Design Education*. Amsterdam: Elsevier.

Petre, M., Fincher, S., Tenenberg, J., Anderson, R., Anderson, R., Bouvier, D., et al. (2003). *"My criterion is: Is it a Boolean?": A card-sort elicitation of students' knowledge of programming constructs* (Computing Laboratory Technical Report No. 6-03). Canterbury: University of Kent.

Riel, A. J. (1996). *Object-Oriented Design Heuristics* (1 ed.): Addison Wesley.

Rowland, G. (1992). What do instructional designers actually do? *Performance Improvement Quarterly, 5*(2), 65-86.

Rumbaugh, J., Jacobson, I., & Booch, G. (1998). *The Unified Modeling Language Reference Manual*: Addison Wesley.

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering, 10*(5), 595-609.

Wiedenbeck, S. (1985). Novice/expert differences in programming skills. *International Journal of Man-Machine Studies, 23*, 383-390.

Yourdon, E., & Constantine, L. (1978). *Structured Design*: Prentice Hall/Yourdon Press.

Appendix A: Design Brief
Getting People to Sleep

In some circles sleep deprivation has become a status symbol. Statements like “I pulled another all-nighter” and “I’ve slept only three hours in the last two days” are shared with pride, as listeners nod in admiration. Although celebrating self-deprivation has historical roots and is not likely to go away soon, it’s troubling when an educated culture rewards people for hurting themselves, and that includes missing sleep.

As Stanford sleep experts have stated, sleep deprivation is one of the leading health problems in the modern world. People with high levels of sleep debt get sick more often, have more difficulties in personal relationships, and are less productive and creative. The negative effects of sleep debt go on and on. In short, when you have too much sleep debt, you simply can’t enjoy life fully.

Your brief is **to design a "super alarm clock" for University students** to help them to manage their own sleep patterns, and also to provide data to support a research project into the extent of the problem in this community. You may assume that, for the prototype, each student will have a Pocket PC (or similar device) which is permanently connected to a network.

Your system will need to:

- Allow a student to set an alarm to wake themselves up.
- Allow a student to set an alarm to remind themselves to go to sleep.
- Record when a student tells the system that they are about to go to sleep.
- Record when a student tells the system that they have woken up, and whether it is due to an alarm or not (within 2 minutes of an alarm going off).
- Make recommendations as to when a student needs to go to sleep. This should include "yellow alerts" when the student will need sleep soon, and "red alerts" when they need to sleep now.
- Store the collected data in a server or database for later analysis by researchers. The server/database system (which will also trigger the yellow/red alerts) will be designed and implemented by another team. You should, however, indicate in your design the behaviour you expect from the back-end system.
- Report students who are becoming dangerously sleep-deprived to someone who cares about them (their mother?). This is indicated by a student being given three “red alerts" in a row.
- Provide reports to a student showing their sleep patterns over time, allowing them to see how often they have ignored alarms, and to identify clusters of dangerous, or beneficial, sleep behaviour.

In doing this you should (1) produce an initial solution that someone (not necessarily you) could work from (2) divide your solution into not less than two and not more than ten parts, giving each a name and adding a short description of what it is and what it does – in short, why it is a part. If important to your design, you may indicate an order to the parts, or add some additional detail as to how the parts fit together.

Appendix B: Design Criteria

At SIGCSE 2003, Sally Fincher and Marian Petre interviewed a number of eminent CS educators, asking them what they would like to know about:

- what their students know, understand, or experience about software design
- what criteria they would apply to determine these things
- what criteria they would like their students to apply in assessing alternative software designs;
- on what dimensions they would vary alternatives if they were presenting alternative software designs for comparison and critique.

From these the researchers identified a collection of software design criteria identified by educators as being of interest or importance and extracted the 16 most prominent. Most were expressed by the educators using single-word, often technical terms, such as *coupling*, *encapsulation*, and *intelligibility*. Fincher and Petre's experience with a previous study (Petre et al, 2003) was that students were often unable to 'unpack' such professional terms, and so the researchers expressed each of the criteria as a descriptive phrase. They checked the descriptors with three CS educators, presenting them with both the descriptors and the original terms and asking them to match the two—which they were able to do accurately. They also presented one educator with just the phrases and asked him to express them as single or double word terms. He was able to do this, and his terms matched theirs. Here, the "short-form" phrases appear in square brackets, following each descriptive phrase.

1. Hiding the internal workings of each part of the solution from the user, presenting them with a simple interface to its functionality. [Encapsulation]

2. Knowing how each part of the solution could be implemented. [Implementability]

3. Making sure related things appear together. [Cohesion]

4. Making sure that un-related things are linked via a narrow (internal) interface. [Coupling]

5. Making sure the design is made up of appropriately-sized "chunks". [Chunking]

6. Being able to explain what each part of the solution is, and what it does, to yourself. [Intelligibility]

7. Being able to explain what each part of the solution is, and what it does, to others. [Explainability]

8. Constructing a solution using the simplest thing that gets the job done. [Parsimony]

9. Working to achieve a solution of maximum generality. [Re-usability]

10. Ensuring that the parts which make up the solution map onto the structure of the problem. [Recognition of structure]

11. Designing so that someone else can implement the solution with little (or no) additional information or domain expertise. [Clarity]

12. "Sanity-checking" the solution, by checking back to the specification. [Design-phase testing]

13. Designing a system that can be easily maintained. [Maintainability]

14. Considering the technological implementation (target platform or device) and designing for efficient use of that resource. [Engineering]

15. Using ideas that I know work. [Input re-use]

16. Expressing the functionality clearly. [Clear functionality]

Appendix C: requirements generated to identify expression of ambiguity

1. Allow a student to set an alarm to wake them up.

2. Allow a student to set an alarm to remind them to go to sleep.

3. Record when a student tells the system that they are about to go to sleep.

4. Record when a student tells the system that they have woken up, and whether it is due to an alarm or not (within 2 minutes of an alarm going off)

5. Make recommendations as to when a student needs to go to sleep. This should include "yellow alerts" when the student will need to sleep soon, and "red alerts" when they need to sleep now.

6. Store the collected data in a server or database for later analysis by researchers. The server/database system (which will also trigger the yellow/red alerts) will be designed and implemented by another team. You should, however, indicate in your design the behaviour you expect from the back-end system.

7. Report students who are becoming dangerously sleep-deprived to someone who cares about them (their mother?). This is indicated by a student being given three "red alerts" in a row.

8. Provide reports to a student showing their sleep patterns over time, allowing them to see how often they have ignored alarms, and to identify clusters of dangerous, or beneficial, sleep behaviour.

9. Wake the student up.

Appendix D: Design criteria selection within participant groups

For every figure in this appendix, the Y axis is % of sub-population i.e. FC, Grad and Educator. The X axis is the critieria number as given in appendix B. So, for figure 15 the left-most bar represents that 50% of the graduating students put criterion one in the "most important" category for the first task.

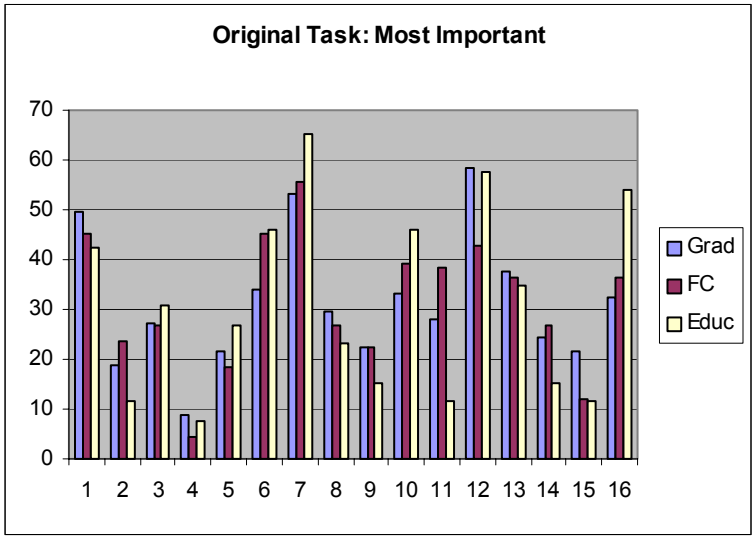


Figure 15. The most important design criteria selected by each group for the original task

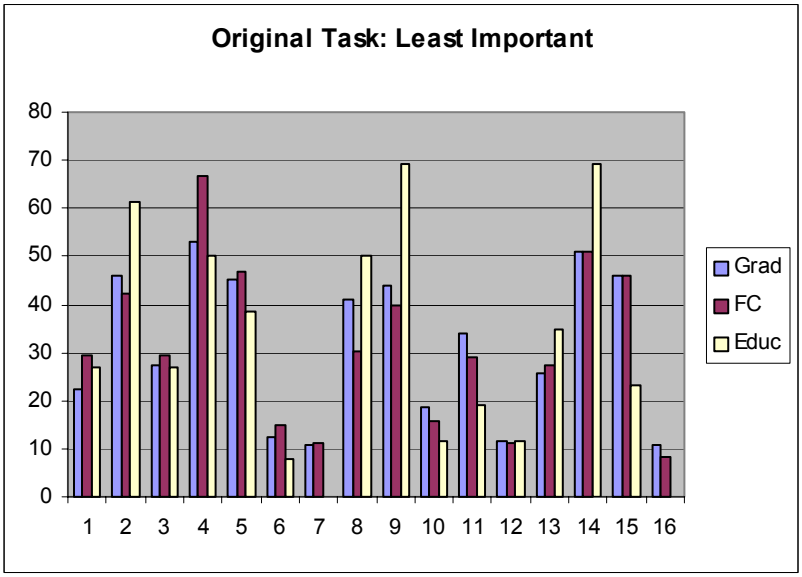


Figure 16. The least important design criteria selected by each group for the original task

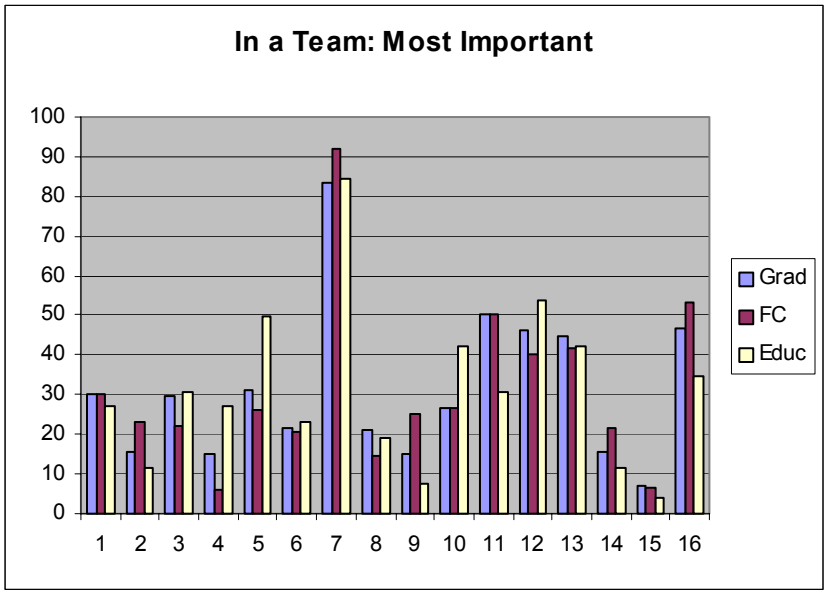


Figure 17. The most important design criteria selected by each group for the team scenario

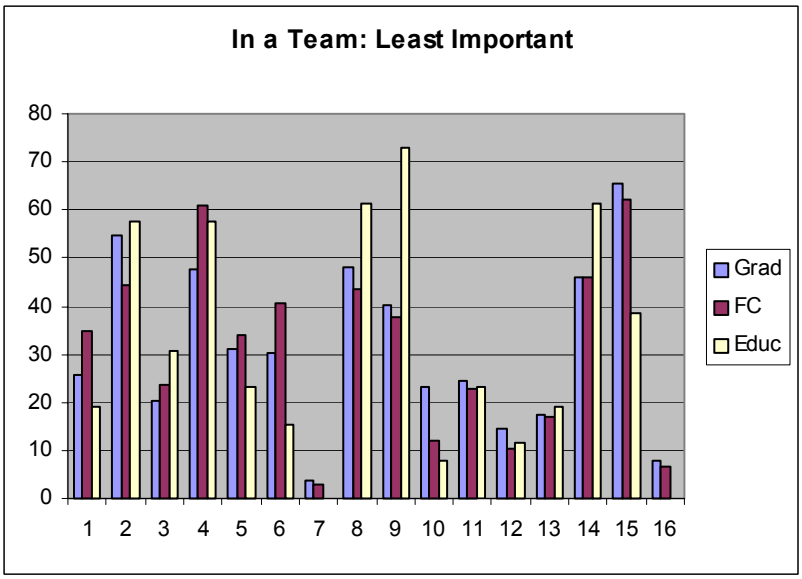


Figure 18 The least important design criteria selected by each group for the team scenario

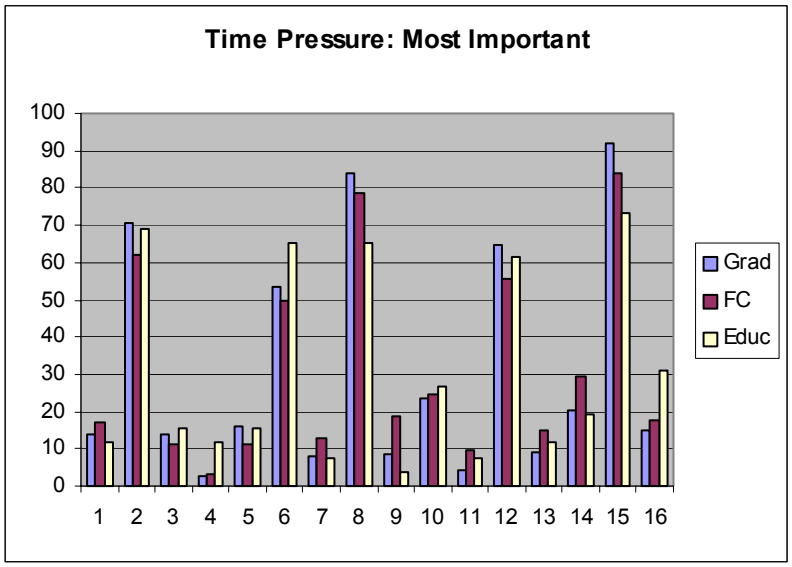


Figure 19. The most important design criteria selected by each group for the time pressure scenario

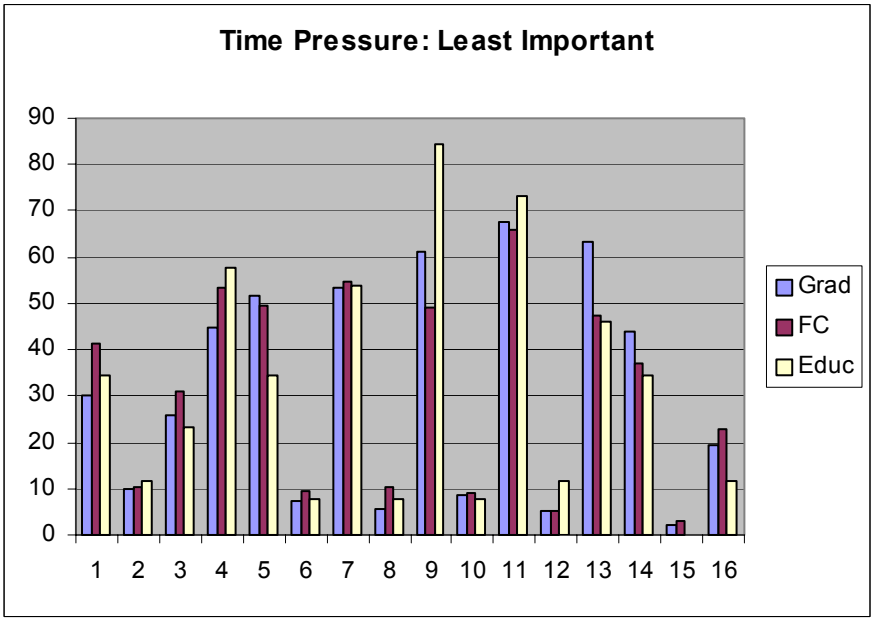


Figure 20 The least important design criteria selected by each group for the time pressure scenario

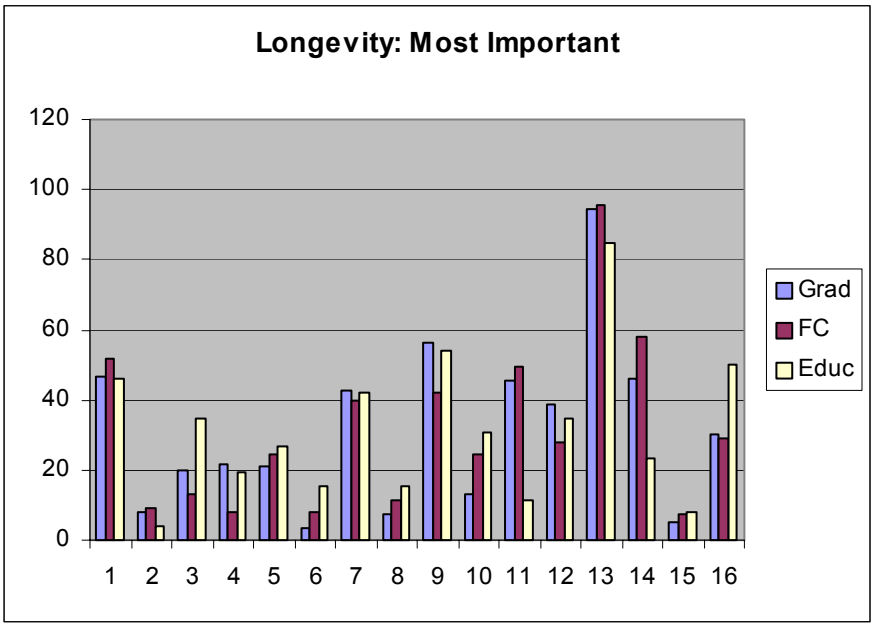


Figure 21. The most important design criteria selected by each group for the longevity scenario

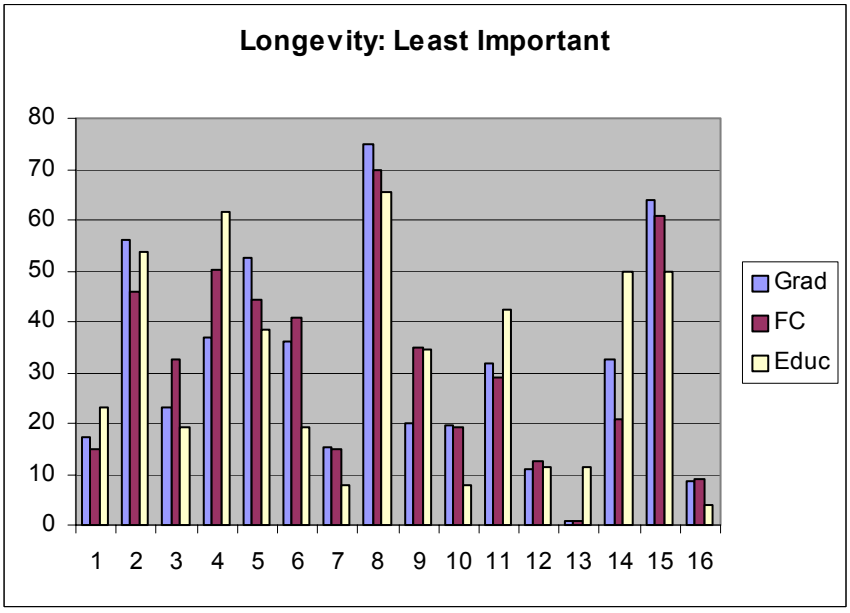


Figure 22. The least important design criteria selected by each group for the longevity scenario

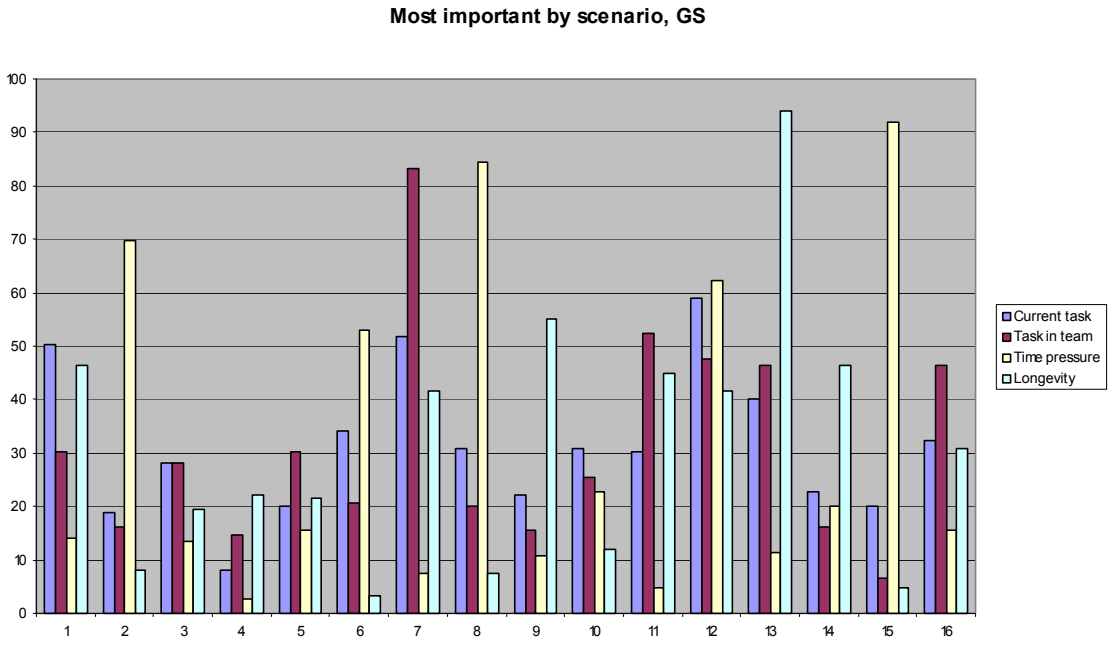


Figure 23. The GS students' most important design criteria, by different scenarios

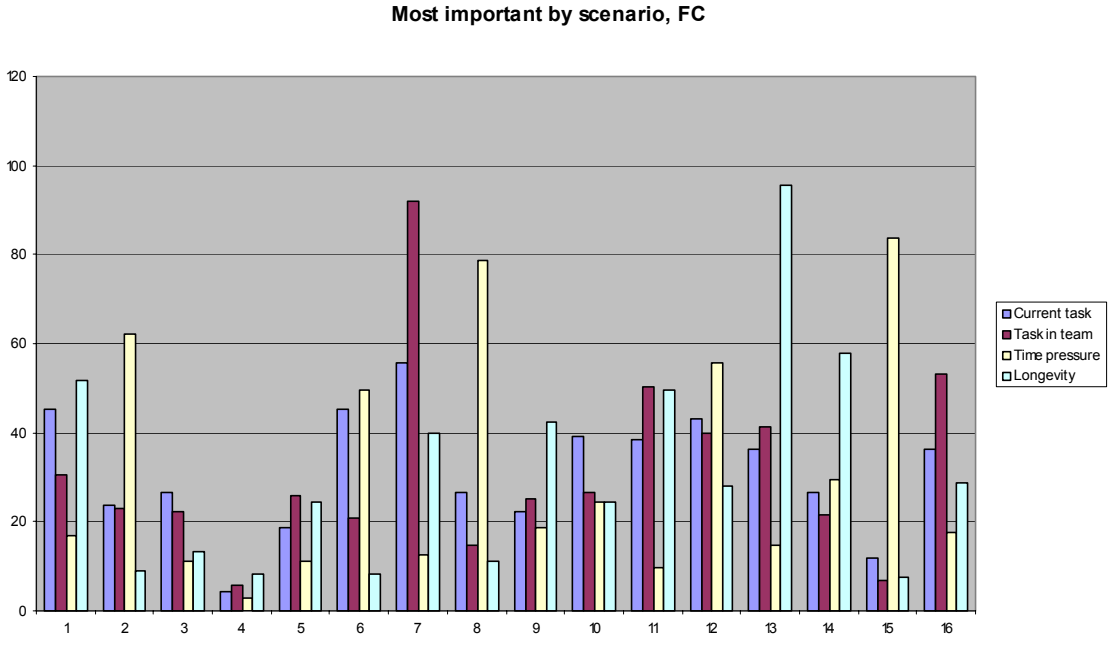


Figure 24. The FC students' most important design criteria, by different scenarios