Formality and Informality in Requirements Engineering

Joseph A. Goguen*
Department of Computer Science and Engineering
University of California at San Diego, La Jolla CA 92093-0114

Abstract: This paper is an overview of a new approach to requirements; the exposition takes formality and informality as its theme. The approach considers that requirements are information and that information is social. Ethnomethodology and semiotics are used to explore the nature of information and requirements. Some limits of formalization, and the importance of tacit knowledge and evolution motivate new methods for acquiring and tracing requirements.

1 Introduction

Requirements are information, and information is *situated*, i.e., we must consider how it is produced and used, not merely how it is represented. The situations that determine the meaning of requirements are not merely technical, but involve *social* context in significant ways. The social situatedness of information means that to some degree, it is irreducibly *informal*. However, there are some advantages to informality and situatedness for requirements engineering, including greater flexibility and efficiency. This paper is an overview, drawn mainly from [6, 7] and [8].

2 Understanding Requirements

Requirements are properties that a system should have in order to succeed in the environment where it will be used. This definition (from [5]) refers to a system's context of use, and thus to the social as well as the technical. Much of the information needed for requirements is embedded in the social worlds of users and managers. It is informal and depends on context for its interpretation. Moreover, much of it is *tacit*, i.e., cannot be verbalized by those who have it (see Section 2.1). On the other hand, the representations needed for constructing computer-based systems tend to be formal, or at least semi-formal.

Important social issues that complicate requirements include culture, organizational structure, legal and economic constraints, work practices of end users, marketing strategies, and much more. Because such issues cannot be modelled by traditional technical methods, novel approaches are needed.

*On leave from Programming Research Group, Oxford University Computing Lab.

Of course, technical approaches can be highly successful, and are certainly necessary for many problems; e.g., for humans to survive in a hostile environment like space, factors like air, water and heat must be carefully quantified and controlled. However, purely technical approaches cannot take adequate account of the informal social, political and cultural factors that are so often responsible for failures of real systems. Both the formal, context insensitive, and the informal, socially situated aspects of systems are crucial to the success of requirements engineering; [5] suggests that the task of requirements engineering is to reconcile them; see also the discussion in [21].

2.1 Tacit Knowledge

It can be hard to get adequate data on which to base requirements. Experience shows that simply asking managers what they want often works poorly. They do not (usually) know what is technically feasible, and cannot accurately describe what their workers and clients really do, or even what they themselves really do; the same applies to wants and needs.

This is not because managers are incompetent; in fact, they are often real experts at their job. Rather, it is due to what philosophers [18] call the problem of *tacit knowledge*, i.e., the phenomenon that people are able to do things, without being able to say how they do them. Some examples are riding bicycles, speaking languages, negotiating contracts, reconciling personal differences, evaluating employees, and using a word processor. One important reason for this difficulty is the situatedness of the information involved.

But to build a system that effectively meets a real business need, it is usually necessary to find out what workers, clients and managers actully do and want. It is best to go where the work is actually done, and carefully observe what really happens. Various methods from sociology seem promising for this purpose.

2.2 Social Approaches

We distinguish *social* approaches from *cognitive* approaches. The latter focus on a single autonomous rational "user" interacting with a computer and possibly a larger system. Phrases like "human factors," "human-computer interface," "agent," and "user centered" all point to a cognitive approach. Actually, the

"human factors" approach represents an even older style, which takes humans as introducing additional technical factors to be measured and engineered in much the same way as traditional physical factors like bandwidth. (The movie *Apollo 13* dramatizes NASA's human factors approach, showing how it was mitigated by social context, e.g., military traditions like responsibility and chain of command.)

2.3 Ethnomethodology

Traditional sociology has been much influenced by what it takes to be orthodox science, which first formulates a theory, then derives some predictions, and finally tests them. This is done *objectively*, in that the desires and biases of the scientist should not affect the outcome; there is a rigid separation between subject and object, between observer and observed.

Modern physics has moved far from this model of science, so it should not be surprising if sociology, and the social aspects of computing, had to go even further. In particular, if objective information is replaced by situated information, then orthodox techniques for formulating and testing hypotheses, e.g., statistical sampling, are not valid, because observed events cannot be assumed drawn from the same sample space. However, statistical methods underpin much of sociology, e.g., the design and evaluation of questionnaires. This does not mean statistics and questionnaires are useless, but rather that they should be used with caution where context plays a significant role.

Ethnomethodology can be seen as a reaction against the "scientific" approach of traditional sociology. Ethnomethodology reconciles a radical empiricism with the situatedness of social data, by looking closely at how members of a group actually organize their behavior. A basic principle is that members are held *accountable* for certain actions by their group; moreover, exactly those actions are significant for that group. Members performing such actions can be asked for an *account*, i.e., a justification[1]. We call this the *principle of accountability*.

From this, it follows that social interaction is *orderly*, i.e., can be understood. Since the participants understand it because of accountability, analysts should also be able to understand it, if they can discover the methods and categories that members themselves use to make sense of their interactions. For this purpose, it is important to use "naturally occurring" data, collected in a situation where members are engaged in activities that they regularly and ordinarily do; otherwise, the principle of accountability will not apply, and we cannot be sure that events in the data have any natural social significance. For example, data collected in interviews cannot be used (unless we want to study what happens in interviews!).

Ethnomethodology tries to determine the *categories* and *methods* that members use to render their actions intelligible to one another; this contrasts with presupposing that the categories and methods of the analyst are necessarily superior to those of members. The methods and categories of members are identifiable through the ways that members are held accountable by their group. Through immersion in data from some particular social group (such as stock brokers), particular competencies are gradually acquired that let an analyst be a sensitive, effective "measuring instrument" in that domain. In this way, subjectivity is harnessed rather than rejected[2].

We conclude this subsection with some aspects of ethnomethodology that may be considered negative. The use of naturally occurring data precludes many convenient "quick and dirty" ways to collect information, including questionnaires and interviews. Also, controlled experiments are unsuitable, as are solitary equipment operators. Since data must be grounded in the concrete circumstances of its production, ethnomethodology cannot be (directly) applied to systems that have not yet been built.

Analysts must understand members' concepts and methods. For some particular purpose, it is only necessary to understand certain members' concepts and methods, to a certain degree; but it can be hard to determine what needs to be understood, and to what degree. Ethnography can help by providing an initial understanding of the group being studied. Ethnomethodological analysis is labour intensive; projects may involve hundreds of hours for recording, transcribing and analysing data.

Ethnomethodology can be hard to understand; relatively comprehensible expositions of some important points appear in [12], [22], and [5]. Conversation analysis studies details of overlap, response, interruption, repair, etc. in ordinary conversation [19], while interaction analysis uses video data. Projects at Oxford have used ethnomethodology to better understand requirements in various domains, including stock brokerage and telecom operations; see Section 4.2.

[1] This does not mean that such accounts are always, or even usually, requested by members of the group, or that they are necessarily given when requested.

[2] It is worth noting that ethnomethodological analysis is often done in groups, creating a social situation in which members are accountable for their accounts.

2.4 Situatedness

We can now be more precise about what *situatedness* means: events can only be understood in relation to the concrete situation in which they occur; they acquire meaning through interpretation in that situation. The following *qualities of situatedness* (from [5], partly following Suchman [22]) may help clarify this:

1. *Emergent:* Events cannot be understood at the level of the individual, that is, in terms of individual (cognitive) psychology, because they are jointly constructed by members of some group through their on-going interactions.
2. *Local:* Events are seen as such in a particular context, including a particular time and place.
3. *Contingent:* The construction and interpretation of events depends upon the current situation, potentially including the current interpretation of prior events. Interpretations are subject to *negotiation*, and relevant rules are interpreted locally, and can even be modified locally.
4. *Embodied:* Events are linked to particular physical contexts, and the particular way that (human) bodies are embedded in a context may be essential for interpreting some events.
5. *Open:* Accounts of events cannot in general be given a final and complete form, but must remain open to revision in the light of further analyses and further events.
6. *Vague:* Practical information is only elaborated to the degree that it is useful to do so; the rest is left grounded in tacit knowledge.

These qualities give rise to basic limitations of information, and hence of requirements.

2.5 Sociology of Science

There have been exciting new developments in the sociology of science. One important voice is Bruno Latour [11]. Latour calls a representation that can be interpreted in essentially the same way in a variety of contexts an *immutable mobile*. This is information that has been (at least partially) dried out; often it is what Latour calls a "*re-representation*," i.e., information that has undergone concentration; for example, a large set of observations of planetary motion might be summarized in a single equation. Latour claims that these three qualities, *immutability*, *mobility*, and *concentration*, are characteristic of scientific information.

Formalization tends to increase these qualities. Indeed, they may be considered as criteria for the success of formalization. As Latour [11] points out, the construction of immutable mobiles can be a way to achieve control. For example, if an analyst compresses large amounts of information into simple graphical representations, then anyone who wishes to disagree must mobilize the resources to re-represent comparable amounts of information. The presentation of dataflow diagrams in requirements meetings illustrates this, since the use of such diagrams is beyond most users and managers, due to the huge volume of information involved in large projects. Several tools have been developed to help organize requirements. However, most are based on naive models that take little account of social context (see [17] for a review). For example, the IBIS model records decisions in a network of issues, positions and arguments; however, this is not a natural category system for its users.

2.6 Requirements Evolution

Constantly changing requirements are a major difficulty in building large complex systems; let us call this *requirements evolution*. Tracing design decisions, specs, and code back to the requirements they are supposed to meet involves maintaining a complex network of links in the face of constant change. Real development projects for large complex systems rarely attempt this, and those that do find it excessively burdensome, because the current state of practice requires manual entry and update of all dependencies. Section 5.3 discusses an approach to traceability motivated by the observations in this paper.

3 What is Information?

We speak of "information technology," "information systems," "information science," "information engineering," and even "the information age," but it is an open scandal that we do not have any theory, or even definition, for information that is adequate for familiar applications in business and science.

To understand information, we must broaden our perspective beyond specific technical factors. In contrast to statistical theories of information following Shannon [20], and representational theories of information like the situation theory of Barwise and Perry [1], we suggest a *social theory of information* [6]. No theory of the kind we need can be "objective" or "realist," in the sense of assuming a pre-given distinction between subject and object, or an objectively given real world. Thus, traditional semiotics is not adequate, because it assumes that signs represent things in a real, objective world; we need a social semiotics.

3.1 Towards a Social Information Theory

It seems very difficult to develop a theory of information that can provide an adequate foundation for requirements engineering, and the material in this section should be considered a preliminary exploration. In that spirit, I suggest that an *item of information* is

an interpretation of a configuration of signs for which a social group can be held accountable[3]. Here, each item of information is tied to a particular relation of accountability for a particular interpretation in a particular group; thus, a given configuration of signs could be interpreted in different ways, giving rise to different items of information. It takes *work* to interpret signs to make them information, and this work is done in some particular context, making use of the resources and constraints of that context.

In classical semiotics, signs are configurations that do not necessarily have significance. However, the very notion of sign as configuration already presupposes a category system (e.g., a certain character set), so that signs must have at least the significance of belonging to this system. Hence the notion of "sign" in the above definition is that of a category in what Section 2.3 calls a category system, rather than part of a pre-given system of objective distinctions.

We can distinguish information that can be understood in a wide variety of contexts from information that is so thoroughly situated that it cannot be understood except in relation to certain very particular contexts. We call these types of information *dry* and *wet*, respectively [4, 5]. There is really a continuum of "humidity" for information, i.e., there is "damp" information, of which cookery recipes seem a typical example. A fairly extreme case is the "raw data" collected in a scientific experiment; although it may be just a collection of numbers, it is very highly situated, because those numbers only make sense to a small group who share a very particular context. On the other hand, an equation summarizing that data is relatively more dry, and a general physical law is even drier. This suggests that formalization is a key technique for making information drier, i.e., less mutable, more mobile, and more concentrated.

Information, however dry, must still be interpreted in some local context. Dry information is distinguished by the possibility of its being interpreted in what counts as the same way for practical purposes in a useful variety of contexts. Latour [11] discusses the example of cartographic maps: given the proper instruments and proper conditions (e.g., good weather), such maps can be used anywhere in the world; but each such use is still a local interpretation. Information is always situated in some particular social context: there is no such thing as abstract, ideal information, which is independent of its context. We can relate this to the "local language games" of Lyotard [15], which were inspired by late work of Wittgenstein.

This approach to information is very different from that of knowledge representation in artificial intelligence, which is objectivist and realist, and takes little or no account of context, including how information is acquired, used, and evolved.

3.2 Implications for Requirements

Many difficulties with developing large software systems arise from how we handle requirements, taking insufficient account of context, particularly social context. The prevailing belief that information is (or should be) totally dry may be a major obstacle to developing and accepting better methods. The discussions above suggest there may be viable alternatives, though more work is certainly needed.

It can be argued that truly adequate requirements for a system can only be determined after the system is actually being used successfully. This is another facet of the situatedness of information; members revise their accounts in the light of new events, or of new interpretations for prior events, and even what counts as an event is negotiable. Support for this can be found in empical work on plans and explanations [13, 9, 22]. More radically, it could be argued that time, in the sense of a linear ordering imposed upon events, is itself the result of the retrospective reconstruction of causal chains to explain events, i.e., to account for them in relation to shared values.

4 Requirements Acquisition

This section discusses some methods for getting data and determining requirements, following [7].

4.1 Some Methods and their Limitations

Perhaps the most common method for getting information about users' needs and habits is *introspection*. Although this can be useful, the introspection of an expert in a different field, such as computer science, may not reflect the experience of the intended users. Experts tend to use what they remember or imagine of themselves; for user interface design, this can be far from the assumptions, questions and fears of actual users. For example, if a word processer unexpectedly centers some text, a user may not try to understand why; many users seem to believe computers just *are* sometimes puzzling or irritating, and that it is neither necessary nor valuable to explain their more bizarre behavior. Cognitive scientists tend to be surprised at this, because their rationalistic theories suggest that a user who finds a bug in a model should correct it.

It is difficult to introspect what work settings look like, or the conditions under which a new technology will be learned or used. For example, many users learn

[3] This definition is intended to be useful in practice, rather than a metaphysical assertion about the nature of information. The sense of accountability intended is that of Section 2.3.

and use technology in conditions that require multiple and ongoing splitting of attention, e.g., due to collaborative relationships.

Questionnaires, whether administered in writing or an interview, are limited by their simplistic model of interaction, which assumes that a given question has the same meaning to all subjects. This excludes interactions that could be used to establish shared meaning. Although open ended interviews are less constrained, they are still limited by the need for participants to share basic concepts and methods, so they can negotiate shared meanings. Open ended interviews are also more vulnerable to interviewer bias. Similar limitations apply to focus groups and their cousins, JAD (or RAD) groups. These methods are also vulnerable to political manipulation by participants, as many requirements engineers know from bitter experience.

Protocol analysis asks a subject to engage in some task and concurrently talk aloud, explaining his/her thought process. Protocol analysis is also used to reflect on tasks retrospectively, i.e., after they have been accomplished. Proponents claim this can be considered a "direct verbalization of specific cognitive processes" ([2], p. 16). They claim protocols are traces of "autonomous cognitive activity". However, language is intrinsically social, created for a conversational partner (this property is called *recipient design* in conversation analysis). When an experimenter asks a person to solve a problem and talk aloud, that person imagines a partner with certain desires, and tries to address those desires (subjects may be rebellious as well as cooperative). Thus, protocols are an unnatural discourse form, based on an incorrect cognitivist model that ignores social context. This unnaturalness can be shown by specific linguistic features [7].

None of these methods can elicit tacit knowledge. The principles of ethnomethodology provide a powerful framework for a deeper consideration of the limitations of traditional methods, as well as a basis for methods that do not have the same limitations.

4.2 Video-Based Elicitation

The Video-Based Requirements Elicitation project at Oxford University is exploring techniques to reveal tacit, interactional work practices that are invisible to standard requirements methods. The following are some goals of this project:

1. To develop an effective new requirements method that can be used by ordinary requirements engineers in actual projects.
2. To identify informal practices in the workplace that must be supported by some new or updated computer-based system.
3. To ease the introduction of new systems by understanding where disruptions might and might not be tolerable.
4. To help manage user expectations by determining where users might want a new system to give a better service than the old one, through analysis of current work practices.

In this project, video recordings of actual work are analyzed using principles from ethnomethodology and other areas to better understand interactional practices in the workplace [10, 14].

4.3 Combining Methods and Zooming

Despite their limitations, I do not suggest that any method is useless (except possibly protocol analysis). In fact, their strengths seem complementary, so that various combinations could be useful. In particular, it is helpful to start with an ethnographic study to uncover basic concept systems and methods used by members, typical patterns of work, etc. After this, one might use questionnaires or interviews to explore what problems members see as most important, how members apply various classification schemes, etc. Then one might apply discourse, conversation or interaction analysis for a deeper understanding of selected issues. This will likely overthrow aspects of prior analyses.

Discourse analysis of stories can be used to explore the value system of an organization, and discourse analysis of explanations can provide a kind of situated task analysis [4]. Conversation and interaction analyses can help overcome limitations of other methods. Interaction analysis reveals details of non-verbal interaction in real work environments, but the effort required to produce video transcripts suggests that it should be used very selectively. Ethnography should be used continually to provide context.

Thus, I suggest a "*zooming*" method for requirements elicitation [5], whereby more expensive but detailed methods are used selectively for problems found by other methods to be especially important. The various methods based on ethnomethodology can be seen as analoguous to an electron microscope: they provide an instrument that is very accurate and powerful, but also expensive, requiring careful preparation to ensure that the right thing is examined. One should not use an electron microscope without first determining where to focus it as exactly as possible, using first the naked eye, a magnifying glass, an ordinary microscope, etc. Similarly, in developing requirements, one should use ethnography, and perhaps discourse analysis, interviews, or questionnaires, before using conversation or interaction analyses.

5 Formalization

According to *Webster's Dictionary*, "*formal*" means definite, orderly, and methodical; it does not necessarily entail logic or proofs. Everything that computers do is formal in the sense that syntactic structures are manipulated according to definite rules. Formal notations and methods are syntactic in essence but semantic in purpose.

The prototypical formal notation is first order logic, which encodes the semantics of first order model theory with formal rules of deduction that are provably sound and complete. Unfortunately, theorem provers for first order logic can be difficult to work with. Formal notations can also capture higher levels of meaning, e.g., they can express security requirements, but such notations are even more difficult.

The orderliness of social life (due to accountability) and the Henley Regatta example in [8] suggest that social interaction might be formalizable; but there are limits to how successful any such formalization can be. In particular, it will not be easy to formalize domains where there are many *ad hoc* special cases, or where much of the knowledge is tacit.

Formalization will be more successful on narrow and orderly domains, such as sporting events, that have long traditions, rule books, referees, regulating bodies, etc. For example, it would be more difficult to formalize a children's game than a boat race, and much more difficult still to formalize human political behavior. There are degrees of formalization, from dry to wet, and it can be important not to formalize beyond the appropriate degree. Cooking recipes are an interesting example, showing how an intermediate degree of formalization is possible and helpful, whereas a very formal treatment would be unhelpful, if it were even possible. This also applies to typical uses of dataflow diagrams and other semi-formal notations in requirements engineering.

5.1 Limits of Formalization

Because any use of a formalism is situated, the qualities of situatedness impose basic limits: any formalization will necessarily be emergent, contingent, local, open, and vague. However, formalization does tend to reduce these qualities. This implies that without human intervention, a formalization may well be inadequate for its intended application; this is illustrated by common incidents like computer systems sending checks or bills for a zero amount. For a discussion of limits and problems with formal methods, see [8].

5.2 Advantages of Informality

Advocates of formal methods often say that formalization eliminates the vagueness and ambiguity of natural language. But our discussions above show this is not really so. Even the most formal notation still requires context for its interpretation: it must be learned, and then used, before it can have meaning. But more importantly, informality has some real advantages.

We first consider what philosophers call the *efficiency* of language, which is its ability to mean different things in different contexts. For example, if the word "help" is shouted by a swimmer flailing his arms wildly, it means something very different from its appearance in a menu on a computer screen. Moreover, further refining context can yield still different meanings, e.g., if the swimmer is an actor in a movie, or the menu item summons police to a bank. We can play such games indenfinitely, changing meanings, perhaps drastically, by further refining contexts. Moreover, we can create many many other contexts; language seems infinitely plastic. Note that the actual physical situation in which words are spoken is also part of the context that gives them meaning.

Without ambiguity, it would be much more difficult to communicate, perhaps impossible, since so many new words would be needed to make up for all the contexts that might arise. In fact, language *is* efficient, precisely because of its sensitivity to context. Thus, the ambiguity of natural language is essential to its practical use.

Vagueness also contributes to efficiency, but at a finer level of detail than ambiguity. Words like "tall" and "there" do not have precise meanings (such as "seven feet"), so that (for example) "the tall one" can refer to different things in different contexts, and those things can have radically different heights. If "tall" could only mean "seven feet in height," it would not be very useful.

These considerations apply directly to requirements. Context is certainly needed to resolve ambiguity and vagueness. Moreover, requirements documents are often deliberately written to be ambiguous, to hide deep political disagreements; sometimes they are even written to be deliberately misleading. Vagueness and ambiguity also help in stating tradeoffs that cannot be resolved until a later stage of development.

5.3 Tracing Requirements

The path to an RSD (Requirements Specification Document) and beyond can be highly non-linear, involving multiple hypotheses, false starts, experiments, dead ends, etc. Traceability is crucial for survival in such a dynamic environment, but is seldom straightforward, as it involves not just objects and relations to be traced, but also their context, from the motivation for the trace to the total system development con-

text. The TOOR system [17] was designed for defining, instantiating, updating and tracing hypermedia artefacts and relations in several modes, through an intuitive template-driven graphical interface. TOOR's object orientation supports user-definable classes and inheritance for requirements objects and relations, and provides a database with automated checks and annotations. Its design has been deeply influenced by the social considerations presented in this paper, including the following four points:

1. Requirements are situated, resulting from negotiations whose outcome depends on the interests, organizational position, technical background, etc. of participants. Moreover, the relations among requirements and other objects also depend on context and result from negotiation. Hence, a trace system should allow great flexibility in defining and updating artifacts and relations. TOOR uses the powerful modules of parameterized programming [3] to represent context in various ways, and to allow users to define and update relations; this supports any category systems that users may wish to employ. Modularity can also improve quality and reduce cost through reuse.

2. Requirements are an inextricable part of the development process. Tracing project artefacts forward and backward from requirements is useful throughout the lifecycle. This implies that registering requirements and relations among them and other project artefacts is not just a documentation chore, and tracing requirements (and other objects) is not just a management activity. Instead, important requirements issues arise throughout the lifecycle, and appropriate tool support can make it much easier to resolve such issues. This requires integrating requirements information with information about analysis, specification, coding, etc.

3. Requirements evolve throughout a project's life. An RSD is a set of requirements agreed at a specific time, but this does not freeze requirements; as development proceeds, new ideas, design diagrams, specifications, code, etc. are produced, generating new objects and relations, and inevitably modifying requirements. Requirements change in content and form, becoming more consistent, accurate and clear; new attributes are added and old ones deleted. Requirements relate to one another and to other artefacts, and these relations also change in content and in form.

TOOR deals with evolution in a uniform way: classes are declared for each kind of artifact and relation we wish to control, and are instantiated as development proceeds. The framework of declarations also evolves. In particular:

- New classes can be added for new kinds of item.
- New attributes can be added to existing classes for new kinds of property.
- Classes and attributes can be deleted.
- Axioms can be changed to reflect a new view of relations and their composition.

And of course existing objects can be updated.

4. Tracing is situated, and thus should produce results that are meaningful in the given situation. In a system with many thousands of interrelated objects, getting all information available will not be useful, since the configurations before and after tracing will be nearly the same. Thus, selective tracing is necessary. However, the selection criteria may not be clear. A user can often get insight by browsing, guided by experience and intuition; in TOOR, this can be constrained in various helpful ways. In other cases, the user may be able to make very precise selections using regular expressions. Thus, TOOR supports several different trace modes.

6 Summary and Conclusions

This paper takes a broad view of the role of requirements in system development. The construction, interpretation and updating of requirements is situated, and this gives rise to some fundamental limits. The facts that evolution is inevitable and unending, and that much of the pressure for change comes from social context imply that strong support for tracing requirements is needed.

Formalization plays a more basic role in requirements engineering than in other engineering disciplines. Since requirements capture *is* a process of formalization, software development requires the construction of new models for each application, in addition to updating already established models.

All this can be summarized by saying that more emphasis should be placed on *context* in system development. Context includes being embedded in the world, which can make information directly available so that detailed models are unnecessary. Another way to say this is that information arises *co-dependently* between humans and contexts, as noted long ago by Peirce [16].

Rising to a still higher level, we might say that beauty, or perhaps better authenticity, comes from relating properly to context. This includes choosing the proper level of formality. Inappropriate formality can be alienating, while properly contextualized formality can be inspiring.

Acknowledgements

I thank Prof. Jawed Siddiqi for his valuable comments, and the coauthors of the papers from which

this overview was drawn. This research was supported in part by British Telecommunications plc, the CEC under ESPRIT-2 Working Groups 6071 and 6112, and a contract managed by IPA in the "New Models for Software Architectures" program sponsored by NEDO (Japan).

References

[1] Jon Barwise and John Perry. *Situations and Attitudes*. MIT, 1983.

[2] K. Anders Ericsson and Herbert Simon. *Protocol Analysis: Verbal Reports as Data*. MIT, 1984.

[3] Joseph Goguen. Principles of parameterized programming. In Biggerstaff and Perlis, editors, *Software Reusability, Volume I: Concepts and Models*, 159–225. Addison Wesley, 1989.

[4] Joseph Goguen. The dry and the wet. In Falkenberg, Rolland and El-Sayed, editors, *Information Systems Concepts*, 1–17. Elsevier North-Holland, 1992. IFIP WG 8.1 (Alexandria, Egypt).

[5] Joseph Goguen. Requirements engineering as the reconciliation of social and technical issues. In Jirotka and Goguen, editors, *Requirements Engineering: Social and Technical Issues*, 165–200. Academic, 1994.

[6] Joseph Goguen. Towards a social theory of information. In Star, Bowker and Turner, editors, *Social Science Researsch, Technical Systems and Cooperative Work*. Addison-Wesley,to appear 1996.

[7] Joseph Goguen and Charlotte Linde. Techniques for requirements elicitation. In Fickas and Finkelstein, editors, *Requirements Engineering '93*, 152–164. IEEE, 1993.

[8] Joseph Goguen and Luqi. Formal methods and social context in software development. In Mosses, Nielsen and Schwartzbach, editors, *TAPSOFT 95*, 62–81. Springer, 1995. LNCS 915.

[9] Joseph Goguen, James Weiner and Charlotte Linde. Reasoning and natural explanation. *International Journal of Man-Machine Studies*, 19: 521–559, 1983.

[10] Christian Heath, Marina Jirotka, Paul Luff and Jon Hindmarsh. Unpacking collaboration: the interactional organisation of trading in a city dealing room. In *European Conference on Computer Supported Cooperative Work '93*. IEEE, 1993.

[11] Bruno Latour. *Science in Action*. Open University, 1987.

[12] Steven Levinson. *Pragmatics*. Cambridge, 1983.

[13] Charlotte Linde and Joseph Goguen. Structure of planning discourse. *Journal of Social and Biological Structures*, 1: 219–251, 1978.

[14] Paul Luff, Marina Jirotka, Christian Heath and David Greatbatch. Tasks and social interaction: the relevance of naturalistic analyses of conduct for requirements engineering. In Fickas and Finkelstein, editors, *Requirements Engineering '93*, 187–190. IEEE, 1993.

[15] Jean-François Lyotard. *The Postmodern Condition: a Report on Knowledge*. Manchester, 1984.

[16] Charles Saunders Peirce. *Collected Papers*. Harvard, 1965.

[17] Francisco Pinheiro and Joseph Goguen. Design and use of an object-oriented tool for tracing requirements. *IEEE Software*, pages 52–64, March 1996.

[18] Michael Polanyi. *The Tacit Dimension*. Routledge and Kegan Paul, 1967.

[19] Harvey Sacks, Emanuel Schegloff and Gail Jefferson. A simplest systematics of the organization of turn-taking in conversation. *Language*, 504: 696–735, 1974.

[20] Claude Shannon and Warren Weaver. *Mathematical Theory of Communication*. Illinois, 1964.

[21] Jawed Siddiqi. Challenging universal truths of requirements engineering. *IEEE Software*, March: 18–19, 1994.

[22] Lucy Suchman. *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge, 1987.