In P. Romero, J. Good, E. Acosta Chaparro & S. Bryant (Eds). Proc. PPIG 17 Pages 255 - 267

A Framework for Evaluating Qualitative Research Methods in Computer Programming Education

Enda Dunican

Computing Department,
Institute of Technology Carlow, Ireland,

Enda.Dunican@itcarlow.ie

Abstract. This work-in-progress paper[1] examines in detail the issue of qualitative research method evaluation. In particular, it focuses on method evaluation from the perspective of Computer Science education research. To this end, it presents the reader with an evaluative framework. The key issue underlying the development of this framework is *pragmatism*, i.e. one that provides the researcher with a simplistic and context-specific evaluation technique. In terms of simplicity, it proposes a *broad-brush* approach to evaluation where approximation is used and based on a number of evaluative criteria. In addition, the framework is based on a set of properties that may be construed as pragmatic, in terms of them reflecting issues of real importance to a researcher, who is attempting to decide on which qualitative method to use in a given context. An example of framework utilisation is then presented with respect to the *Grounded Theory* research method in the context of computer science education research. This entails a multiple-litmus type test on the Grounded Theory research method where measurements are presented for each of the framework properties. Finally, the paper will draw attention to the reusability of the framework by presenting its structure from a meta-perspective and indicates that it can be used to evaluate any qualitative research method in any context.

1 Introduction

The modern Computer Science (CS) education researcher is faced with a plethora of research methods of both a qualitative and quantitative nature. With this vast array of research methods at their disposal, the researcher may find it difficult to determine which method (or group of methods) is potentially suitable for their research needs.

[1] This work-in-progress paper is a part of a larger project that is using the Grounded Theory method to generate theory pertaining to the difficulties experienced by novice programming students in Irish third-level institutes.

The purpose of this paper is to develop a framework to evaluate qualitative research methods that can be used to assess their suitability for computer programming education research. The framework is based on the development of a unique set of properties that will serve as evaluation criteria, under which a qualitative research method may be critically analysed. Implementation of the framework on a particular qualitative research method, namely Grounded Theory will be presented in this paper.

This paper is written from the perspective of a CS researcher entering the field of qualitative research. It takes a pragmatic approach, towards research method evaluation. In essence, it takes the perspective of a researcher concerned with finding a method that suits their particular research question and research context and is akin to the perspective of Kendall [25] who presents to the reader 'not through the voice of a methodologist, but from the experience of an ordinary researcher'(p.756) trying to make sense of the social context of his/her research and other ancillary concerns. Finally, it is hoped that a simple and accessible framework of this type will provide CS education researchers with the armoury required to make an informed choice in terms of qualitative methods. This can provide them with a mechanism to assess the potential viability of qualitative research methods as either alternatives to traditional quantitative methods or as potential candidates for mixed method implementation.

Grounded Theory

Grounded Theory (GT) is an inductive qualitative research method. Rather than starting with a hypothesis and trying to prove it, the GT researcher begins by collecting data in the field and lets the theory emerge or emanate from the data. In this regard, it is postulated that the theory is actually *grounded* in the data. Data is usually in the form of interview transcripts or observational notes. Research subjects are chosen using *theoretical sampling* which is based on their potential for contribution to the development of theory. Conducting GT research entails a number of levels of coding and analysis. *Open coding* examines the text for items of interest, with the ultimate aim of accumulating codes into categories. Here the researcher uses the *constant comparative* approach where they constantly compare new instances of the category with those already encountered until he/she saturates the category (i.e. no new insights in the category can be gained from the data). *Axial coding* entails relating categories to their sub-categories around the axis of a central category, based on linkages between their properties. *Selective coding* entails identifying a central phenomenon and relating central categories to it using statements of relationships. Very often, in selective coding, a 'storyline' is generated that narrates the categories and their relationships [22]. The net outcome of GT research is a theory that contains a central phenomenon, its causal conditions, its intervening conditions and its consequences.

Conceptual Origins of the Framework

The framework proposed in this paper provides the prospective researcher with a set of tools for research method evaluation. The conceptual origin of this type of evaluation lies in the *Cognitive Dimensions* framework developed by Green [20]. This framework proposes a list of dimensions that provides the user with a mechanism with which they can evaluate information-based artefacts e.g. visual programming languages. In terms of the cognitive dimensions presented in his framework, Green [20] states that taken together they 'describe enough aspects to give a fair idea of how users will get on with the system'(p.3). In terms of this framework, the litmus-type test presented in this paper can be ascribed to Green's approach whereby it can provide the prospective researcher with an insight into how suitable a given research method is to their research area/question. Green and Petre [21] describe this evaluation approach as 'broad-brush' whereby the user can evaluate 'cheaply' to derive approximate values. A further similarity between this framework and that of cognitive dimensions pertains to the notion of *overlap*. Green, when describing his list of dimensions indicates that there is unavoidable overlap between them. Avoidance of overlap when developing this framework proved just as elusive e.g. issues pertaining to 'sampling' and 'prior knowledge' were found to be applicable to both the properties of *load* and *vagueness of implementation.*

Framework Overview

This framework provides the prospective researcher with a set of properties that can be applied to a research method in a given research context. These 'properties' have been developed to derive a unique set of criteria that can be applied to qualitative research methods with the aim of ascertaining their suitability or effectiveness in terms of substantive computer programming education research. In terms of each property, the respective research method is analysed in order to determine its position along a continuum in terms of the issue at hand. For example, a given research method will display a level of 'conceptual overlap' (a framework dimension that will be described later), somewhere on a continuum between high and low. The key requirements in devising the properties presented in the framework were conceptual simplicity and context specificity. The properties consist of the following

Conceptual Overlap - Reflects how much overlap there exists between the core principles of the research method and the discipline-specific background of the researcher.

Methodological Overlap - Reflects the level of core methodological similarity between the discipline-specific methodologies familiar to the researcher and those of the research method. The term 'methodological' is used to represent the key activities undertaken in the chosen research method at both a process level and a data gathering/acquisition level.

Load - Reflects the level of demand imposed by the research method on the researcher within the research context. It reflects the level of challenge required to develop a piece of coherent and substantive work using the research method within the chosen context.

Vagueness of Implementation - Reflects the level of methodological variation, unclearness or "fuzziness" with respect to the research method application.

Fitness for Purpose - Reflects how well the chosen research method is suited to the context of study. This is measured in terms of how well it is suited to answering the issues posed in the research question.

Conceptual Overlap

This property represents the conceptual similarities between CS-related principles and theory espoused by the respective research method. Its purpose is to isolate the core conceptual issues in the respective research method and ascertain its level of congruence with fundamental CS concepts with which, CS education researchers will be familiar. It is probable that a high level of this property enhances ease of implementation and speeds up both the process of methodological choice and actual implementation. Furthermore, it may result in the research method being conceptually appealing to the researcher given the high level of commonality in terms of concept and implementation. The rationale behind this particular assertion stems from the notions of *programming by analogy* and *reuse* that permeate the CS discipline [23], [32]. Computer programmers very often excel at methodological application when they can form an analogy between the task at hand and concept they have experienced before.

Theory building in GT is focussed around the notion of a *category* [22].In addition to this, the notion of *abstraction* is paramount in terms of category development. In simple terms, the researcher analyses the relevant narrative for the existence of codes. On completion of this, the codes are analysed and those that conform to a common theme are grouped together. This higher order commonality is referred to as a concept [1]. Concepts are then grouped into areas of commonality to form the highest-level abstract notion, namely that of a *category*. Furthermore, in relation to category abstraction, the notion of the 'subcategory' emerges.

As well as the classification of identified categories, grounded theorists are also concerned with the *properties* and *dimensions* associated with each category. Properties represent the attributes or characteristics of a category, whilst dimensions refer the range of variability of a given property. Therefore, the researcher must be aware of all aspects of a category i.e. in terms of how it encapsulates properties, dimensions and behaviour.

In CS these concepts of abstraction, classification and sub-categorisation discussed in terms of GT above, are also viewed as central principles in terms of computer program development. Computer programming is concerned with the identification of *objects* that are essentially entities in the real world about which data must be stored, and behaviour must be implemented. Furthermore, objects consist of attributes that represent the properties of the object. Classification entails the grouping together of objects that have a common set of properties (attributes) and operations. In relation to abstraction the concept of 'generalisation/specialisation' is applied. This represents an approach where attributes and behaviours that are common to several types of object classes are grouped (or abstracted) into their own class, known as a *supertype*. In turn, the attributes/properties and behaviour of the supertype object are inherited by those object classes i.e. *subtypes*.

In summary, developers of computer systems will frequently want to view the system from different levels of abstraction and this is considered to be an intrinsic aspect of software development. Likewise, Pidgeon and Henwood [30] view this as an essential aspect of GT where they indicate that 'the success of the initial coding will depend in part on choosing an appropriate level of abstraction for the concepts in question'(p.93).

Methodological Overlap

This property evaluates the level of core methodological similarity between computer program development and the respective research method under consideration. The term 'methodological' is used to represent the key activities undertaken in the chosen research method at both a process level and a data gathering/acquisition level.

At a process-level, GT implementation essentially entails the process of alternating between data collection and analysis on an iterative basis, a technique ubiquitously referred to as the *constant comparative* method of data analysis. Specifically, this type analysis involves taking a piece of data and comparing it with emerging categories in terms of how it is similar and different with the aim of categorisation and subsequent theory generation. Pidgeon and Henwood [31] describe GT analysis as an iterative process, where 'researchers often move between steps (and the steps merge into one another) as the analysis proceeds' (p.87). Furthermore, successive iterations through this process enable the analyst to further refine the theory until theoretical saturation is reached. The iterative approach to process is commonplace in computer software development e.g. evolutionary prototyping where successive levels of iteration through the prototyping process move the developer closer towards the completed system.

The process of open coding in GT entails an analysis of the data for the existence of *concepts*. Essentially the researcher scans through the text for items of interest.

Borgatti [8] refers to these concepts as 'nouns and verbs of a conceptual world'(p.2). Concepts are then examined for their properties and dimensions.

Within software development, identification of entities/objects in occurs in a similar vein. According to Whitten [10] 'many methodology experts recommend the searching of the requirements document or other associated documentation and underlining the nouns that may represent potential objects'(p.454). Once these nouns are accepted as objects/entities in the proposed system, the developer goes about identifying the attributes of each one.

Load

This property represents the level of demand imposed by the research method on the researcher within the research context[2]. It reflects the level of challenge required to develop a piece of coherent and substantive work using the research method. Given the elaborate nature of GT in terms of conceptual and methodological scope, the application of this property to GT raises significant discussion.

On the initial reading of GT literature, the first noticeable issue encountered in this research was the apparent demanding nature of this method. Turner [18] supports this assertion when he refers to the 'demanding process of interpreting research data' (p.227) in a GT study. Analysis of GT literature suggests that it might be a technique that is difficult to implement in a proper and comprehensive manner. This observation seems to be echoed by other commentators [3], [14], [33].

Yet another demand imposed by the GT method is the notion of elegance. Brown et al. [9] describe elegance as a desirable characteristic of a GT study where the fewest possible concepts are generated with the widest possible scope. Glaser [19] refers to elegance by highlighting the need for developing concepts with 'as much variations as possible in the behaviour and problem under study'(p.125). Essentially, elegant GT develops theory where multiple manifestations of a concept are identified and described. The demand for elegance is a requirement that is not alien to computer programmers. Developing computer programs that implement elegant algorithms is often viewed as a tacit aim of many computer programmers. In summary, given the CS researcher's background, when considering which research method to use, one that imposes a need for elegance in a similar importunate fashion to that of their own discipline may require significant consideration before committal.

The issue of prior subject-specific knowledge also imposes demand on the GT researcher and this has significant relevance within CS research. Given the fact that many CS education researchers are involved as programming educators, there is a

[2] This is similar in concept to *Hard Mental Operations*, Green and Petre [21]

danger that they bring their biases with them to a study. Furthermore, the researcher must conduct a fine balancing act between eliminating their bias and possessing the required level of sensitivity to the data they are collecting. In this regard, Strauss and Corbin [22] present the GT-related notion of *theoretical sensitivity* as the 'attribute of having insight, the ability to give meaning to data, the capacity to understand and the capacity to separate the pertinent from that which isn't' (p.42).The potential danger here is that if the researcher fails in this aforementioned fine balancing act, the emergent theory may be based on assumptions, bias or speculation. This has conceptual similarity to the 'premature commitment' cognitive dimension described by Green and Petre [21]. In fact, Charmaz [11] identifies 'premature commitment' as a potential weakness of the GT method where she states that 'premature commitment to categories means that the researcher has not fully explored the issues, events and meanings within the research problem or setting' (p. 1164). Safeguarding against bias in these instances can be achieved by the researcher constantly asking themselves whether or not the concept under investigation has originated from themselves or their interviewees.

On the other hand, within the CS discipline, prior knowledge of a concept, task or algorithm is often viewed as beneficial, in terms of enabling the programmer to reuse their previously-acquired task knowledge in a given situation. In describing the argument made by the GT practitioners who consider prior knowledge an impediment to data-based theory building Cutcliffe [12] states that 'instead of allowing the theory to emerge from the data, the researcher, albeit implicitly, is likely to enter into a deductive process'(p.1481). This situation may be exacerbated by the fact that most computer programming tasks are deductive. Partridge [4] when referring to CS states 'the underlying framework of this science has always been based on deduction (reasoning from the general to the specific) rather than induction (reasoning from the specific to the general)'(p.36). In this context, it may be more likely for a CS practitioner to unwittingly drift back into a deductive reasoning process.

The final issue in relation to load pertains to the incorrect partial use of the GT method. This results from researchers isolating and utilising only some aspects of the GT method and ignoring other aspects given their perceived complex or demanding nature [2]. According to Woods et al [5] many researchers purporting theory generation in their research are actually only partially using GT techniques and fall short of a 'true grounded theory'(p.35). They go on to point out that many studies claiming to be GT research 'often fail to develop any substantive theory'(p.35). Furthermore, when faced with a complex methodology, analysts may be tempted to 'cherry pick' parts of the method that are less demanding that others. Once again, this issue is related to the notion of inelegant theory described earlier. In the context of CS, one may often encounter this type of problem where programmers utilise 'quick-and-dirty' [26] solutions to a programming problem, only for this shortcut to manifest itself as a defect later on in the software systems life-cycle. The further down the life-cycle these defects are uncovered, the more costly they are to fix. Slaughter et al [29] advocate 'fixing defects as early in the software life-cycle as possible, because the cost of correction increases the later in the process the defect is discovered' (p.68). It may be

likely that a lack of rigorous adherence to true grounded theory in a similar 'quick-and-dirty' fashion could result in flawed research design that is difficult to remedy further down the research project life-cycle.

Vagueness of Implementation

This property reflects the level of methodological variation or "fuzziness" within the research method. A research method exhibiting a high level of this property will have numerous variations on how it can be implemented and may present difficulty for the researcher, leaving him/her unsure as to whether or not they have used the correct variation of the method with respect to their research question and whether or not they have utilised in a proper and comprehensive manner. It is desirable that a research method exhibit a low level of this property.

The first issue with respect to this property relates to the two different schools of thought in GT i.e. the Glaser versus Strauss philosophical debate. In this regard it is incumbent on the researcher to make explicit which version they are using. In light of these numerous variations, Esteves et al. [15] advise the researcher that there are 'philosophical and methodological assumptions that must be taken into account when selecting and adopting a specific approach'(p.135). Interested researchers are directed the significant literature on this topic ([17],[34]) which is outside the scope of this paper.

There also seems to be confusion and differing opinions in the literature on method implementation. Cutcliffe [12] indicates that 'there appears to be conflicting opinions and unresolved issues regarding the nature and process of grounded theory'(p.1476). Vagueness seems to be exacerbated by a lack of comprehensive examples in GT literature. According to Boeije [7] 'the literature does not make clear how one should go about constant comparison nor does it address such issues as whether different types of comparison can be distinguished'(p.393). In terms of the actual coding process itself to Brown et al. (2002) argue that 'the lines between the three coding levels are blurred'(p.4). In terms of the type of sampling used Cutcliffe [12] indicates that 'it is reasonable to say that the literature on this is confusing and conflicting'(p.1478).

To intensify the level of vagueness exhibited by GT, there seems to be conflicting viewpoints on prior knowledge. Glaser [19] advocates collecting data in advance of engaging in a literature review in order that 'the theory will not be preconceived by pre-empting concepts'(p.31). Smith and Biley [27] take a less extreme position than Glaser by suggesting that a general reading of the literature is acceptable as long as 'the reading is not too extensive'(p.20). Strauss and Corbin [22] suggest that a literature review be conducted in a prudent fashion. They indicate that 'familiarity with relevant literature can enhance sensitivity to subtle nuances in the data just as it can block creativity'(p.49). In terms of CS education research it is likely that the researcher will possess *a priori* knowledge of computer programming acquired from both literature and professional experience, and there it is incumbent upon him/her to

adopt the aforementioned prudent approach so that the identification of new phenomena is not stifled by this pre-existent knowledge.

In summary, given the extensive availability of literature on GT and its numerous variations, one may encounter methodological difficulty given its propensity for vagueness. Therefore, it is incumbent on the CS researcher to exercise prudence in its utilisation.

Fitness for Purpose

This property was devised to reflect how well the chosen research method is suited to the context of study. A method containing a high level of this property is capable of answering the questions that need to be answered in a research context e.g. novice computer programming in Irish third-level institutes. It should be noted that the label of this property should not be confused with 'fit', a notion which is seen as a desirable characteristic of GT where the theory generated 'fits' the actual data collected and is not derived from bias or other non-data-related sources.

In terms of assessing the level of this property it is important to highlight in advance the nature of the research context. We take novice programming in Irish third level institutions as an example research context, an area where qualitative research appears to be lacking, with most published research to date being of a quantitative nature reflecting a task-oriented basis (where students fill in questionnaires or engage in an aptitude-type test, e.g. [6], [13], [24]).

It seems logical that the selection of any research method should be based on the nature of the research question. [9]. According to Pidgeon and Henwood [31] GT is 'particularly suited to the study of local interactions and meanings as related to the social context in which they actually occur'(p.75). Priest et al [33] go a step further by indicating that it 'aims to generate theory through inductive examination of data in subject areas that may be difficult to access with traditional quantitative methods'(p.31). In this regard, it may happen that quantitative research conducted to date may not explore, in a profound way the individual difficulties experienced by students in their particular educational setting. In determining the suitability of GT to remedy this situation Pidgeon and Henwood [31] postulates that 'grounded theory places great emphasis upon the attention to participants' own accounts of social and psychological events and of their associated local phenomenal and social worlds'(p.76). As mentioned earlier, this reflects the status of Irish novice computer programming research, an area that is not short of valuable research (albeit mostly of a quantitative nature), yet to a large extent, devoid of context-specific substantive theory and constructs.

In summary, given the issues discussed so far with respect the application of this property to GT it appears that GT is ideally suited to studying the complex issues pertaining to novice students' experiences with computer programming. Therefore, it

may be likely that GT will not only enable the researcher to classify the types of problem experienced in novice computer programming, but also shed light on how students actually deal with these problems.

Using the Framework

In practical terms, using the framework entails the application of each of the framework properties to a chosen research method with the simple aim of ascertaining its suitability for the chosen research project. Fig 1 illustrates the application of the framework to the GT research method in terms of the CS education research context.

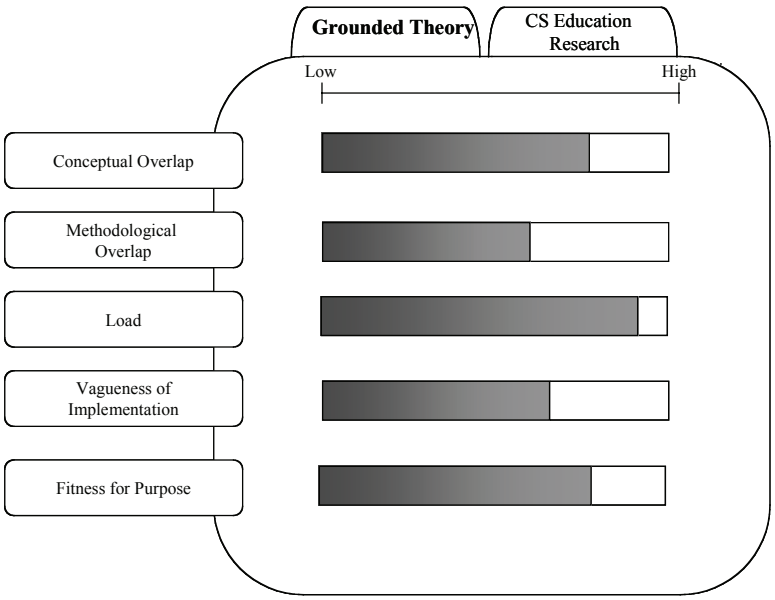

Fig 1 : Applying the Framework

From a visual and conceptual perspective the utilisation of the framework depicted in fig 1 resembles a litmus-type test. Obviously, unlike a litmus test that relies on a single indicator, this framework utilises a number indicators that are represented by measures attributed to each of the properties. The levels presented for each property are approximate values that are based on the analysis of GT in terms of the literature researched and the authors significant experience in the area. This approximation-type measurement is based on the Cognitive Dimensions 'broad-brush' approach to evaluation described by Green and Petre [21].

It is clear from fig 1 that GT has a desirable measure for all of the framework properties apart from *vagueness of implementation* and *load*. In practical terms, these high levels may not prove to be major impediments but rather serve as warning mechanisms, that encourage the researcher to conform to rigorous and careful

analysis. To summarise the depiction in fig 1, GT has high conceptual and methodological overlap with CS (in particular computer programming) and is suitable(fit) for research in this context. However, given the high level of demand(load) associated with GT and its propensity for vagueness in certain aspects of its analysis, the researcher should progress with prudence and precision in order to avoid bias, flawed theory generation and other potential pitfalls. In light of this, the framework utilisation suggests that GT is a suitable research method in this context.

Conclusion

To date, the majority of CS education research in Irish third-level institutes has been of a quantitative nature. In order to address this deficiency, mechanisms need to be devised that enable researchers to utilise the rich analytical power present in qualitative techniques. In pursuit of this objective, this paper has presented a framework that enables a researcher to evaluate in a pragmatic and simplistic fashion, the suitability of a candidate qualitative research method to research in this CS education context. To this end, framework has derived a set of properties that are conceptually simple and context-specific. The evaluation technique used in this framework enables a 'quick-and-easy' determination of research method suitability/usability, where the fundamental principles of the candidate method are isolated and analysed under the collective lenses of the framework properties. The net result of this analysis is a multi-litmus-type indication pertaining to the suitability of the method with respect to the research context. The simplistic structure of the framework permits both extensibility and transferability. In terms of the former, additional properties can be derived for the chosen context. In terms of transferability, when viewed from a meta-perspective the framework can be utilised with any qualitative method in any context. In this regard, it is likely that the properties presented in this paper may applicable to many contexts. It is hoped that development of frameworks of this nature will make qualitative research methods more accessible to the CS community at large where the possibilities for stimulating and illuminative qualitative research are endless.

References

1. Allan, G. (2003). 'A Critique of Using Grounded Theory as a Research Method'. Electronic Journal of Business Research Methods. Vol. 2. Issue 1. (pp. 1-10).
2. Babchuk, W.A. (1997) 'Glaser or Straus ?: Grounded Theory and Adult Education'. Midwest Research-To-Practice Conference in Adult, Continuing and Community Education. Michigan 1997. (pp. 1-8).
3. Backman, K. (1999). 'Challenges of The Grounded Theory Approach to a Novice Researcher'. Nursing and Health Sciences. Vol. 1. (pp. 147-153).
4. Partridge, D. (1997). 'The Case for Inductive Programming'. *IEEE Computer*. Vol. 30. No. 1. (pp. 36-41).

5. Woods, L. et al. (2002). 'An Overview of Three Different Approaches to the Interpretation of Qualitative Data. Part 2 : Practical Illustrations. Nurse Researcher. Vol. 10. No. 1. (pp. 43-51).
6. Bergin, S. Reilly, R. (2005). 'Programming: Factors that Influence Success'. Proceedings of Annual Conference of Special Interest Group in Computer Science Education (SIGCSE).Norfork, St. Louis, Missouri. (pp. 411-415).
7. Boeije, H. (2002). 'A Powerful Approach to the Constant Comparative Method in the Analysis of Qualitative Interviews'. Quality & Quantity. Vol. 36. (pp. 391-409).
8. Borgatti, S. (2004). 'Introduction to Grounded Theory'. (Online - http://www. analytictech.com/mb870/introtoGT.htm 28/11/04).
9. Brown, S.C. et al. (2002). 'Exploring Complex Phenomena: Grounded Theory in Student Affairs Research'. Journal of College Student Development. March/April 2002. Vol. 43. No. 2. (pp1-11).
10. Whitten, J. et al. (2004). Systems Analysis and Design Methods. 6th Edition. McGraw-Hill.
11. Charmaz, K. (1990). 'Discovering Chronic Illness : Using Grounded Theory'. *Social Science and Medicine*. Vol. 30. No. 11. (pp. 1161-1172).
12. Cutcliffe, J.R. (2000). 'Methodological Issues in Grounded Theory'. Journal of Advanced Nursing. Vol 31. No. 6. (pp.1476-1484)
13. Daly, C. Waldron, J. (2004) 'Assessing the Assessment of Programming Ability'. Proceedings of Annual Conference of Special Interest Group in Computer Science Education (SIGCSE).Norfork, Virginia. (pp. 210-213).
14. Douglas, D. (2003). 'Inductive Theory Generation: A Grounded Approach to Business Enquiry'. Electronic Journal of Business Research Methods. Vol. 2. Issue 1. (pp. 47-54).
15. Esteves, J. et al. (2002) 'Use of Grounded Theory in Information Systems Area : An Exploratory Analysis'. Proceedings of the European Conference on Research Methodology for Business and Management'. (pp. 129-136).
16. Franchuk, J.E. (2004). 'Phenomenology vs. Grounded Theory: The Appropriate Approach to Use to Study the Perspectives of Instructional Librarians on Their Roles in Information Literacy Education for Undergraduates'. (www.slis.ualberta.ca/ cap04/judy/paper%20page%203.htm – 25/02/2005)
17. Glaser, B. (2004). 'Remodelling Grounded Theory'. Forum Qualitative Science. Vol. 5. No. 2.
18. Turner, B. (1981). 'Some Practical Aspects of Qualitative Data Analysis : One Way of Organising the Cognitive Processes Associated with the Generation of Grounded Theory'. Quality and Control. Vol. 15. (pp. 225-245).
19. Glaser, B. (1978). Theoretical Sensitivity. Mill Valley CA. Sociology Press.
20. Green, T.R.G. (1996) 'An Introduction to the Cognitive Dimensions Framework'. Extended abstract of invited talk at MIRA workshop, Monselice, Italy.(http://homepage.ntlworld.com/greenery/workStuff/Papers/introCogDims /index.html - 20/11/2004)
21. Green, T.R.G., Petre, M. (1996). 'Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework'. Journal of Visual Languages and Computing. Vol. 7. (pp. 131-174).

22. Strauss, A. Corbin, J.(1998). Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage. Second Edition.
23. Harandi, M.T. 91993). 'The Role of Analogy in Software Reuse. Proceedings of ACM/SIGAPP symposium on Applied Computing : States of the Art and Practice. (pp. 40-47).
24. Hyland, E. Clynch, G. 'Initial Experiences Gained and Initiatives Employed in the Teaching of Java Programming in the Institute of Technology Tallaght.' Proceedings of Conference on Principles and Practice of Programming in Java 2002. Dublin. (pp. 101-106).
25. Kendall, J. (1999). 'Axial Coding and the Grounded Theory Controversy'. Western Journal of Nursing Research. Vol. 21. No. 6. (pp. 743-757)
26. Kremer, R. (2004). 'Software Engineering – The Problem'. (http://sern.ucalgary.ca/ courses/cpsc/333/w2005/Problem.html - 18/1/2005)
27. Smith, K. Biley, F. (1997). 'Understanding Grounded Theory'. Principles and Evaluation'. Nurse Researcher. Vol. 4. (pp.17-30).
28. Morse, J.M. (1991). 'Qualitative Nursing Research : A Free for All'. Qualitative Nursing Research : A Contemporary Dialogue.(Morse, J.M. ed) Sage, London. (pp. 14-22)
29. Slaughter, S.A et al. (1998). 'Evaluating the Cost of Software Quality'. Communications of the ACM. Vol. 41. No. 8. (pp.67-73)
30. Pidgeon, N. (1996). 'Grounded Theory : Theoretical Background'. Handbook of Qualitative Research Methods for Psychology and the Social Sciences. J.T. Richardson (ed). (pp. 75-85)
31. Pidgeon, N. Henwood, K. (1996). 'Grounded Theory : Theoretical Practical Implementation'. Handbook of Qualitative Research Methods for Psychology and the Social Sciences. J.T. Richardson (ed). (pp. 75-85)
32. Repenning, J. Perrone, C. (2000). 'Programming by Analogous Examples. Communications of the ACM . Vol. 43. No. 3. (pp. 90-97).
33. Priest, H. et al. (2002). 'An Overview of Three Different Approaches to the Interpretation of Qualitative Data. Part 1 : Theoretical Issues. Nurse Researcher. Vol. 10. No. 1. (pp. 32-42).
34. Rennie, D. (1998). 'Grounded Theory Methodology. The Pressing Need for a Coherent Logic of Justification'. Theory & Psychology. Vol. 8. No. 1. (pp. 101-119).