2000

The Role of Interpretive Evaluation in Engineering Information Systems Requirements

Galal H. Galal
Middlesex University - U.K., galal@acm.org

Janet T. McDonnell
University College London, j.mcdonnell@cs.ucl.ac.uk

Ray J. Paul
Brunel University, ray.paul@brunel.ac.uk

The Role of Interpretive Evaluation in Engineering Information Systems Requirements

Galal H Galal, School of Computing Science, Middlesex University, London, UK, Galal@acm.org
Janet T McDonnell, Department of Computer Science, University College London, UK, J.McDonnell@cs.ucl.ac.uk
Ray J Paul, Department of Information Systems and Computing, Brunel University, UK, Ray.Paul@Brunel.ac.uk

Abstract

The requirements for complex systems inevitably change continuously. Any successful software or information systems engineering approach needs to observe this simple fact. This paper argues for the critical importance of formative evaluation activities in any non-sequential, or learning-based RE process. We argue that evaluation with its focus on understanding and interpreting the evaluation results, is distinct from measurement. We also outline how evaluation activities can be performed from an interpretivist perspective, in a way that systematically informs formative evaluation activities during gradual, experimental requirements engineering activities.

Key words

Requirements engineering; non-sequential process models, interpretive evaluation; qualitative data in requirements engineering; qualitative research methods in requirements engineering.

Introduction

It has been repeatedly asserted that requirements engineering is the software and information systems development activity that most influences the quality and life-cycle costs (Boehm, 1981; Roman, 1985; Schach, 1993). Experience has shown that for all but the most trivial information systems, it is extremely hard to produce a complete and valid (in the sense of producing the desired effects) set of requirements. This difficulty of fully pre-specifying a system has spawned proposals for process models that rely on a sequence of short specify-develop-evaluate cycles. The aim is to enable the lessons learned from relatively cheap trials with prototypes, or similarly partially-engineered systems, to inform later development. We refer to these models collectively as non-sequential process models, The entire concept of a non-sequential process model relies crucially on an evaluation component to support the exploration of requirements.

This paper starts by considering the fundamental ideas behind three main non-sequential models. It then moves on to discuss what is critical in evaluation activities make them operate in a reliable way. The paper then distinguishes between evaluation and measurement, and throws some light on the philosophical and technical issues that surround evaluation in particular. It is then argued that it is the "interpretive" side of evaluation that matters most to complex information systems: what do the lessons learned from deploying a system increment or a prototype mean for the host setting, and what obligations will the host system have to meet? Lastly, the paper briefly outlines a particular approach, based on the Grounded Theory method from the social sciences, to make a interpretive stance a practical option for information systems evaluation.

Developing Requirements Incrementally

In the development of complex information systems, the specification of requirements includes a functional aspect expressed in terms of the services that the system needs to perform; and a non-functional aspect referring to performance and quality attributes such as response time, reliability, usability and so on. However, it has for some time been recognised (Boar, 1984; Gilb, 1988) that the requirements of substantial information systems cannot be fully articulated prior to implementation effort. Often this position is stated simply as "the users can not know all of their requirements in advance". Other critiques focus on the shortcomings of the traditional “Waterfall” view of the system development process, in which activities are assumed to occur in a strictly sequential manner (see for example McCracken & Jackson, 1982; Pressman, 1994). These insights have led to the emergence of a number of alternative process models that do not assume that a complete or final statement of requirements exists. Examples of these are Prototyping (Carey, 1990), the Evolutionary Model (Gilb, 1988) and the Spiral Model (Boehm, 1986). Each of these models make assumptions about what are the most critical aspects of the problem, e.g. the user interface, the scope of the system, or certain types of risks. All such models presume a form of experimental, limited-scale system engineering activity followed by an evaluation which forms the basis of further design decisions.

We refer to these alternative process models, collectively. as non-sequential process models. These, although they vary in detail depending on the priorities given to system engineering issues, have in common an underlying assumption that it is inherent in the nature of complex information systems that their requirements cannot be fully specified in advance.

Since all the above models rely on initial, exploratory development of the system, the quality of feedback (from the experiments) is rather sensitive to how the first increment (or prototype) of the system is devised. This initial product needs to "excite" the users to engage with it extensively so that the feedback is as rich, and hence as useful, as possible. The aim is to maximise the number of problem areas uncovered early in the development. This suggests that some ordering of requirements is important, to decide which mix of functional and non-functional requirement should be implemented in the first prototype[1]. By the same argument care needs to be given to the structure, scope and content of each subsequent increment or prototype.

The Importance of Evaluation

It is for non-sequential process models that evaluation acquires a particularly critical nature. Poor evaluation simply nullifies the value of engineering the system in an incremental fashion in the first place. We believe that the construction of suitable evaluation frameworks should be included within the scope of the domain analysis, which is already an essential task in requirements engineering.

Aims of evaluation in non-sequential process models include assessment of aspects of quality such as the usability of the human-computer interface and the system's functionality. Although individual process models characterise this differently, all models give attention to assessment of a system's impact on the social and organisational settings into which it is introduced. The subjective nature of many of the qualities of information systems (Iivari, 1988) implies that in-process evaluation frameworks need to be sensitive to the subjective and qualitative aspects of the information system and its stakeholders. The results of evaluation are an essential source of information. They are vital for subsequent iterations of design and build activities. The evaluation findings lead to changes, refinements and shifts in priorities which can affect functional requirements, the system's architecture or aspects of its detailed design. These shifts may constitute fundamental changes to the way the design problem is framed. Experience with the developing system itself will also lead to changes in the criteria which are relevant for successive evaluation activities. It is clear that as far as non-sequential process models are concerned, and their underlying exploratory orientation, the construction of evaluation frameworks, that are systematically derived from and strongly grounded in the context of the system's use should be one of the requirements engineer's chief concerns.

[1] We would like to add here that the same question arises for a COTS approach to systems building. It might not be practical or wise to procure all needed COTS at the beginning, and thus an approach that aims at gradually exploring and prioritising the objectives of the domain is probably more effective.

Where qualitative requirements are taken into account there is too often a lack of rigour in the process. We simply look over the users' shoulders while they experiment, or we collect their (essentially ad-hoc) comments on the system to incorporate into plans for a later increment (again in an ad-hoc way). This all means that the process of gathering, integrating and interpreting the results of experimenting with an increment is too much a matter of chance, and thus prone the leaving out of vital information that might enhance the quality of the system. There is no solid foundation on which to base the inevitable task of making trade-offs between design alternatives. The whole process is non-systematic and highly susceptible to random influence from the systems developers or those in close contact with them.

We cannot simply introduce rigour by focussing on evaluating what can be easily measured. Evaluation is not the same as measurement. Evaluation concerns comprehensiveness, integration and interpretation; whereas the act measurement seeks to describe particular properties, which are characteristically single-dimensional, according to agreed standards and following agreed procedures. Different measures tend not to integrate with one another to convey a multi-dimensional holistic perspective; they are also not directly concerned with merit which is what evaluation seeks to establish (see Scriven, 1991). Therefore, it is not sufficient to quantify particular aspects of software and information systems, but it is vital to carefully consider how meanings, and hence actions, might be attached to various discoveries. This sense of evaluation is particularly important for formative evaluation: that which aims at contributing to the *shaping* of the programme or artefact being evaluated. This is exactly what we seek to achieve when we conduct evaluations of information systems or prototypes.

The central characteristic of the formative, in-process evaluation of putative information systems is that the raw material with which it must deal is primarily qualitative. Qualitative data are the unstructured and non-numeric data, such as interview and observational material. This is the type of data that is encountered first and which predominates in information system development. Even if numeric measurements are to be applied, the decisions as to which categories of data to be observed, and what interpretations are derived from it are essentially qualitative. It is therefore vital that an effective approach to evaluating evolving information systems has to pay close attention to the rigorous handling and analysis of qualitative data.

Having now established the status, nature and role of evaluation that we advocate for supporting incremental or prototyping-based requirements engineering process, we now proceed to give details of the approach that we have devised for dealing with this problem.

A Methodological Response

From the above discussion of the aim of evaluation during requirements engineering and the type of the data that arises, we arrive attwo methodological imperatives:

1. Focus on formative, interpretive evaluation.
2. Attend to qualitative data and methods suitable to its analysis.

We have looked at the social and human sciences, which are rich with methods that operationalise the two aims above, and selected the Grounded Theory method to construct our framework.

In short, Grounded Theory aims at producing theory from data in a rigorous and systematic fashion, hence the term "grounded". Grounding in this case refers to the grounding of any theoretical proposals put forward in the data available: all the data and nothing but the data. The process is fundamentally iterative, with sources of data periodically "sampled" as the evolving theory takes shape and gaps or inconsistencies are uncovered. Particular techniques address the problem of the establishment of "concepts" and "terms" that apply to the phenomenon under study. In fact, even the selected phenomenon itself may be altered as a result of the analysis if, for example, the data suggests that another phenomenon integrates the data in a more coherent way, and better accounts for the various observations that triggered the analysis in the first place.

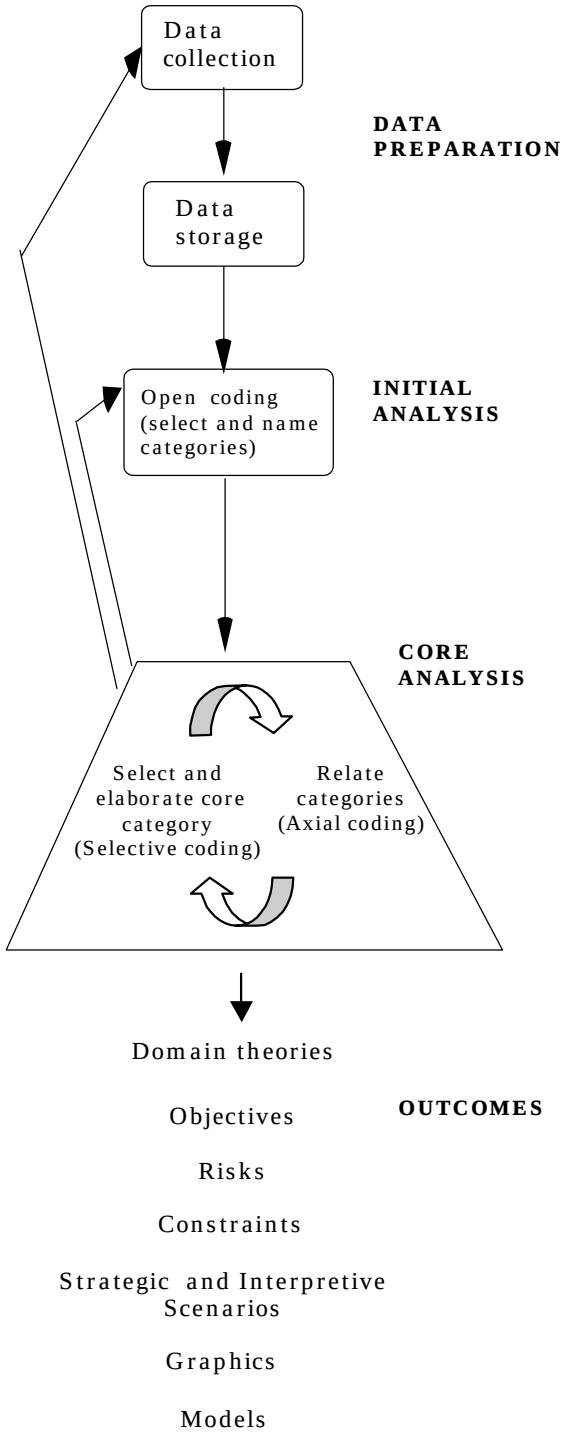

Figure 1: Core steps in Grounded Theory analysis (adapted from Pidgeon et al., 1991)

Figure 1 above summarises a set of steps that operationalise the main concepts of Grounded Theory. For more information on the method, the reader is referred to (Pidgeon et al., 1991; Strauss & Corbin, 1990,1998).

The following steps summarise our approach for effecting formative, qualitative data-oriented evaluation of information systems during their development:

Step 1) Collect and code qualitative data that characterises the "informational need" situation in the domain concerned. (It is also possible to build a series of "fit" constructs related to this need, but we do not expand on this aspect of our work in this paper due to scope limitations.) Typically a number of iterations are entered into until the data categories are stable and as fully developed as possible, meaning that such categories fully fit and account for all the primary data collected. This analysis can be done with respect to a number of contextual levels within which the informational need exists (such as these of *environment*, *organisation*, *domain* and *system* as illustrated in Figure 2 below). This step approximates two types of analyses known as "Selective Coding" and "Axial Coding" in the Grounded Theory literature.

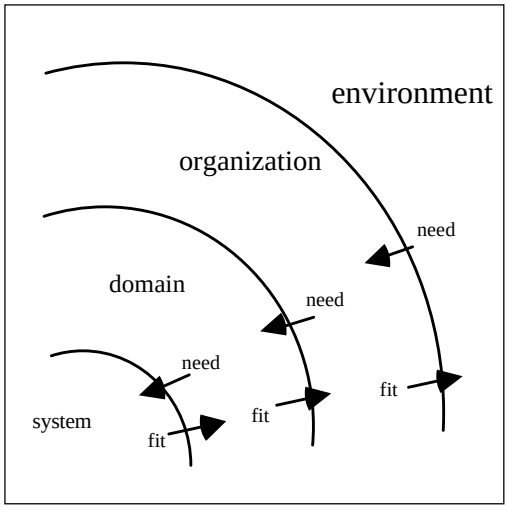

Figure 2: The flow between "need" and "fit" theories at four levels of analysis.

Step 2) Systematically relate the data categories in one or more theoretical formulations that account for the informational need in a process-oriented writing style showing cause, effect, interactional and contextual relationships. This is done with the aid of an organising device known as the "paradigm model", proposed by Strauss and Corbin. The paradigm model is used to integrate the various concepts reached in a causal/ relational form. The paradigm model organises the data categories into the roles of *phenomenon*, *conditions*, *actions and interactions* and *consequences*. After relating concepts in this way, a central category is selected, around which a "story-line" is developed and written. The story-line is in effect a theoretical formulation that attempts to characterise the phenomenon of interest.

[This step is based on the "Selective Coding". The theoretical framework that organises the data is causal, and is based on that devised by the originators of the Grounded Theory method. The causal nature of the paradigm model was found very useful in our work in organising the multifarious concepts gleaned during the analysis. The central category that we tried to explain in each case was related to the "informational need" present in the situation, which is arguably the essence of any requirements engineering activity.]

Step 3) Using the categories coded earlier, and with the aid of the processual accounts (using various instantiations of the paradigm model) made for the phenomenon under investigation, a set of "projected scenarios" are constructed. It is against these scenarios that system increments or prototypes are evaluated. These scenarios are developed in collaboration with the system's clients and users, against the background of qualitative data collected and their analyses.

Step 4) The evaluation is effected by using the projected scenarios as a guide against which an increment of the system is evaluated: both dynamically in terms of the cause-effect relationships it describes, and statically in terms of the concepts that feature in it. The concepts are in effect monitors that are used to assess whether a part of the system is operating, or is benefiting the organisation, as originally envisaged. The results of this evaluation can point to new requirements. These are hard to uncover with less systematic or less comprehensive evaluation.

Example of application

We have applied our framework to a case study involving the development of a document management system for a law practice in London. There were a number of sources for the (mainly qualitative) data collected. The prime source of data was around 10 semi-structured and unstructured interviews conducted with users, other stakeholders of the system, as well as outside legal experts to help in characterising the wider environment. The other sources of data included observational notes made by the researcher while acting in the role of a trainer/ consultant in the document management technology employed. A total of 18 visits to the organisation concerned were made in this role. Given the practical role that the main researcher played in this project, and the organisation's keen interest in getting involved with document management technologies, data collection was relatively easy. The only problem was the occasional unavailability of lawyers for interview or comment on analysis results. (The lawyers involved charged their time to clients in units of 6 minutes, which meant it was easy to quantify potential loss of fee for time allocated to activities related to the project.) This meant that at times it was necessary to remind the clients that their close and timely involvement was essential to the successful deployment of the system.

Below, we show sample instantiations of the steps that we introduced above.

Step 1: We analysed qualitative data at the levels of the immediate *domain*, at which the need for a system had arisen; the *organisation* that contains the domain; and the wider *environment*.

The result of the analysis is a set of concepts and terms that relate to the software or information system in question. Table 1 below shows how the properties of the "Case" category were organised and described.

We must point out here that the procedure is considerably more sophisticated than merely selecting interesting-looking terms or nouns. Grounded Theory procedures prescribe a set of techniques for continuous cross-comparison and refinement of any selected terms. Note also that the loci in the data where various properties arose are indexed in the table inthe column headed "Data location ref.".

Concept: **Cases**

Property	**Dimensional range**	**Value**	**Data location ref.**
Duration	v. long... short	Variable	RS2
Managed by	lawyer type	Client partner	RS13
Assigned to	personnel types	a team (partner, lawyer, secretary)	RS13
Constrained by	procedure types	court rules & procedures, time limits	VM2
Generates	document types	lists, memos, attendance notes	VM11

Table 1: Concept table for "Case"

Step 2: Relate the established concepts in a theory-like formulation, with causal links as appropriate. Each level of analysis will have a "theory-of-need" developed that narratively articulates and links the concepts that bear on the system (the phenomenon) in question. An example of such a theory is given below:

Domain-level need theory

The large and loosely structured volumes of documents {causal condition/context} that a typical law practice {causal condition} has to deal with is such that a significant amount of time and money {intervening condition} is expended on managing the documents involved in any particular case within acceptable performance indicators {intervening condition}. The documents managed have very dynamic (changing) relevance to any particular case and to each other {context}. Document relevance is normally decided by a lawyer. Current personnel have a limited IT experience {context}, which gives rise to the current manual system and its procedures {action/interaction strategies}. This leaves scope for improving the effectiveness and efficiency of the document management process {consequence}.

Note that the underlined terms are those uncovered during analysis, and are typically further described and indexed in the data as we have shown in Table 1.

Step 3: Using both the terms and theories developed through the analysis, and in consultation with the system users and clients, we developed a set of "projected" scenarios against which the system is evaluated. A fragment of such a scenario is given below:

A Projected Scenario

Providing an informational artefact that maintains document descriptions (entered by user) and enables document management and production in an acceptable and usable way

leads to =>

easy and acceptable operation of an alternative (automated) document management system

which leads to =>

more cost effective (faster and cheaper) document management in the domain concerned.

Note that the scenario, by itself, appears extremely simple. However, the important point here is that every term that features in the scenario, and in fact significant parts of the processual aspects of the scenario itself, have been fully grounded in the data collected at the various levels of analysis.

Step 4: Using the scenario as a guide, we were able to conduct a systematic discussion as to the merits of the deployed system increment. The scenario served as a background against which various interpretations were derived, and these informed the clients' decisions about which features. It also supported their appreciation of what changes to the working environment were needed to

make most effective use of the system being introduced. The evaluation occurred in two dimensions: dynamically in terms of the cause-effect relationships that the theories describe, and statically in terms of the concepts that feature in them. The concepts in effect acted as monitors or poles around which evaluation data were collected and integrated.

Our approach to using procedures based on Grounded Theory in evaluating system increments has helped us deal successfully with a huge quantity of qualitative data, to sift through and understand it, and to reach those all too important "interpretations" of the result of the experiment from the point of view of the host setting. Moreover, the responsibilities of the host setting towards the system became much clearer to the parties involved, in terms of what they were and their degree of criticality to the system.

Another, rather surprising, observation concerns the speed and ease with which the system's users and clients understood our formulations about how the organisation is, and how it may be in the future, affected by the type of system they desire. This occurred during a session where users and clients were presented with all the terms that we uncovered on small 3"x5" record cards: one side has the term it describes, the other has its properties as suggested by the data. The set of cards (25 in total), were laid on large flip-chart sheets on which the four levels of analysis (System; Domain; Organisation and Environment) were drawn as onion-skin layers. The annotated flip-chart sheets were used as a white board, but put on a table in a way resembling military field maps. No subsequent sessions were needed to address the same topics. The clients comments on the relevance and accuracy of the formulations presented were solicited and these were found to be both highly accurate (but not totally so) and fully relevant.

Outcomes

In terms of the particular problem under study, the formulations that we obtained, were highly rated by the clients for the way it helped them further understand and conceptualise the relevance of the system to their organisation. It also enabled them to appreciate the significance and magnitude of organisational changes necessary for deploying the proposed document management system. We translate this to real benefits of our approach in planning for business process re-engineering (BPR) efforts, often found necessary – if on a limited scale – for new IT systems. Although all categories of users understood and approved the changes proposed, it was not possible to test the efficacy for these changes in practice. However, the intention was not to provide a perfect or final set of interventions, but to provide the method needed for deriving them, as well as the conceptual background against which they can be juxtaposed and evaluated in a rigorous way.

A somewhat surprising observation was the way in which relatively IT-innocent stakeholders readily assimilated the analysis presented, the evidence being the ease with which they amended some of the concepts presented to them, in terms of property values and location on the model. The scenarios were read as causal relationships between concept cards that were placed on the System, Domain, Organisation and Environment bands of analysis.

The system's architect also commented that the field evaluation and other analysis results correlated significantly with his perception of the system, and that the majority of facilities found to be lacking in the study were scheduled for addition to the system. The system's architect also commented that areas of risks uncovered in the analysis "would have been useful in setting appropriate priorities and targets during development". A list of risks and constraints was one of the products of applying our approach. These produced valuable insights relating to issues that were of concern at some point during the engineering of the technical system.

From the wider system design point of view, the systematic analysis of data about the increment in use enabled us to design "environmental accommodations" that were defensible on the basis of the data. These included particular procedures aimed at overcoming system utilisation problems, such as primary document coding.

From the point of view of the analysis work, the Grounded Theory-based approach helped in organising and analysing a very large volume of diffuse qualitative data. For example, 231 initial concepts were systematically and comprehensively condensed into 35 concepts and their properties. These were used to derive formulations that contributed to characterising, understanding, relating and prioritising requirements. Moreover, by virtue of the way theoretical formulations were reached, a remarkable degree of traceability was achieved. Every single "concept" in the theories expressing requirements, as well as the scenarios was traceable all the way back to the original data and through the intermediate analyses.

The limitations of the approach were that the style of analysis mandated by Grounded Theory was very time consuming. Also, the quality of this type of analysis critically relies on the analyst's skills and experience. The reader is directed to Glaser (1978) for a discussion of the techniques that can be used to develop the requisite skills for Grounded Theory analysis, and to Strauss and Corbin (1998) for a discussion on the types of skills required of a Grounded Theory researcher.

Conclusions

Evaluation is a fundamental and critical activity that needs to be rigorously conducted in any gradual, experiment-based, approach to requirements engineering that aims at maximising what can be learnt from deploying a system increment or a prototype. It is vital that such evaluation makes full use of the soft, qualitative data available in a way that can inform further system development. It is also vital that the value of any such experiment is fully explored through an interpretive approach that formally seeks to establish such meanings and implications through evaluation.

We have found that procedures based on the Grounded Theory method from the social sciences went a long way towards achieving our objectives in supporting incremental and learning-oriented approaches to the engineering of the requirements for complex information, systems. This is achieved by making systematic use of the soft, qualitative data that arise naturally during the process of studying systems in use in the settings for which they are being designed.

References

Boar, B. H. *Application Prototyping, a requirements definition strategy for the 80's* New York: Wiley, 1984.

Boehm, B. *Software Engineering Economics* Englewood Cliffs: Prentice-Hall, 1981.

Boehm, B. "A Spiral Model for Software Development and Enhancement" *ACM SIGSOFT Software Engineering Notes*, 11(4), 14-24, 1986.

Carey, J. M. "Prototyping: Alternative systems development methodology" *Information and Software Technology*, 32(2), 119-126, 1990.

Gilb, T. *Principles of Software Engineering Management* Wokingham: Addison-Wesley, 1988.

Glaser, B. G. *Theoretical Sensitivity* Mill Valley, California: The Sociology Press, 1978.

Guba, E. G., & Lincoln, Y. *Fourth Generation Evaluation* Newbury Park, California: Sage, 1989.

Henerson, M. E., Morris, L. L., & Fitz-Gibbon, C. T. *How to Measure Attitudes* Newbury Park, California: Sage, 1987.

McCracken, D. D., & Jackson, M. A. "Life Cycle Concept Considered Harmful" *ACM SIGSOFT Software Engineering Notes*, 7(2), 29-32, 1982.

Patton, M. Q. *Qualitative Evaluation and Research Methods* (2nd ed.). Newbury Park, CA: Sage Publications, 1990.

Pidgeon, N. F., Turner, B. A., & Blockley, D. I. "The use of Grounded Theory for conceptual analysis in knowledge elicitation" *International Journal of Man-machine Studies*, 35(2), 151-173, 1991.

Pressman, R. G. *Software Engineering - A Practitioner's Approach* (4th ed.). New York: McGraw-Hill, 1997.

Roman, G. "A Taxonomy of Current Issues in Requirements Engineering" in R. H. Thayer & M. Dorfman (Eds.), Computer, April 1985. Reprinted in: *System and Software Requirements Engineering* (pp. 14-22 (Computer)). California: IEEE Computer Society Press, 1985.

Schach, S. R. *Software Engineering* (2nd ed.). Boston: Irwin, 1993.

Scriven, M. *Evaluation Thesaurus* (4th ed.). Newbury Park, California: Sage Publications, 1991.

Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., & Twidale, M. "Integrating ethnography into the requirements engineering process" in S. Fickas & A. Finkelstein (Eds.), *IEEE International Symposium on Requirements Engineering: RE '93*, (pp. 165-173), 1993.

Strauss, A., & Corbin, J. *Basics of Qualitative Research, Grounded Theory Procedures and Techniques*. California: Sage, 1990.

Strauss, A., & Corbin, J. *Basics of Qualitative Research, Techniques and Procedures for Developing Grounded Theory* (2nd ed.) California: Sage, 1998.