Repertory Grid: Investigating Personal Constructs of Novice Programmers

Neena Thota
University of Saint Joseph
Nape, Rua de Londres 16, Macau
+853 66814445
neenathota@usj.edu.mo

ABSTRACT
In this paper, the repertory grid is presented as a technique to explore novice programmers' experiences within the context of an action research project. The theoretical and methodological aspects of the technique are discussed. The findings from the technique that combined quantitative and qualitative data analysis methods are provided. These findings relate to the learning process, learning content, and learning support as experienced by the students in an introductory object-oriented programming course. The repertory grid technique is then appraised for its relevance and usefulness to the project, and for its contribution to the diversity of computer science research methods. Insights gained from the use of the technique are shared with the community of computer science educators.

Keywords
Repertory grid, novice programming, mixed methods.

1. INTRODUCTION
The repertory grid technique is a form of structured interview to find out a participant's preferences on a given topic and the way these preferences are ordered on a rating scale. The repertory grid was designed by Kelly [27] who believed that to understand a person one has to understand how that person interprets personal choices. Kelly drew his insights from his Personal Construct Theory (PCT) which supports interpretivist research and is guided by pragmatic logic that assigns the burden of discovery to the researcher to find out what the participant in the phenomenon under investigation has learned.

This paper describes an action research project in which the repertory grid technique was used to gain a deeper understanding of the perspectives of novice programmers. The experience of using the technique and the findings from the study are presented, along with recommendations for other computer science educators who teach introductory programming. It is proposed that there should be a wider use of the repertory grid as a research method in Computing Education Research (CER).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Koli Calling '11, November 17--20, 2011, Koli, Finland.
Copyright 2011 ACM 978-1-4503-1052-9/11/11…$10.00.

The outline of the paper is as follows: In Section 2, the theoretical background of the technique is explained. In Section 3, the appropriateness of using the technique to explore the experiences of novice programmers is examined. Section 4 establishes the context of the action research project in which the repertory grid technique was used. Section 5 deals with data collection procedures such as grid setup and elicitation of the participants' preferences. This section also describes the content analysis techniques that integrated quantitative and qualitative measures. Empirical findings about the learning experiences of the students are categorized and summarized in Section 6. The paper concludes (Section 7) with a discussion of the implications of the findings, and with a critique of the repertory grid as a technique for understanding the personal constructs of novice programmers.

2. THEORETICAL BACKGROUND
The repertory grid is a set of rating scales with the ratings arranged in rows and columns. The four components [26] of a repertory grid are: (a) the *topic* that deals with the general field of personal knowledge within which the grid is situated; (b) the *elements* that are the examples of the topic; (c) the *constructs* (preferences elicited from the participant) that are the units of description about the elements; and (d) the *ratings* that each participant gives to each element on each construct. During the interview, three of the elements are taken together and the respondent is asked which two of the elements are the same in some way, and different from the third element. The participant's expressions are noted as constructs on a rating scale. The participant then rates the elements on this construct. The process is repeated until no more constructs can be elicited. The repertory grid yields data in the form of numeric grid ratings and descriptive constructs. The quantitative data can be statistically analyzed, while the personal constructs are amenable to qualitative content analysis.

The repertory grid technique is anchored in PCT, with its fundamental principle of *constructive alternativism* that perceives each person as a scientist capable of formulating constructs about the world. Kelly [27] postulated eleven corollaries to amplify his theory: People develop varying representations of their experiences (Experience corollary) by representing them as constructs (Construction corollary), with each individual being unique in his or her way of seeing the world (Individuality corollary). However, people are similar to the extent that they see meaning in events similarly (Commonality corollary), and even though a person can choose alternatives (Choice corollary), it is possible to be aware of and understand the constructs of others (Sociality Corollary). The constructs are bipolar in nature (Dichotomy corollary), and refer to a finite range of events (Range corollary). The constructs are arranged in a hierarchy (Organization corollary), with some constructs more applicable to

many events (Modulation corollary), while others have less internal consistency (Fragmentation corollary). These corollaries are worked into the design of the repertory grid that allows a person to express bipolar constructs about related events and to rate the constructs according to individual preferences.

The repertory grid interview technique, with its grounding in PCT, allows a researcher to understand a respondent's individual perspectives and to find commonalities with the personal constructs of co-participants in the same event [20]. It is a suitable tool for an action research project that recognizes learners and teachers as participants [48]. The output of interviews that embody students' experiences in their own voice has ecological validity [16], while the systematic collection of complementary quantitative and qualitative data provides internal validity [18]. The technique reduces researcher interference or bias as compared to more traditional interviewing techniques [1].

Repertory grids have been used extensively in education, market research, politics, and organizational and business applications. In the field of education, the analysis of grid data can act as an illuminative approach for evaluating teaching and to investigate learning outcomes [28]. The interview technique can help to raise the focal awareness of the participants and enable them to voice their tacit thoughts [40]. Investigations into the influence of open-ended technology projects on students' thinking [42] and changes in students' understandings of design in information technology [41] have revealed that implications for teaching can be drawn from the data generated by repertory grids. New findings on implicit learning and knowledge have led to the application of the technique in areas such as artificial intelligence research on knowledge acquisition for expert systems [21], as data gathering methodology in information systems [44], as a means of improving performance in IT teams [6], and for evaluating the usability of mobile technologies [14]. However, the study reported in this paper is the first instance of the use of the repertory grid technique in CER aimed at exploring the experiences of novice programmers.

3. Experiences of Novice Programmers

Learning programming is a perennial problem [8] that continues to be the focus for studies in CER. It is a well-acknowledged fact that introductory programming courses have high failure and drop-out rates [3, 29]. Several studies highlight that students lack the knowledge and skills for problem solving [30, 32]. Novice programmers, in particular, struggle with basic program design due to lack of specific knowledge and strategies [10, 36]. Not surprisingly, computing education researchers are interested in research into students' experiences of learning programming.

A review of the literature shows that phenomenography has been used as a research method in studies that explored students' experiences of programming. Booth [5] found that the experience of programming was fundamental to students generating, expanding, and refining programming conceptions. Bruce et al. [7] studied the ways that students go about learning to program and identified five different ways in which learning approaches, activities, and motivations influenced the ways of seeing programming, programs, and the programming language. However, none of these phenomenographic studies focused on the specific use of resources or programming activities to obtain feedback about the course design. On the other hand, Eckerdal [13] used content analysis of interviews to investigate how different resources were used by students to learn programming and how the students had experienced that the different resources supported them in their learning. Eckerdal concluded that depending on the student's approach, the resources provided superficial or meaningful support for learning. Eckerdal recommended that resources and learning activities that involve use and understanding in a complex way facilitate deeper learning approaches to programming.

Most evaluations of innovative course designs in computing [23, 39, 45] rely on surveys to obtain feedback about the course components. A study that combined ethnographic and cognitive methods to understand students' programming experiences within a course context [35] was confined to the use of in-class lecture and online newsgroup. There is no existing study that has used the repertory grid technique to elicit the personal constructs of novice programmers as a means of understanding the course experience or as a means of evaluating the course design.

It is argued that knowledge of how students experience programming is vital for drawing conclusions about the kind of learning environment and learning experiences that assist students to achieve desirable learning outcomes. An explicit consideration of students' perspectives on the usefulness of learning resources, activities, and assessments can inform teaching practices in introductory programming courses. In the action research study reported in this paper, the repertory grid technique serves to illuminate the tacit understandings of novice programmers and acts as an evaluation tool for a course design that was underpinned by theoretical considerations.

4. RESEARCH CONTEXT

The action research project, discussed in this paper, implemented and evaluated the outcomes of a redesigned introductory Object-Oriented Programming (OOP) course for undergraduate students. The nature of the problems in the introductory programming course that motivated this action research can be found in [46]. A theoretical framework (also described in detail in [46]) was derived from the literature for:

- Constructive alignment of learning outcomes with assessment tasks (programming related online quizzes, exams, assignments, projects, and reflective journals);
- Design of learning and teaching activities (pair and team work, lectures, labs, demonstrations, and peer tutoring) to encourage students to use deep learning approaches to achieve the learning outcomes;
- Creation of a learning environment that enabled students to experience a variety of educationally critical ways [31] of learning to program through active and collaborative learning;
- Creation of a learning context with multiple media (IDE, visualization software, UML editor, graphical library classes, web-based learning objects, videos, games, and multimedia tutorials) to enhance the learning experiences.

The course was implemented in two semesters (2008 to 2009) corresponding to two cycles of the action research project. The course design was revised in the light of the initial feedback obtained from the students, and was further fine-tuned after the second evaluation. During the course implementation, questionnaires, students' journals, work assessments, and observations served as formative feedback [15] of the course design. A mixed methods two-phase sequential explanatory design [9] was used for the summative evaluation of the outcomes of the action research project. In each cycle, the learning

approaches of the students were ascertained through a questionnaire [4] to determine to what extent the learning context influenced the learning approaches of the students. To investigate how the learning environment influenced the learning experiences of the novice programmers, students with maximum variation of scores for deep and surface learning approaches were identified for interviews with the repertory grid. The specific research question that was addressed was: *How does the learning environment influence the learning experiences of the students?*

In cycle 1, the course enrollment included one group of 12 students from Information Systems, Business Technology Management, Design, and Pre University. All 26 students gave their consent for the research. The students were non-native English speakers of mixed nationalities (Chinese, Portuguese, Brazilian, and Nigerian). In cycle 2, there were 3 groups with a total of 82 students from Business Administration, Design, and Information Systems. As in the previous cycle, the students were of mixed nationalities (Chinese, Portuguese, Russian, Australian, Brazilian, and Nigerian). Seventy-two of these students gave their consent for the research. A sample size of 15 to 25 participants for an interview with a repertory grid is considered sufficient to generate enough constructs to approximate the universe defined by the intervention [44]. Fourteen students in cycle 1 of the action research project and 15 students in cycle 2 were interviewed for this study.

5. DATA COLLECTION AND ANALYSIS

This section describes the procedures related to the grid setup and construct elicitation that were undertaken in this study. The data analysis combined quantitative and qualitative content analysis measures. The construct categorization and intercoder reliability procedures are given in detail to establish the rigor of the research method.

5.1 Grid Setup

Figure 1 depicts a sample, of a completed grid from this study, with the grid components that have been identified. In the figure, the constructs that were elicited from a particular student (code ARC1S2, or student number 2, from action research cycle 1), and the ratings given by the student for each element on each construct can be seen. The topic of the grid was *Exploring the nature of learning situations* in the OOP course. The elements that were supplied were the learning situations in the programming course (reflective work, software used, practice quizzes, written exam, team project, pair programming, peer help, lecturer feedback, web resources, and lecturer material).

There is a wide disparity of views about the benefit of the interviewer supplying/eliciting the elements and constructs in a repertory grid [20]. In this study, the elements in the grid were supplied to the participants. The supplied elements provided a basis for gathering participants' conceptions of the programming context, were grounded in theoretical considerations, and enabled comparisons of the responses of respondents [44] from the 4 student groups. The use of 8 or 10 homogenous, representative, and unambiguous elements is recommended in a grid [12]. Following these recommendations, 10 learning and teaching activities familiar to the students in the OOP course were supplied as elements, and the constructs were elicited during the interview with each student. At the end of the elicitation period, one additional construct (*Overall learnt a lot - Overall did not learn much*) was supplied by the interviewer to each student to rate. The ratings on the supplied construct [25] enabled a summary of the students' views about the course and facilitated content analysis.

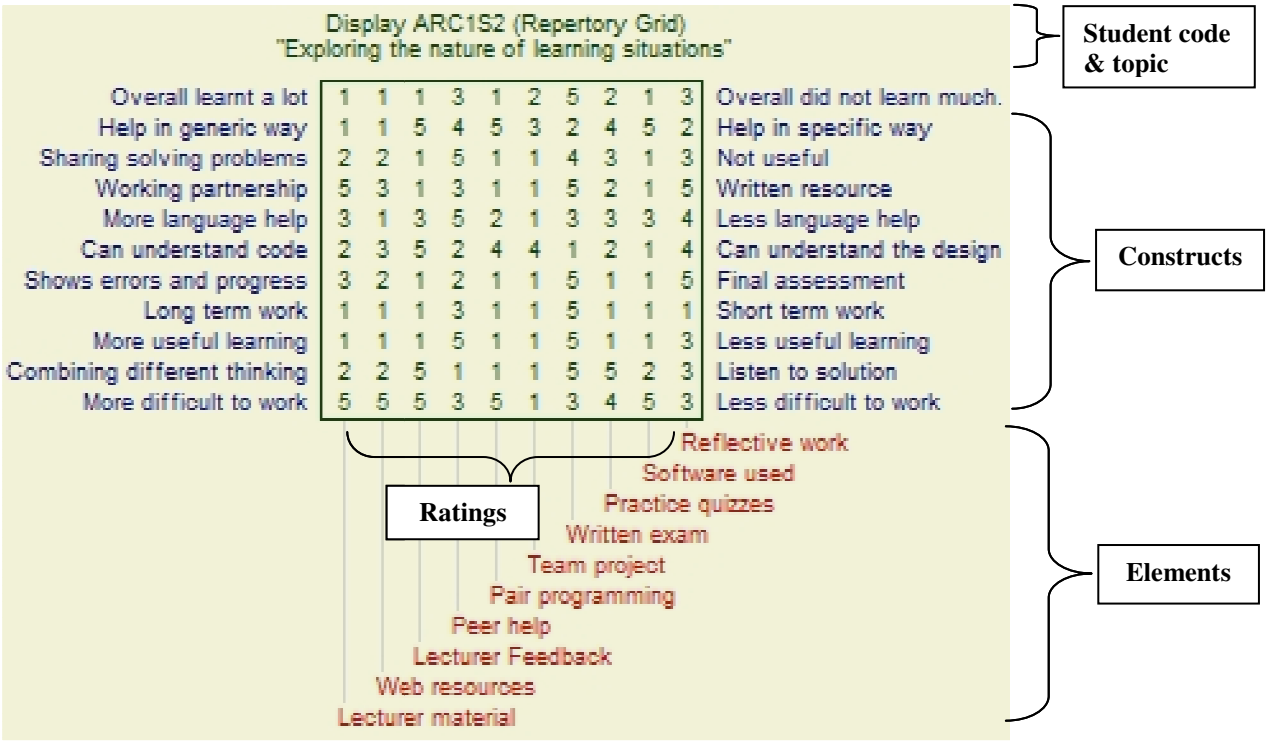

Figure 1. Sample of a completed repertory grid from the study.

5.2 Construct Elicitation

Computer support for construct elicitation in repertory grids allows the researcher to give feedback and rapid analysis, while simultaneously attempting to avoid distortion of the elicited constructs through personal intervention [43]. Given the popularity of the repertory grid technique, several computer-based programs are available for grid elicitation and analysis. Of these, two Windows-based programs, Idiogrid 2.4[1] and Rep IV[2] were considered. Idiogrid provides many advanced statistical analysis features, which seemed unnecessary given the interpretative nature of this study. During the pilot trials, the Rep IV elicitation script proved intuitive to use and the analysis features were adequate for predominantly qualitative content analysis.

The built-in elicitation script in Rep IV offers 3 elements at a time for consideration by the respondent. A rating scale ranging from two to nine is provided by the software. A 5-point scale was used for this study, as a 7-point scale in a repertory grid approaches the limits of a participant's discriminatory abilities, and anything above 5 points presents difficulty in visual examination of the grid [43]. Two pilot trials (conducted with other cohorts) helped to finalize the following procedures for the interviews in this study:

1. The method for the construct elicitation was read from a prepared sheet of paper to ensure consistency in the instructions given to the participants. This would reduce any bias on the part of the researcher.
2. Interviews were recorded and the elements that were offered by the triadic grid elicitation script were written in a notebook so that they could be referred to later to contextualize the construct. This removed the possibility of imposing the researcher's interpretation of the construct labels.
3. At the end of the interview, the participants were asked if they had any suggestions for improving the course. Since the grid data is only indicative, the suggestions would provide explanation, validation, and triangulation with other data such as the researcher's personal observations.

Each interview lasted about an hour during which 9 to 10 constructs per student were entered in the grid. The following qualifying phrase [26] was used to focus the students' attention on the topic during the interview: *Which two of these (three elements) are the same in helping you to learn programming, and different from the third?* The first part of the question yielded the *emergent* pole of the construct (defined by the "1" end of the scale), and the second part yielded the *implicit* pole of the construct (defined by the "5" end of the scale). The process of *laddering down* [12] was then employed to clarify the meaning of the constructs. These techniques proved useful, especially with those students who were non-native English speakers.

Rep IV automatically generates graphic plots of focused cluster analysis and principal components analysis of the ratings. These graphs were discussed with the student, and if any student wished to adjust ratings or the wording of constructs, then the changes were immediately carried out. The opportunity to talk about or interpret the patterns added substantially to the researcher's understanding of student thinking, thus making the student an essential part of the evaluation process [47].

[1] http://www.idiogrid.com/

[2] http://repgrid.com/

5.3 Data Analysis

In the repertory grid technique, each elicited construct constitutes a basic unit of analysis that expresses a single unit of meaning. Content analysis of grid constructs can take the form of simple frequency counts of the number of times the construct occurs, or where the elements are identical, as in this study, content analysis can be used to aggregate the meanings of the constructs and categorize them into themes [43].

Honey's [25] content analysis technique, which was used in this study, provides a way to combine statistical analysis of the numeric ratings of the elements with analytic induction [34] to aggregate constructs into categories. Such inductive categories emerge from the data and are not applied *a priori*. Honey's [25] approach to content analysis was based on the assumption that the ratings on the elicited constructs that match the ratings on the supplied construct portray the individual's personal meaning of the topic. The correlation between the ratings of a construct and the rating of the supplied construct (*Overall learnt a lot - Overall did not learn much*) was measured by computing the sum of difference and the percentage similarity score using the procedures outlined in [26]. The similarity scores for the constructs were labeled with a high, intermediate, or low (H-I-L) index [25].

The process of inductive content analysis to create categories of the constructs was carried out in several stages. The criterion of definition for a category was to identify the learning experience of the novice programmers. Following this criterion, 112 constructs (elicited from the repertory grid interviews in the first action research cycle) were initially analyzed by manual examination of the constructs. Coding labels that had sufficient resemblance to the original piece of datum were selected, and in an iterative process of reflexive scrutiny the constructs were refined to seven categories and assigned to three meta-categories:

- Learning process (Learning as experiencing; Learning through reflection);
- Learning content (Learning by coding; Learning through information; Learning from assessment);
- Learning support (Learning through scaffolding; Learning from collaboration).

Categories that are developed from a data driven content analysis of constructs are closer to the raw data, but have low potential for replicability by others as they involve an individual construing the constructs of others [22]. There is also the problem of overgeneralization of the derived categories [17]. Therefore, coding consistency checks were applied in this study. Two colleagues, neither of whom had taught programming, were given the categories with the descriptions. Along with the author, the three acted as independent coders. To enhance the consistency check, each coder allocated the full set of 112 constructs to the categories, rather than using a representative sample.

Intercoder reliability is essential for validating a coding scheme and reflects the amount of agreement or correspondence among two or more human coders in assigning exactly the same rating to each object [33]. For the nominal-level data in this study, Krippendorff's Alpha was used to calculate intercoder reliability coefficients. The index accounts for the level of measurement and agreement expected by chance and computes reliability estimate for judgments made at any level of measurement, any number of coders, with different sample sizes, and with or without missing

data [24]. Krippendorff's Alpha can be calculated by a macro[3] that is freely available for SPSS. As a measure of crosschecking, the reliability figures were recalculated with ReCal3[4], an online intercoder reliability calculator that additionally provides results for average pairwise percent agreement, Fleiss' Kappa, and Cohen's Kappa.

The reliability coefficients, obtained for the first attempt to allocate the constructs to the categories by all three coders (two colleagues and the author), were 0.53 for Krippendorff's Alpha (59.82% for average pairwise percent agreement). A discussion among the coders revealed that the bipolarity of constructs led to many disagreements. The nature of the implicit and emergent poles of constructs was explained to the coders, and a decision was taken to be guided primarily by the emergent pole (which shows how two elements are alike). The three coders then independently carried out a second round of coding, and the figures for acceptable levels of agreement are reported as 0.81 for Krippendorff's Alpha (84.52% for average pairwise percent agreement). Krippendorff's alpha is known to be conservative and an index of .70 or higher is considered as acceptable. A further verbal process of harmonization between pairs of coders was conducted until full agreement was reached on the construct categories.

After the constructs were categorized, the mean percentage similarity score was computed for each category and was used to estimate the relative importance of that category. The aggregated set of constructs represents the categorized views of all the individuals and additionally conveys a summary of individual meanings [26]. Summary tables were created with the category headings, and frequencies of constructs from each category. Personally salient constructs (referring to the H-I-L indices) on which there was consensus in the group were identified. Honey [25] compared only the top-and-tail data in his construct analysis and discarded the intermediate data. However, this study preserved the information about *each* individual's views of the elements in the programming course. The categorization and summary procedures were repeated for the 131 constructs elicited in the second cycle of the action research project.

Kelly [27] defined validity as the capacity of a grid to enable a person to elaborate constructs, and he equated validity with usefulness. In this study, explicit measures were taken to combat threats to validity and trustworthiness through the pilot trials, and the coding consistency and intercoder reliability checks described in this section. Finally, to enhance the credibility of the findings, stakeholder checks were applied through discussions of the findings with interested students who had participated in the interviews and with the colleagues who acted as coders.

6. FINDINGS

The personal constructs of the students revealed which elements of the learning environment were considered helpful for learning programming. Findings from the repertory grid interviews in the first cycle, including students' constructs, have been summarized and reported earlier [46] and are repeated here to show the comparison with the findings from the second cycle. Table 1 shows the numeric findings from the repertory grids in the two cycles of the action research project. In the table, the category names are followed by the number and percentage of constructs in that category, the mean percentage score, and the total number of constructs labeled by the H-I-L index. The categories are ordered by the percentage similarity scores.

The data shows that in cycle 1, constructs relating to informational resources have the highest mean similarity score (53.57%) for all constructs, indicating that these resources were considered the most important for learning programming. The constructs on *learning as experiencing* (52.37%) were ranked second to learning from informational sources. *Learning through reflection* (45.77%) and *learning through scaffolding* (43.75%) were considered as more helpful for learning programming than learning from planning and coding activities (40.71%) and programming assessments (40.38%). Constructs relating to collaborative work have the lowest mean similarity score (33.85%) suggesting that the students had some problems with group work.

In cycle 2, the constructs relating to *learning through reflection* have the highest mean similarity score (56.05%) for all constructs, whereas in the previous cycle, *Learning through information* (53.57%) was considered overall effective. Comparison of the similarity scores shows the increased importance attached in cycle 2 to scaffolding measures (54.05% vs. 43.75% in cycle 1) and programming practice (49.09% vs. 40.71% in cycle 1). The similarity score for constructs relating to collaborative work was again the lowest (33.64%), indicating recurrent problems with group work. In cycle 2, *learning from assessment* (46.11%) was considered marginally more effective than *learning as experiencing* (44.71%).

Major changes were made to the course design after the first action research cycle, based on the formative and summative evaluations that were conducted. However, in this paper, only specific changes made to the course design that are related to findings from the repertory grid interviews are noted. The categories that emerged from the constructs are discussed next, grouped by meta-categories (learning process, learning content, and learning support). Students' constructs are indicated in italics.

6.1 Learning Process

6.1.1 Learning through Reflection

Constructs in this category related to students' reflections on their personal understanding or knowledge development. Course elements that encouraged self-reflection included writing journal entries about the learning plans, goals, and motivations.

In cycle 1, activities that were easy to comprehend, generated ideas or reflected the skill or ability of the student to program were considered moderately effective for learning and understanding. Personal development and learning through *effort*, *new ideas*, and *discovery about knowledge* were valued as also *learning about weakness and improving*. Self-study skills were perceived to be only for individual benefit and not conducive for developing *leadership and organizational skills*. There was some awareness of learning as problem solving, *combining different thinking*, and being able to *change your learning*. The constructs reflected the value of writing reflections. As such, journal writing was retained as an activity for the second cycle, though the frequency was reduced from weekly submission to three times during the semester.

[3] http://www.afhayes.com/public/kalpha.sps

[4] http://dfreelon.org/utils/recalfront/recal3/

Table 1. Categories of constructs from two cycles of the action research project

Cycle 1 constructs				Cycle 2 constructs			
Category names	**Number %**	**Mean %**	**Total H-I-L**	**Category names**	**Number %**	**Mean %**	**Total H-I-L**
Learning through information	7 6%	53.57	2H, 3I, 2L	Learning through reflection	19 15%	56.05	7H, 8I, 4L
Learning as experiencing	19 17%	52.37	8H, 4I, 7L	Learning through scaffolding	42 32%	54.05	13H, 12I, 17L
Learning through reflection	26 23%	45.77	5H, 12I, 9L	Learning by coding	11 8%	49.09	8I, 3L
Learning through scaffolding	20 18%	43.75	6H, 6I, 8L	Learning through information	4 3%	46.25	1H, 1I, 2L
Learning by coding	14 13%	40.71	4H, 2I, 8L	Learning from assessment	9 7%	46.11	2H,4I, 3L
Learning from assessment	13 12%	40.38	2H, 5I, 6L	Learning as experiencing	35 27%	44.71	4H, 10I, 21L
Learning from collaboration	13 12%	33.85	1H, 5I, 7L	Learning from collaboration	11 8%	33.64	2H, 9L

In cycle 2, the course elements were perceived as contributing to understanding about oneself and understanding the *practical* and *conceptual* aspects of programming. The constructs revealed perceptions of programming as a way of thinking to solve problems. The notion of programming as thinking was variously construed as *need to think*, *logical thinking, thinking about achievement*, and to a lesser extent as about *individual thinking* and *thinking about experience*. Ideas generated during the course were classified as *creative* and *business*. The course was seen as offering an insight into one's learning.

6.1.2 Learning as Experiencing

Constructs in this category described the environment in which learning took place or were related to value judgments of the course design. Some constructs were affective outcomes expressing emotions or feelings about the learning activities.

In cycle 1, constructs that related to *long term work, more variety*, and *active course work* revealed the course aspects that the students found relevant for their learning. There was no consensus about the usefulness of compulsory or optional activities for study and practice. Constructs relating to *personal opinion, freedom to research or ask, freedom of choice*, and *more personality based* activities pointed out study preferences and areas for improvement of the course design. These needs were explicitly addressed in the course redesign for the second cycle.

In cycle 2, the course elements were experienced as leading to *individual development* and helpful for learning. The course generated *new ideas*, but was also *more challenging* than other courses. Constructs with moderate ratings referred to the course elements as *complements knowledge*, being able to *change knowledge*, and encouraging *creativity for improvement*. The course was experienced as *interesting, easy to follow, with clear direction for work,* and *encouraging independent work*. Less salient constructs pointed to the course as being *boring* and requiring *hard work* for compulsory assignments.

6.2 Learning Content

6.2.1 Learning from Assessment

This category summarizes aspects that relate to assessments and grading. The practice quizzes, pair assignments, written exam, and the team project included formative and summative assessment. Evaluation of programming skills and affective values was undertaken using a holistic grading scheme.

In cycle 1, students were able to differentiate between formative assessments, that allowed students to practice and receive comments about their errors, and summative assessments that simply returned the grade. Assessment activities were generally perceived as giving the *result of knowing*, the way to *get answers*, and helping one to become aware of *errors and progress*. There was indication of the need for *help before learning*, and more *practice*, to show *what is missed in learning*. In the second cycle, more practice exercises with solutions were provided.

In cycle 2, students could again differentiate between formative assignments that enabled one to *prepare with answers*, and summative assignments that just showed the *result of preparation*. Assessments were seen not only as a way to improve the grade, but also as a chance for improving one's work and testing understanding.

6.2.2 Learning by Coding

This category describes programming related activities that included designing a program (with use cases, CRC cards, and UML diagrams), and reading and writing code.

In cycle 1, the students' constructs were divided on what constitutes effectiveness for learning programming –knowing how to write code or knowing how to design a program. Some constructs focused on the difficulty of writing code as compared to reading code. Students perceived real-time programming as being more helpful for learning. Several constructs referred to working on the *big picture* or *ideas for plan*, or *steps to plan* when programming. Students were able to differentiate the use of programs from thinking about programming, and *doing programming* from *learning about programming*. Some constructs related to the need for more solutions for programming problems and step-by-step tutorials. In the second cycle, more emphasis was given to reading code to prepare students to be able to write code.

In cycle 2, none of the constructs ranked closely with any individual's rating on the overall construct. On the other hand, constructs that moderately matched the overall construct

differentiated between using a tool for programming, learning programming, and *thinking about programming* rather than *how to do programming*. The lowest ranked constructs showed that students perceived programming more as writing code, than as planning and design of programs.

6.2.3 Learning through Information

This category relates to learning from resources such as lecture slides, notes, books, and web pages.

In cycle 1, the resources were perceived as overall most effective for learning programming. Students differentiated the learning of theoretical programming concepts from developing affective values such as *communication and organisation skills*, and *cooperative work*. The code samples and information provided for the assignments or freely available on the Internet were considered the most helpful for learning to program. In the second cycle, the layout of the course web page was further improved and specific resources were created to tackle programming errors and misconceptions.

In cycle 2, there were only four constructs in this category, with two constructs from the same student. This student found the information sources provided *accurate information*, unlike the *questionable information* given by peers. However, as compared to the reference material provided, this student preferred *redirection* or *personal intervention* when trying to solve a problem. The other two students found the resources as *useful for new ideas* and to a lesser extent to *motivate* to learn.

6.3 Learning Support

6.3.1 Learning through Scaffolding

This category relates to learning from feedback and solutions from peers, student tutor, lecturer, BlueJ IDE, learning objects, and Jeliot visualizations.

In cycle 1, feedback about programming errors and mistakes was considered useful for increasing understanding and for improving the work. There was no clear consensus about the effectiveness of help before, or after completing an individual assignment. The need for more *individual help*, *comments after work*, *more language help*, and *classmate help* was evident. Students construed the many forms of help as *share knowledge, ask for help, want to help*, and *need to help*. In the second cycle, more detailed feedback was provided for assessments, and a student tutor (who spoke Chinese and English) was appointed to help students.

In cycle 2, this category had the largest number of constructs (n = 42). Course elements that were helpful for coding errors, for review, and problem solving were perceived as highly beneficial. The provision of *help by choice* and *self-help* was appreciated. Specific help in the form of feedback for Java code, program design, and sample solutions was found moderately useful. The usefulness of help from the student tutor in Chinese was also mentioned.

6.3.2 Learning from Collaboration

This category relates to collaboration among students for pair programming and team projects.

In cycle 1, this category averaged the least score for overall effectiveness for learning (33.85%). Inexperience with group work and conflicts within teams were seen as problems. Some constructs did relate to the benefit of sharing ideas and solving problems together. The concept of *working partnership* and the association of *team work and knowledge* was also evident. More attention was paid in the second cycle to developing team building, negotiation and conflict solving skills amongst students.

In cycle 2, this category again averaged the least score for overall effectiveness for learning (33.64%). Some course elements were seen as offering moderate opportunities for discussion and *sharing learning process*. There was a marked preference for individual work and less for the compulsory team and pair work.

6.4 Summary

Table 2 shows the meta-categories with the mean similarity scores of the categories from both cycles, and the overall scores that show the relationship with the *Overall learnt a lot* construct. The elements supplied in the repertory grids were the learning situations in the course. The meta-categories refer to the elements in terms of effectiveness for learning programming. The students' perceptions revealed that the *learning process* was considered as the most beneficial for learning (49.73%) as compared to the *learning content* (46.02%) or the *learning support* (41.33%).

In summary, the personal constructs elicited in cycle 2 showed a significant shift in the way the students experienced the course. Learning through informational sources gave way to a focus on course activities that led to understanding. Students' perceptions of *thinking* as a way of programming were evident. The benefits of scaffolding were clearly identified in both cycles as playing a role in improving programming competencies. Formative assessments continued to be helpful, while collaborative work was still perceived as problematic. The perceptions of the students indicated they did not quite like working in teams, even if the collaborative work resulted in high quality projects. Support for learning (provided by personal feedback from lecturer and tutor, through adaptive and interactive software, visualizations, and multimedia tutorials) was considered as more conducive for learning than collaborative pair and team work. The investigation of the personal constructs and the perceptions of effectiveness of the learning and teaching activities revealed which factors might be important for the students and should thereby be the focus for further fine tuning of the course design.

Table 2. Perceived overall effectiveness of course components

Meta category	Category	M %	Overall M %
Learning process	Learning through reflection	50.91	49.73
	Learning as experiencing	48.54	
Learning content	Learning through information	49.91	46.02
	Learning by coding	44.90	
	Learning from assessment	43.25	
Learning support	Learning through scaffolding	48.90	41.33
	Learning from collaboration	33.75	

7. DISCUSSION

The repertory grid technique was employed for understanding and to investigate the learning experiences of novice programmers. The benefits of the technique for data gathering and analysis are discussed, along with some limitations that arose. The insights gained from using the technique to evaluate the outcomes of a redesigned introductory OOP course for undergraduate students are shared.

In this study, questionnaires, student journals, work assessments, observations, and repertory grid interviews were employed to gather data. However, the repertory grid provided richer data than any of the other data sources or any of the traditional course evaluation questionnaires used previously by the author. The repertory grid enabled a holistic view of the programme for course evaluation and feedback, and it facilitated the reflection-in-action [38] that is essential for an action research project. It enabled formative monitoring of the influence of the learning environment and summative evaluation of the innovation validity [2] crucial for determining the outcomes of an action research project aimed at improving programming learning outcomes. The person-oriented aspect in personal construct theory allowed a deeper understanding of the experiences of the students, while the acknowledgment of their different perspectives and the suggestions offered for course betterment enabled improvements to the course design.

Three specific benefits [11] of utilizing repertory grids are applicable to this study:

1. By asking the students to construe the same phenomenon (the programming course) in terms of effectiveness for their learning, the repertory grid enabled shared cognitions to be identified without the drawback of using a priori categories;
2. The interviews with the grids allowed the students (most of whom were non-native English speakers) to articulate their experiences, and simultaneously enabled the elicitation of further details;
3. The qualitative and quantitative data that was obtained was rich enough to examine each participant's unique constructs, while permitting rigorous content analysis and reliability checks.

The repertory grid technique contributes to the diversity of computer science research methods that explore the learning experiences of novice programmers. It is appropriate to discuss here the relationship between the technique and the phenomenographic approach to analyzing students' experiences. The outcomes in phenomenographic research are shaped both by the researcher and the object of research [31]. This is the second order perspective by which the researcher discerns qualitatively different ways in which a phenomenon is understood. The individual voices disappear and the categories of outcomes are at the collective level. In the repertory grid technique there is less focus on the researcher's interpretation. The researcher thematically gathers the constructs together while preserving information about each individual's constructs about the phenomenon in the individual's own words. The focus of the study reported in this paper was to learn how students with different learning approaches underwent the course experience, and to use this understanding to tailor the course for better learning outcomes. In the future, it would be a worthwhile exercise to use the phenomenographic approach to investigate qualitatively different ways in which students experience a similarly designed introductory programming course.

Some necessary skills for grid practitioners are the ability to subsume another's construing, to suspend personal values, to listen credulously, and to act with reflexivity [19]. For the author, the development of these skills was not easy to cultivate and became a part of the process of maturing as a researcher. In the role of a researcher, subsuming involved seeing the world through the students' eyes, and even experiencing some of the feelings involved, but also maintaining a sense of self as being separate from the other. The skill of suspending personal values when eliciting the constructs and the skill of truly listening to a student was not easy to cultivate. Although discussed in the context of psychotherapy and counseling, Kelly's [27] notion of a credulous attitude also presented some problems. The students lacked knowledge of the theoretical basis of the course design, which in turn affected their constructs. However, it was difficult to refrain from being defensive in the face of what seemed as unwarranted criticism of the course. Developing the skill of reflexivity also meant examining and finding out some uncomfortable truths about one's own role, actions, idiosyncratic beliefs, and emotions. This research was truly an exploration of the personal constructs of the researcher-practitioner.

There are some limitations to the repertory grid technique. The setup of grids, the construct elicitation, and the data analysis procedures have considerable costs in terms of time and effort. Much of the tedium of doing these tasks can be alleviated by using computer based grid elicitation and analysis software. However, caution needs to be exercised in not over relying on the quantitative grid ratings [20]. In this study, which was predominantly qualitative, precedence was given to the interpretive aspects of construct summaries. The personal theories, while providing rich data for course redesign, are not easily generalizable to other contexts. However, the local knowledge can be applied by reflective transfer to new practice situations [37].

This research study used the repertory grid interview technique to gain insights into how novice programmers experience programming. The construct categories that emerged from the grid data analysis related to the learning process (reflection and experiencing), learning content (information, coding, assessment), and learning support (scaffolding and collaboration). These categories pertain to essential elements in a course design that a teacher needs to attend to influence the learning outcomes. The results illuminate the value of a holistic approach in focusing on all aspects of the learning environment. Students' constructs of their course experience serve to improve a course developer's understanding of the contextual influences on students' learning. In this study, the understanding of how the students viewed the programming activities and experiences provided valuable insights into how to structure the course for more effective learning. These insights are listed here, with the belief that situational understandings can be of universal significance by opening up possibilities for action in other contexts [15].

Learning process: (Learning through reflection; Learning as experiencing)

Integrating reflective writing in programming assignments stimulates students to explore their experiences, to articulate new understandings, and to develop metacognition.

Creating a variety of learning experiences enables students to become more engaged in programming and leads to enrichment of learning.

Learning content: (Learning from assessment; Learning by coding; Learning through information)
Formative programming assignments and projects, that are interesting, challenging, and model real life, give opportunities to students to be creative.

Effective learning is supported and a broad perspective of programming is gained when students actively engage in the planning process (designing with UML diagrams, use cases, and CRC cards) followed by the coding, testing, and refactoring phases.

Educational media (IDEs, visualization software, web resources, and learning objects) can be leveraged to provide powerful learning experiences that enable students to investigate, explore, experiment, and practice programming.

Learning support: (Learning through scaffolding; Learning from collaboration)
Meaningful and timely feedback from peers, tutors, and software, in addition to the feedback given by the lecturer can help students to understand programming errors.

Specific training for team building, conflict resolution, cooperative learning, and negotiation skills should be given to improve pair and team programming.

8. CONCLUSIONS

In this study, the repertory grid technique addressed epistemological concerns about ways of experiencing an introductory OOP course. It offered the basis for mixing methods in an action research project. It was underpinned by personal construct theory and grounded in constructivist and postpositivist paradigms. It enabled the blending of qualitative and quantitative methods to elicit personal constructs that embodied novice programmers' experiences in their own voice. The data illustrated the students' response to the shared course context, which in turn influenced refinements to the course design. At a personal level, the reflection on the personal constructs of the students shed new light on the teaching practice and was an empowering experience for an educator considering a theoretical framework for an introductory programming course design.

9. ACKNOWLEDGMENTS

The author wishes to thank the students and colleagues who participated in the study.

10. REFERENCES

[1] Alexander, P. and Van Loggerenberg, J. 2005. The repertory grid: "Discovering" a 50-year-old research technique. In *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, (White River, South Africa, September 20 - 22, 2005), 192-199.

[2] Bain, J.D. 1999. Introduction [Special Issue on Evaluation]. *Higher Education Research & Development*, 18, 2, 165-172.

[3] Bennedsen, J. and Caspersen, M., E 2007. Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39, 2 32-36.

[4] Biggs, J.B., Kember, D. and Leung, D. 2001. The revised two-factor Study Process Questionnaire: R-SPQ-2F. *British Journal of Educational Psychology*, 71, 1, 133-149.

[5] Booth, S. 1992. *Learning to Program: A Phenomenographic Perspective*. Acta Universitatis Gothoburgensis, Göteborg, Sweden.

[6] Boyle, T. 2005. Improving team performance using repertory grids. *Team Performance Management*, 11, 5/6, 179-187.

[7] Bruce, C., McMahon, C., Buckingham, L., Hynd, J., Roggenkamp, M. and Stoodley, I. 2004. Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. *Journal of Information Technology Education*, 3, 143-160.

[8] Carbone, A., Hurst, J., Mitchell, I. and Gunstone, D. 2009. An exploration of internal factors influencing student learning of programming. In *Proceedings of the 11th Australasian Computing Education Conferenc,* Hamilton, M. and Clear, T. Eds., ACS, Darlinghurst, Australia, 25-34.

[9] Creswell, J.W. and Plano Clark, V.L. 2007. *Designing and Conducting Mixed Methods Research*. Sage, Thousand Oaks, CA.

[10] De Raadt, M., Watson, R. and Toleman, M. 2009. Teaching and assessing programming strategies explicitly. In *Proceedings of the 11th Australasian Computing Education Conference*, Hamilton, M. and Clear, T. Eds., ACS, Darlinghurst, Australia, 55-64.

[11] Dick, P. and Jankowicz, D. 2001. A social constructionist account of police culture and its influence on the representation and progression of female officers: A repertory grid analysis in a UK police force. *Policing: An International Journal of Police Strategies and Management*, 24, 2, 181-199.

[12] Easterby-Smith, M., Thorpe, R. and Holman, D. 1996. Using repertory grids in management. *Journal of European Industrial Training*, 20, 3, 3-30.

[13] Eckerdal, A. 2006. *Novice students' learning of object-oriented programming.* Licentiate thesis, Uppsala University, Sweden.

[14] Edwards, H.M., McDonald, S. and Young, S.M. 2010. Choosing field methods: A reflection on a RepGrid study. In *Proceedings of the Sixth Nordic Conference on Human-Computer Interaction: Extending Boundaries*, (Reykjavik, Iceland, 2010), ACM, New York, NY, 639-642.

[15] Elliott, J. 2009. Building educational theory through action research. In *Handbook of Educational Action Research*, Noffke, S.E. and Somekh, B. Eds., Sage, London, UK, 28-38.

[16] Entwistle, N. 2005. Contrasting perspectives on learning. In *The Experience of Learning: Implications for Teaching and Studying in Higher Education*, Marton, F., Hounsell, D. and Entwistle, N. Eds., University of Edinburgh, Centre for Teaching, Learning and Assessment, Edinburgh, 3-22.

[17] Ezzy, D. 2002. *Qualitative Analysis: Practice and Innovation*. Routledge, London, UK.

[18] Fisher, B., McSweeney, P. and Russell, T. 1991. The application of repertory grid technique to course evaluation -

A pilot project. *Assessment & Evaluation in Higher Education*, 16, 2, 109-132.

[19] Fransella, F. 2005. Some skills and tools for personal construct users. In *The Essential Practitioner's Handbook of Personal Construct Psychology*, Fransella, F. Ed. Wiley, West Sussex, UK, 41-56.

[20] Fransella, F., Bell, R.C. and Bannister, D. 2004. *A Manual for Repertory Grid Technique*. Wiley, West Sussex, UK.

[21] Gaines, B.R. and Shaw, M.L.G. 2003. *Personal construct psychology and the cognitive revolution*. University of Calgary-Knowledge Science Institute Web site: http://pages.cpsc.ucalgary.ca/~gaines/reports/PSYCH/SIM/SIM.pdf (Accessed May 21, 2010).

[22] Green, B. 2004. Personal construct psychology and content analysis. *Personal Construct Theory & Practice*, 1, 3, 82-91.

[23] Guzdial, M. and Forte, A. 2005. Design process for a non-majors computing course. *ACM SIGCSE Bulletin*, 37, 1 361-365.

[24] Hayes, A.F. and Krippendorff, K. 2007. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1, 77-89.

[25] Honey, P. 1979. The repertory grid in action: How to use it to conduct an attitude survey. *Industrial and Commercial Training*, 11, 11, 452-459.

[26] Jankowicz, D. 2004. *The Easy Guide to Repertory Grids*. Wiley, West Sussex, UK.

[27] Kelly, G.A. 1955. *The Psychology of Personal Constructs*. Norton, New York, NY.

[28] Kreber, C., Castleden, H., Erfani, N., Lim, J. and Wright, T. 2003. Exploring the usefulness of Kelly's Personal Construct Theory in assessing student learning in science courses. *Teaching in Higher Education*, 8, 3, 431-445.

[29] Lahtinen, E., Ala-Mutka, K. and Järvinen, H. 2005. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37, 3 14-18.

[30] Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. and Thomas, L. 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36, 4 119-150.

[31] Marton, F. and Booth, S. 1997. *Learning and Awareness*. Laurence Erlbaum Associates, Mahwah, NJ.

[32] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33, 4 125-140.

[33] Neuendorf, K. 2002. *The Content Analysis Guidebook*. Sage, Thousand Oaks, CA.

[34] Patton, M.Q. 1990. *Qualitative evaluation and research methods*. Sage, Newbury Park, CA.

[35] Postner, L. and Stevens, R. 2005. What resources do CS1 students use and how do they use them? *Computer Science Education*, 15, 3 165-182.

[36] Robins, A., Rountree, J. and Rountree, N. 2003. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13, 2, 137-172.

[37] Schön, D.A. 1995. Knowing-in-action: The new scholarship requires a new epistemology. *Change*, 27, 6, 26-34.

[38] Schön, D.A. 1983. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, NY.

[39] Soh, L., Samal, A. and Nugent, G. 2007. An integrated framework for improved computer science education: Strategies, implementations, and results. *Computer Science Education*, 17, 1, 59-83.

[40] Solas, J. 1992. Investigating teacher and student thinking about the process of teaching and learning using autobiography and repertory grid. *Review of Educational Research*, 62, 2, 205-225.

[41] Stein, S.J., Docherty, M. and Hannam, R. 2001. Making the process of design explicit within an information technology environment. *Annual Conference of the AARE*, University of Notre Dame, Fremantle, WA, Australia. http://www.aare.edu.au/01pap/ste01296.htm (Accessed May 21, 2010).

[42] Stein, S.J., McRobbie, C.J. and Ginns, I. 1998. Insights into pre-service primary teacher's thinking about technology and technology education. *Annual Conference of the AARE*, Adelaide, SA, Australia. http://www.aare.edu.au/98pap/mcr98085.htm (Accessed May 21, 2010).

[43] Stewart, V. and Stewart, A. 1981. *Business Applications of Repertory Grid*. McGraw-Hill, Maidenhead, UK.

[44] Tan, F.B. and Hunter, M.G. 2002. The repertory grid technique: A method for the study of cognition in information systems. *MIS Quarterly*, 26, 1, 39-57.

[45] Tew, A.E., Fowler, C. and Guzdial, M. 2005. Tracking an innovation in introductory CS education from a research university to a two-year college. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, (St. Louis, Missouri, USA, 2005), ACM, 416-420.

[46] Thota, N. and Whitfield, R. 2010. Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education*, 20, 2 103-127.

[47] Yorke, D.M. 1987. Construing classrooms and curricula: A framework for research. *British Educational Research Journal*, 13, 1, 35-50.

[48] Zuber-Skerritt, O. 1992. *Action Research in Higher Education: Examples and Reflections*. Kogan Page, London, UK.