Article

A multi-institutional, multinational study of programming concepts using card sort data

Kate Sanders, Sally Fincher, Dennis Bouvier, Gary Lewandowski, Briana Morrison, Laurie Murphy, Marian Petre, Brad Richards, Josh Tenenberg, Lynda Thomas, Richard Anderson, Ruth Anderson, Sue Fitzgerald, Alicia Gutschow, Susan Haller, Raymond Lister, Renée McCauley, John McTaggart, Christine Prasad, Terry Scott, Dermot Shinners-Kennedy, Suzanne Westbrook and Carol Zander

Kate Sanders, *Mathematics and Computer Science, Rhode Island College, USA*
Sally Fincher, *Computing Laboratory, University of Kent, UK*
Dennis Bouvier, *Department of Electrical and Computer Engineering, Parks College of Engineering, Aviation, and Technology, Saint Louis University, USA*
Gary Lewandowski, *Mathematics and Computer Science Department, Xavier University, Cincinnati, USA*
Briana Morrison, *Department of Computer Science, Southern Polytechnic State University, USA*
Laurie Murphy, *Computer Science and Computer Engineering Department, Pacific Lutheran University, USA*
Marian Petre, *Computing Department, Open University, UK*
Brad Richards, *Department of Computer Science, Vassar College, USA*
Josh Tenenberg, *Computing and Software Systems, Institute of Technology, University of Washington, Tacoma, USA*
Lynda Thomas, *Department of Computer Science, University of Wales, Aberystwyth, UK*
Richard Anderson, *Computer Science and Engineering, University of Washington, Seattle, USA*
Ruth Anderson, *Department of Computer Science, University of Virginia, USA*
Sue Fitzgerald, *Information and Computer Sciences, Metropolitan State University, USA*
Alicia Gutschow, *Information Technology and Computer Science, Blue Ridge Community College, Virginia, USA*
Susan Haller, *Computer Science Department, University of Wisconsin – Parkside, USA*
Raymond Lister, *Department of Software Engineering, University of Technology, Sydney, Australia*
Renée McCauley, *Department of Computer Science, College of Charleston, USA*
John McTaggart, *Department of Mathematics and Computer Science, Drake University, USA*
Christine Prasad, *School of Computing and Information Technology, UNITEC, New Zealand*
Terry Scott, *Department of Mathematical Sciences, University of Northern Colorado, USA*
Dermot Shinners-Kennedy, *Department of Computer Science and Information Systems, University of Limerick, Ireland*
Suzanne Westbrook, *Department of Computer Science, University of Arizona, USA*
Carol Zander, *Computing and Software Systems, University of Washington, Bothell, USA*

Abstract: *This paper presents a case study of the use of a repeated single-criterion card sort with an unusually large, diverse participant group. The study, whose goal was to elicit novice programmers' knowledge of programming concepts, involved over 20 researchers from four continents and 276 participants drawn from 20 different institutions. In this paper we present the design of the study and the unexpected result that there were few discernible systematic differences in the population. The study was one of the activities of the National Science Foundation funded Bootstrapping Research in Computer Science Education project (2003).*

Keywords: computer science education research, card sort, elicitation, programming concepts

1. Introduction

This paper presents a case study of the use of a repeated single-criterion card sort with an unusually large, diverse participant group. The study was designed to explore the nature and structure of students' knowledge about programming constructs. Educators know which concepts they teach, but not what students internalize about those concepts nor what conceptual structures students build from them. To illustrate: we might ask if students have a meaning for 'tree'. Which concepts do they group with 'tree', and what name do they give the group? If they group 'tree' with 'list' and 'array' and call the group 'data structures', what other groups of concepts do they associate with 'data structures'? We were interested in whether we could detect discernible differences in these concept constructs between defined populations, e.g. men and women, students and educators, different languages of programming instruction.

The researchers were all experienced, college-level, computer science educators, from a wide range of institutions that used a variety of pedagogic approaches to teach programming. Researchers collected data from their own institutions, following a standard protocol.

In the rest of the paper, we detail the case study. Section 2 discusses the study design, Section 3 some analysis mechanisms we used and Section 4 the findings these mechanisms yielded. In Section 5 we discuss the findings and examine possible limitations of the study. Section 6 concludes the paper.

2. Study design

[Card-sorting] . . . can provide insight into users' mental models, illuminating the way that they often tacitly group, sort and label tasks and content within their own heads (Rosenfeld & Morville, 2002)

The primary method used in this investigation was a repeated single-criterion card sort (Rugg & McGeorge, 1997) designed to elicit participants' construction of programming concepts. We chose this method for several reasons.

- Because of the quantity of participant institutions and diversity of researcher and student populations, it was important to choose a method that was not constrained by any programming task or by the syntax of a particular programming language.
- We could not rely on any one research partner having the skills or background of any other, so we sought a participant-focused rather than researcher-focused technique. Card sorting allows a large, geographically diverse group of researchers to each collect data at their own institutions, following a standard protocol.
- Card sorting is not compromised by the different backgrounds of the participants (with regard to institution, first programming language, age, gender etc.). The observations of Martine (2000) that card sorts (and associated co-occurrence matrix analysis) would allow comparison of responses were compelling in this respect.
- There is evidence to suggest that the way in which participants *organize* concepts reflects their mental representation of the way these concepts are related. Elicitation of internal conceptual structures is problematic because it requires plausible, observable intermediate representation; card sorting may provide such a representation.
- Similarly, the criteria identified in repeated single-criterion card sorting may reflect the participants' meta-knowledge.
- The concepts to be categorized – e.g. variable, method and array – are not necessarily ordered along a scale, making some other knowledge elicitation approaches (such as repertory grids) inappropriate (Rugg & McGeorge, 1997).
- There is a history of sorting techniques being used to investigate programming concepts. Adelson (1981) gave novice and expert programmers randomly ordered lines of computer code and observed how they recalled the code and in what proximity the lines were recalled. The proximity of the lines' recall was taken to represent subjects' imposition of structure on the unstructured data. Davies *et al.* (1995) asked expert and novice computer programmers to sort code fragments into categories that had meaning for them, in order to obtain knowledge about relationships the programmers identified among program components.
- Finally, a recent resurgence of interest in the use of card sorting for usability analysis of interfaces and information architectures (particularly of Websites) meant that there were a number of freely available tools for conducting and analysing card sorts.

Whilst Adelson and Davies *et al.* had used pieces of code as stimuli, we developed a deck of 26 cards each containing a 'minimalist' one-word prompt for a programming concept. These prompts included function, scope, type, method, list, loop, procedure, recursion, expression, dependency, choice, tree, object, state, thread, decomposition, encapsulation, iteration, abstraction, parameter, array, if-then-else, variable, event, Boolean, constant. The prompts were drawn from programming textbooks, from papers on program categorizations and from lists solicited from programming experts and programming educators. A pilot study was conducted with seven participants from two locations.

Each participant was presented with the set of 26 cards. We asked the participants to sort the cards into their own categories, using a single criterion. Participants were asked to provide names for each category and for the overall criterion by which the cards were sorted. For example, a participant might sort all cards based on the criterion 'memory storage' with the categories 'linear' and 'non-linear'. Participants were asked to perform sorts repeatedly until they were unable, or unwilling, to carry out additional sorts.

The study design relied on the following key assumptions.

1. The way in which a participant organizes concepts in a card sort reflects the participant's mental representation of those concepts (following Adelson).
2. By putting a card into a meaningful category (i.e. a named group rather than 'don't know' or 'not applicable') participants demonstrate that the concept on the card has some meaning for them.
3. By putting a card into a category, participants indicate what the category and the related criterion mean to them.

Hence, by examining the ways in which students sorted the cards, we hoped to gain insight into the conceptual structure of their knowledge about programming constructs and program construction.

2.1. Participants

The combined corpus included 276 participants: computer science students and faculty at 22 colleges and universities in Australia, Barbados, Ireland, New Zealand, the UK and the USA. Thirty-three were educators and 243 were students. The student participants were 'first competency programmers', i.e. they were at the point where they could solve at least one problem in the test set devised by McCracken *et al.* (2001) that involves writing a simulator for an algebraic expression calculator. 185 were male and 58 were female. Their ages ranged from 16 to 59. Their performance in computer programming courses varied widely.

The 33 faculty participants were drawn from the same institutions as the students, at least one from each institution. Eighty-two per cent of the educators had taught introductory programming, 36% had a PhD in computer science, 42% had published research in computer science and 82% had professional experience as programmers. All of them fell into at least one of these four categories. Eight were female, and 25 were male. Their ages ranged from 22 to 62. Although each educator was from the same institution as some of the student participants, he or she had not necessarily taught any of the students.

2.2. Data collection

Data collection followed a standard protocol.

1. *Background data:* Background information, including age, gender and programming language familiarity, were collected for each participant. For student participants, grades in programming courses were also recorded.
2. *Task data:* Criterion names and category names were recorded verbatim. Each card was arbitrarily assigned a number from 1 to 26, and for each category the numbers corresponding to the cards placed in that category were recorded.

3. Analysis

Analyzing card sort data is part science, part magic (Maurer & Warfel, 2003)

A portion of the data for one participant is given in Table 1. The leftmost column contains the criterion for each sort (eight criteria for this participant), with the first criterion being 'tangible and abstract'. The next column lists the categories in each sort. In the first sort there are two categories, 'tangible' and 'abstract'; to the right of that column there are columns representing the cards. (In the complete chart, there are 26 such columns, one column for each card. In the example below, however, not all columns are shown.) A symbol × in a column indicates that the card in that column was placed into the category listed on that row. For example, here the terms 'function' and 'procedure' were grouped in the same category (co-occurred) in all eight sorts, and terms 'state' and 'event' were grouped in seven of the eight sorts. The data were also combined into a single spreadsheet, to enable easier comparison across participants.

We used a variety of analysis mechanisms and tools to conduct three types of analysis: exploratory (to help us form more focused questions about the data), characterization (to characterize individuals within the corpus) and contextual (to identify and characterize subpopulations).

3.1. Exploratory mechanisms

We used three qualitative techniques:

1. *verbatim analysis* – seeking agreement on *actual* names of criteria and categories (this was automated);
2. *gist analysis on names* – seeking agreement on *the meaning* of criteria and categories, despite different verbatim naming. (For example, we might consider a sort criterion such as 'object-oriented concepts' to have the same gist as a sort criterion called 'related to object-oriented'. Similarly, 'loop', 'iterative', 'repetition' and 'looping flow' might all be considered to have the same gist.) This analysis was done by reading through the categories and criteria, sometimes by a single person,

Table 1: *Sample data*

Criteria name	*Category name*	*Function*	*Procedure*	*State*	*Event*
Tangible and abstract	Tangible	×	×		
	Abstract			×	×
Principles	Principles				
	Not principles	×	×	×	×
Data	Places to put data				
	Types of grouped data				
	Types of primitive data				
	Everything else	×	×	×	×
Programming structures	Definitely programming structures	×	×		
	Might be programming structures			×	×
	Not programming structures				
Approaches	My object-oriented (OO) world				×
	My structured world	×	×		
	Overlap			×	
OO programming	Pure OO programming				
	Not OO programming	×	×	×	×
Control structures	Control structures				
	Everything else	×	×	×	×
Modularization	Modularization	×	×		
	Everything else			×	×

sometimes by asking each researcher to scan their own data.

3. We also identified the same or similar *grouping of cards* (regardless of what the participant had named them) for groups with exactly the same group of cards and for groups with a one-card difference: one more card, one less card or one different card. These similarities were summarized in pairwise frequency tables which could be generated within a subpopulation.

3.2. Characterization mechanisms

We used three tools to generate representations that characterized an individual within the corpus.

1. We wrote an Excel application to generate *co-occurrence matrices*: identifying the frequency with which pairs of cards appeared together in the same category for individuals.
2. We used the EZSort tool[1] to perform a cluster analysis from the stimulus set for each individual's sorts. We also used the tool to generate a dendrogram (a visualization of the cluster analysis) for each participant.
3. Because we were unable to determine either the similarity metric or the clustering algorithm embedded in this tool we wrote our own software to perform clustering analysis. A hierarchical cluster analysis was computed on a distance matrix for each participant. We generated four distance matrices: using Manhattan distance and Euclidean distance, and using Simple and Jaccard's similarity measures subtracted from one to yield a distance measure. From each of these matrices, we generated dendrograms using the simple (nearest neighbour), complete (maximizing distance between clusters) and Ward's (minimizing intra-cluster distance) methods of clustering (Aldenderfer & Blashfield, 1985).

[1]*EZSort was a freely available tool from IBM. It was archived 25 January 2005.*

3.3. Contextual mechanisms

We performed simple identification of subpopulations by external context, using *background characteristics* such as age, gender, academic performance, and familiarity with specific programming languages. We also identified subpopulations from their internal context by *within-corpus characteristics*, including the average number of criteria per subject, average number of categories per criterion and the top ten categories formed by the participants (defined by the cards included in each category) both for the whole corpus and for various externally identified subpopulations.

Using these multiple mechanisms provided a wealth of data. However, not all products of these analyses were equally amenable to interpretation.

4. Findings

We investigated the complexity of our subjects' categorizations by gender (men versus women) and expertise (educators versus students). Examining the data from a quantitative point of view, we computed the average number of sorts per participant and the average number of categories per sort for these subpopulations. For men and women, we also computed the number of binary sorts (sorts in which there are precisely two categories).

Our qualitative analysis identified three additional groups of criteria, which we also explored to see if there was a difference by gender:

- criteria that order the concepts along a scale from one extreme to another, e.g. objects versus functions, abstract versus concrete, design versus implementation etc.;
- creative analogies, i.e. criteria that make an analogy to some field outside computer science;
- emotional or personal response, e.g. 'words I hate', 'things that cause me grief', 'things I'm comfortable with', 'comfortableness', 'how comfortable I am on the topic', 'overall likeness of what I do' and 'usefulness to me'.

We were surprised to find that these analyses indicated little difference between men and women, or between students and educators. A breakdown is given in Tables 2 and 3.

Men and women produced almost the same number of binary sorts (40% of men (74/185) and 41.4% of women (24/58)) and almost the same number of scalar criteria (16.2% of men (30/185) and 17.2% of women (10/58)). An equal number of men and women (two each) suggested creative analogies. 2.7% of men (5/185) and 1.7% of women (1/58) used criteria that suggested an emotional response to the concepts.

Although educators on average produced more sorts than students (5.2 versus 4.5), consistent with the suggestion by Rugg and McGeorge (1997) that experts tend to produce more criteria than novices, a two-tailed independent groups t test revealed that this result was not statistically significant ($t(39) = -1.57$, $p > 0.12$). Among the averages compared in the above tables, two-tailed independent groups t tests indicated that only the number of categories per sort between men and women differs significantly ($t(409) = 2.90$, $p < 0.01$).

Table 2: *Breakdown of student sorts by gender*

	Men students	*Women students*	*Total students*
Number of students	185	58	243
Number of sorts	831	258	1089
Number of categories	3284	1131	4415
Number of students who used binary sorts	74	24	98
Number of students who used scalar criteria	30	10	40
Number of oppositional criteria	43	14	57
Average sorts per participant	4.5	4.4	4.5
Average categories per sort	4.0	4.4	4.1
Percentage who used binary sorts	40	41.4	40.3
Percentage who used scalar criteria	16.2	17.2	16.5
Percentage whose criteria indicate an emotional response	2.7	1.7	2.5

Partial sorts, sometimes with unnamed categories, are included in the totals here.

Table 3: *Number of sorts, and number of categories per sort, for students and educators*

	Students	*Educators*
Number of subjects	243	33
Total number of sorts	1089	171
Total number of categories	4415	638
Average number of sorts per subject	4.5	5.2
Average number of categories per sort	4.0	3.7

Partial sorts, sometimes with unnamed categories, are included in the totals here.

4.1. Category groups

Exploratory analysis allowed us to draw on additional sections of the data. As part of the background information, participants were asked to self-report a level of familiarity with seven programming languages: Java, C++, C, Ada, Scheme, Pascal and Visual Basic. Participants were also allowed to report familiarity with other languages.

We computed the frequencies with which each language was mentioned. To determine whether knowledge of particular languages has an effect on category formation, we counted how often each category group was formed, in other words, the number of sorts where that combination of stimuli occurred as the *entire* contents of a category group (see Table 4). Then for each of the six languages that were most popular with our subjects, we counted the number of times each category group appeared in the sorts performed by students who reported some knowledge of that language.

We discovered that this global pattern is maintained at the local level. When these ten most frequently occurring categories are extracted for each language, they maintain a high correlation with the global pattern in terms of both frequency and relative position.

Table 4: *Most frequently occurring categories*

Category group	*Number of times category appears*
List, Tree, Array	104
Thread	52
Recursion, Loop, Iteration	48
Function, Method, Procedure	38
If-then-else, Recursion, Loop, Iteration	33
Decomposition, Abstraction, Encapsulation	28
List, Array	28
Thread, Event	28
List, Tree	27
Object, List, Tree, Array	24

The number of sorts where this combination of cards occurred as the *entire* contents of a category.

5. Discussion

The lack of differentiation in the data set, and the similarity of participants one to another, was unexpected. There are several possible explanations for this.

5.1. *Card sort data are too crude to identify the distinctions we seek*

As identified above, card sorting was an appropriate choice of technique for this scale of study. The type of card sorting that we chose was also appropriate given that we wished to remain participant, rather than researcher, focused. However, some choices we made might have flattened the responses of the participants. For example, our use of 'impoverished' stimuli might have been too decontextualized for students to categorize reliably. It may be that the task as presented required a greater dependence on participants' meta-knowledge than we had anticipated and may lead us to question the validity of the intervention.

5.2. *Our analysis methods were not subtle enough to determine distinctions within the data*

The qualitative approaches we took, our rudimentary gist analysis and examination of particular categories revealed the most distinctions between individuals (see Petre *et al.*, 2003, for details). We did not pursue the more intensive qualitative approaches and undertake gist analysis on the full data set, as the cost (in researcher time) was too high and benefit (an unwieldy set of results) of too ambiguous utility. Neither did we undertake superordinate analysis on the gisted categories; the benefit of this would probably have been high and afforded useful insights, but because of the scale of our study (and therefore the number of participants and researchers terms we would have had to reconcile) the process was too cognitively intensive. Co-occurrence matrices and dendrograms also proved unwieldy and difficult to interpret: they represented no significant analytic advantage over 'just looking' at the raw data (such as in Table 1).These issues are discussed more fully in Fincher and Tenenberg (2005) and the problems this study encountered generated new tools especially suitable for this type of analysis (Deibel *et al.*, 2005; Fossum & Haller, 2005).

5.3. *Students – regardless of gender or initial programming language – conceptualize some aspects of programming in the same way*

One of the unusual features of this study is that it involves subjects from so many different institutions, who take different approaches to teaching programming and in particular use different languages in the introductory sequence. Additional differences are introduced due to the fact that introductory students may have studied additional languages, either in secondary school or on their own. Nevertheless, there is a striking similarity between the top ten categories across all groups of students, regardless of programming language familiarity. This suggests that at least some aspects of participants' conceptual structures may be consistent across programming languages.

6. Summary

Through teaching style, textbooks and exercises, computer science educators articulate what they believe to be the appropriate conceptual structures of program construction and of programming constructs for beginning programmers. It is harder to identify the *actual* conceptual structures which students form. This study used a multiple, participant-defined, single-criterion card sort to elicit students' conceptual structures. Traditional methods of analysing card sort data revealed remarkably few differences over externally defined subpopulations such as gender, degree of expertise and programming language background. The size of the data set stressed traditional methods, and the challenges posed by such a large data set generated new tools especially suitable for this type of analysis.

Acknowledgements

The Bootstrapping project was supported by the National Science Foundation Grant DUE-0122560. We are grateful to Barbados Community College for data-gathering opportunities; to Gordon Rugg for advice and guidance on card sorting and laddering; to Robin Blume-Kohout for advice on matrix comparability; to Ian Utting for advice and insight; to Kate Deibel and Janet Davis for help with data collection; to Karen Furuya for help with data analysis; to Robert McCartney for help with the statistical analysis; and to Noel Welsh and Jim Bender.

References

Adelson, B. (1981) Problem solving and the development of abstract categories in programming languages, *Memory and Cognition*, **9** (4), 422–433.

Aldenderfer, M.S. and R.K. Blashfield (1985) *Cluster Analysis*, Newbury Park, CA: Sage.

Bootstrapping Research in Computer Science Education, retrieved 8 February 2005 from http://depts.washington.edu/bootstrp/

Davies, S.P., D.J. Gilmore and T.R.G. Green (1995) Are objects that important? The effects of expertise and familiarity on the classification of object-oriented code, *Human–Computer Interaction*, **10** (2–3), 227–248.

Deibel, K., R. Anderson and R. Anderson (2005) Using edit distance to analyze card sorts, *Expert Systems*, this issue.

Fincher, S. and J. Tenenberg (2005) Making sense of card sorting data, *Expert Systems*, this issue.

Fossum, T. and S. Haller (2005) Measuring card sort orthogonality, *Expert Systems*, this issue.

Martine, G. (2000) Quantification of 'look and feel' copyright infringement in Web page design, Unpublished MSc thesis, University College Northampton, UK.

MAURER, D. and T. WARFEL (2003) *Card Sorting: a Definitive Guide*, retrieved 8 February 2005 from http://www.boxesandarrows.com/archives/card_sorting_a_definitive_guide.php

MCCRACKEN, W.M., V. ALMSTRUM, D. DIAZ, M. GUZDIAL, D. HAGAN, Y.B.-D. KOLIKANT, C. LAXER, L. THOMAS, I. UTTING and T. WILUSZ (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, *ACM SIGCSE Bulletin*, **33** (4), 125–140.

PETRE, M., S. FINCHER, J. TENENBERG, R. ANDERSON, R. ANDERSON, D. BOUVIER, S. FITZGERALD, A. GUTSCHOW, S. HALLER, M. JADUD, G. LEWANDOWSKI, R. LISTER, R. MCCAULEY, J. MCTAGGART, B. MORRISON, L. MURPHY, C. PRASAD, B. RICHARDS, K. SANDERS, T. SCOTT, D. SHINNERS-KENNEDY, L. THOMAS, S. WESTBROOK and C. ZANDER (2003) 'My criterion is: is it a Boolean?': a card-sort elicitation of students' knowledge of programming constructs, Computing Laboratory Technical Report 6-03, University of Kent, Canterbury.

ROSENFELD, L. and P. MORVILLE (2002) *Information Architecture for the World Wide Web,* Cambridge and Sebastopol, CA: O'Reilly.

RUGG, G. and P. MCGEORGE (1997) The sorting techniques: a tutorial paper on card sorts, picture sorts and item sorts, *Expert Systems*, **14** (2), 80–93.

The authors

Kate Sanders

Kate Sanders is an associate professor in the Mathematics and Computer Science Department at Rhode Island College in Providence, Rhode Island. She received an AB in classics from Brown University, a JD from Harvard University and a PhD in computer science from Brown University, with a thesis on artificial intelligence and law. Besides artificial intelligence and law, her research areas include case-based reasoning and computer science education.

Sally Fincher

Sally Fincher is a lecturer in the Computing Laboratory at the University of Kent where she leads the Computing Education Research Group.

Dennis Bouvier

Dennis Bouvier is assistant professor at Parks College of Engineering, Aviation, and Technology, Saint Louis University. He holds BS and MS degrees in electrical engineering from the University of New Orleans and a PhD in electrical engineering from the University of Louisiana, Lafayette.

Gary Lewandowski

Gary Lewandowski is an associate professor of mathematics and computer science at Xavier University.

Briana Morrison

Briana Morrison earned a BSE in computer engineering from Tulane University in 1987 and an MS in computer science from Southern Polytechnic State University (SPSU) in 1995. Briana was a software developer for IBM before joining the Computer Science Faculty at SPSU where she has been teaching for 10 years. Her current research interests include computer science education, accreditation issues, and gender issues in computer science.

Laurie Murphy

Laurie Murphy received her BSc in mathematics from Delta State University in 1986 and her MSc in computer science from Vanderbilt University. She has been an assistant professor in the Department of Computer Science and Computer Engineering at Pacific Lutheran University since 1977. Her current research interests are in computer science education and focus on classroom assessment and increasing the participation of women in computing.

Marian Petre

Marian Petre is a Reader in the Computing Department at the Open University. She holds a BA in psycholinguistics (Swarthmore College) and a PhD in computer science (University College London, UK).

Brad Richards

Brad Richards received his PhD from the University of Wisconsin – Madison in 1996 and joined the Computer Science Faculty at Vassar College that same year. In addition to computer science education, his current research interests include wireless networks and parallel computing.

Josh Tenenberg

Josh Tenenberg is an associate professor in the Computing and Software Systems program in the Institute of Technology at the University of Washington, Tacoma. He holds a BM in music performance (San Francisco State University, USA) and an MS and PhD in computer science (University of Rochester, USA).

Lynda Thomas

Lynda Thomas is a Senior Teaching Fellow at the University of Wales, Aberystwyth. She holds a BSc and

MSc in mathematics from McMaster University (Canada), an MSc in computing from Southern Illinois University and a PhD from the University of Wales.

Richard Anderson

Richard Anderson received his PhD from Stanford University in 1986 and has been on the Computer Science and Engineering Faculty at the University of Washington since 1986. He has worked in many areas of computer science, including analysis of algorithms, parallel computation, scientific computation and symbolic model checking. His current research interests centre around computer science education and educational technology.

Ruth Anderson

Ruth Anderson received a BSc in computer science from the University of North Carolina in 1991 and an MSc in computer science from the University of Washington in 1994. She has been a lecturer in the Department of Computer Science at the University of Virginia since September 2000. Her current research interests are educational technology and computer science education.

Sue Fitzgerald

Sue Fitzgerald is professor of computer science at Metropolitan State University. Sue earned an MS in computer engineering at Iowa State University and a PhD in computer science and telecommunications from the University of Missouri – Kansas City.

Alicia Gutschow

Alicia Gutschow is assistant professor of information technology and computer science at Blue Ridge Community College in Virginia.

Susan Haller

Susan Haller is an associate professor of computer science at the University of Wisconsin – Parkside in Kenosha, Wisconsin. Professor Haller received a PhD from the State University of New York at Buffalo in 1995. Professor Haller's primary research areas are natural language generation and computer science education.

Raymond Lister

Raymond Lister is senior lecturer in the Department of Software Engineering, University of Technology, Sydney, Australia.

Renée McCauley

Renée McCauley received her PhD from the University of Southwestern Louisiana (now University of Louisiana at Lafayette) in 1992. She has been an associate professor of computer science at the College of Charleston since 2000. Her current research interests are in computer science education.

John McTaggart

John McTaggart is associate professor in the Department of Mathematics and Computer Science at Drake University.

Christine Prasad

Christine Prasad holds a BSc (Hons) from Monash University and is currently a lecturer in the School of Computing and IT at Unitec New Zealand.

Terry Scott

Terry A. Scott is currently associate professor of mathematical sciences at the University of Northern Colorado in Greeley, Colorado.

Dermot Shinners-Kennedy

Dermot Shinners-Kennedy is a lecturer in the Department of Computer Science and Information Systems, University of Limerick, Ireland.

Suzanne Westbrook

Suzanne Westbrook received her BS in computer and information sciences from the University of South Alabama in 1988 and her PhD from the University of Southwestern Louisiana (now the University of Louisiana at Lafayette) in 1998. She has been a lecturer in the Computer Science Department at the University of Arizona since 1999 working primarily with undergraduate students. Her current research interests include computer science education and diversity issues.

Carol Zander

Carol Zander received a PhD in computer science from Colorado State University in 1995 and an earlier MSc in mathematics from the University of Colorado. She has taught at the University of Maine, Seattle University, and since 1996 at the University of Washington, Bothell. She has worked in distributed artificial intelligence and has interests in programming languages and object-oriented programming, but her current research centres on computer science education.