Interner Bericht

A Taxonomy for Combining Software Engineering (SE) & Human-Computer Interaction (HCI) Measurement Approaches: Towards a Common Framework

Jenny Preece
H. Dieter Rombach

251/94

Fachbereich Informatik

Universität Kaiserslautern • Postfach 3049 • D-67653 Kaiserslautern

A Taxonomy for Combining Software Engineering (SE) & Human-Computer Interaction (HCI) Measurement Approaches: Towards a Common Framework

Jenny Preece
H. Dieter Rombach

251/94

*
Jenny Preece
School of Computing
Information Systems and Mathematics
South Bank University
London SE1 0AA
England, UK

Herausgeber: AG Software Engineering
Leiter: Prof. Dr. H. Dieter Rombach

Kaiserslautern, September 1994

This work is submitted for publication to "International Journal for Man Machine Studies"

A Taxonomy for Combining Software Engineering (SE) & Human-Computer Interaction (HCI) Measurement Approaches: Towards a Common Framework

Jenny Preece and H. Dieter Rombach*

School of Computing, Information Systems and Mathematics, South Bank University, London SE1 0AA, England, UK.

***Fachbereich Informatik, Universitaet Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern, Germany.**

Abstract

The rapid development of any field of knowledge brings with it unavoidable fragmentation and proliferation of new disciplines. The development of computer science is no exception. Software engineering (SE) and human-computer interaction (HCI) are both relatively new disciplines of computer science. Furthermore, as both names suggest, they each have strong connections with other subjects. SE is concerned with methods and tools for general software development based on engineering principles. This discipline has its roots not only in computer science but also in a number of traditional engineering disciplines. HCI is concerned with methods and tools for the development of human-computer interfaces, assessing the usability of computer systems and with broader issues about how people interact with computers. It is based on theories about how humans process information and interact with computers, other objects and other people in the organizational and social contexts in which computers are used. HCI draws on knowledge and skills from psychology, anthropology and sociology in addition to computer science.

Both disciplines need ways of measuring how well their products and development processes fulfil their intended requirements. Traditionally SE has been concerned with 'how software is constructed' and HCI with 'how people use software'. Given the different histories of the disciplines and their different objectives, it is not surprising that they take different approaches to measurement. Thus, each has its own distinct 'measurement culture.'

In this paper we analyse the differences and the commonalties of the two cultures by examining the measurement approaches used by each. We then argue the need for a common measurement taxonomy and framework, which is derived from our analyses of the two disciplines. Next we demonstrate the usefulness of the taxonomy and framework via specific example studies drawn from our own work and that of others and show that, in fact, the two disciplines have many important similarities as well as differences and that there is some evidence to suggest that they are growing closer. Finally, we discuss the role of the taxonomy as a framework to support: reuse, planning future studies, guiding practice and facilitating communication between the two disciplines.

Key words

Measurement, evaluation, software engineering, human-computer interaction, taxonomy, framework

I Introduction

In software engineering (SE) the focus is on understanding, controlling, managing and improving software products and processes based on engineering principles [1, 2, 3, 9]. In human-computer interaction (HCI) the focus is on understanding the usability of software products, deriving criteria for 'good' human-computer interfaces, and

devising methods and tools for designing and implementing such interfaces [4, 5, 10, 44, 47, 48].

The historical development of the measurement practices used in the two disciplines has different origins with different perspectives. The need to assess and predict the performance of software *per se* was recognized much earlier than the importance of its usability was acknowledged. Consequently, software engineering measurement techniques have been in existence longer and the discipline is more advanced than usability evaluation. Quantitative measurement tends to predominate in SE, whereas qualitative techniques are common in HCI as well as quantitative ones. For example, ethnomethodology [54], which is a descriptive technique, has grown in importance in the last few years. The data collected in ethnographic studies is from real situations rather than from an artificial context such as a laboratory. It is, therefore, richly contextualized and the aim, when interpreting it, is to understand the actions that occurred within that context and not to look for quantifiable causal explanations as in most other data analysis techniques, such as those used in usability engineering [10].

It is becoming recognised increasingly that a common goal for all the disciplines of computer science is ultimately to improve the quality of software and its cost-effective development. In many respects, therefore, it is not surprising that, despite their different origins there are many similarities as well as differences in the approaches and techniques used in SE and HCI measurement. Often these similarities are over-looked as they tend to be more subtle than some of the obvious differences.

In the remaining part of this paper we describe the aims of this work, which are concerned with defining a taxonomy that can be used as a measurement framework by both fields (section II). We do this by defining a number of 'meta-dimensions' based on Basili's approach [11] and then refining these meta-dimensions further into a hierarchical 'top-down' taxonomy, which characterizes both SE and HCI studies (section III). We then use this taxonomy to describe a number of study approaches (section IV) and to characterize examples of individual studies from our own work and that of others (section V) in order to show how the taxonomy may be applied. We do not, however, claim to be able to describe every conceivable example. In section VI we discuss how the taxonomy provides a framework which can be used to: support long term reuse of knowledge gained from doing measurement studies; guide research planning and practice; and facilitate communication between the two fields by providing a common language and framework for measurement. Finally, we draw the conclusion that SE and HCI are coming together and that, although there are differences between SE and HCI measurement practices, they may be fewer than is often thought (section VII).

In order to reduce the number of different, and sometimes cumbersome, phrases for describing the various types of measurement techniques used in the two disciplines, we shall use the term 'measurement' to refer to all forms of quantitative and qualitative measurement and evaluation done in SE and HCI. The individual terms will be used only with reference to specific examples.

II The aim

The aim of the taxonomy is to provide a framework to describe the approaches and techniques used in current SE and HCI measurement. The taxonomy will be used as a vehicle to explain and make explicit aspects of both SE and HCI measurement which are frequently left ill-defined, if not confused. Such a taxonomy will not only be valuable for examining past research in both fields (i.e., reuse) but also for guiding research practice and the choice of approaches, methods and techniques for future research (i.e., planning). It will also facilitate communication between the two disciplines by providing a common vocabulary and measurement framework. In particular, it should serve as a vehicle for encouraging joint projects in which SE and HCI specialists work together.

III The Taxonomy

Various frameworks and taxonomies have been proposed for analysing complex processes such as usability evaluation methods [6, 7], paradigms of interface development [8], approaches to software quality measurement [9, 13, 14, 35, 52, 53, 55], but all these frameworks are based on the specific assumptions of the disciplines. As yet there is no commonly accepted inter-disciplinary framework which takes account of the wide range of criteria and practices relevant to both studies of SE and HCI. This is because 'measurement' has been viewed from different perspectives and has developed from different origins in the two fields.

Basili's group at the University of Maryland have developed a very broad notion of experimentation based on explicit measurement goals which we are adopting as the basis of our framework [3, 9, 11, 35, 55]. The cornerstones of Basili's approach are the Quality Improvement Paradigm (QIP) [55] and the Goal/Question/Metric (GQM) [11] approach. The former defines software development as an experimental discipline based on the scientific method. The latter formulates the definition of explicit measurement goals and their refinement into measurements (i.e., metrics), which we explain in more detail later in this section.

More specifically, the Quality Improvement Paradigm (QIP) proposes six steps for each software project [55]:

- Characterize: understand the environment based upon the available models, data intuition, etc. Establish baselines with the existing business processes in the organization and characterize their criticality.

- Set goals: on the basis of the initial characterization and of the capabilities that have a strategic relevance to the organization, set quantifiable goals for successful project and organization performance and improvement. The baselines provided by the 'characterization step', described above, are used to define reasonable expectations.

- Choose process: on the basis of the characterization of the environment and of the goals that have been set, choose the appropriate process for improvement and supporting methods and tools, making sure that they are consistent with the goals that have been set.

- Execute: perform the processes constructing the products and providing project feedback based upon the data on goal achievement that are being collected.

- Analyze: at the end of each specific project, analyze the data and the information gathered to evaluate the current practices, determine problems, record findings and make recommendations for future project improvements.

- Package: consolidate the experience gained in the form of new, or updated and refined models and other forms of structured knowledge gained from this and prior projects and store it in an experience base so it is available for future projects.

This list indicates that QIP has two important goals. One is to provide project control and feedback for the project being studied. The second is to improve long-term understanding of software development, and particularly measurement practices, so that knowledge is accumulated from project to project in a way that enables future projects to benefit from the experience and findings of previous projects. Within a single company this can be viewed as long-term corporate learning. The importance of viewing software development knowledge and experience as on-going is also a trend in HCI [48]; particularly in large companies where product life cycles stretch over many versions and families of products.

These underlying aims of improving both current projects and future projects together with its explicit six stage process outlined above and explained in detail by Basili et al. [55] have provided a basis for developing our taxonomy. By modifying the steps from the QIP, adding new information from our own work and synthesizing knowledge from both SE and HCI we have developed a taxonomy which can be used to characterize any study, by specifying:

(i) the goal of the study (what is being looked at and why?)

(ii) the plan of the study (what is the underlying philosophy, how much and what kind of external influence is brought to the study and what is the location and design of the study?)

(iii) the study methods employed (who does the study, what do they do and when do they do it?), and

(iv) the kind of techniques that are used (how is data being collected, analyzed and validated, and how is the information derived from it being communicated both back to the project itself and reused to inform future projects ?)

Each of these dimensions can be further sub-divided and may itself be regarded as a dimension which further refines to more categories. Thus, the structure is heirarchical and can be thought of as a recursive tree. In the remainder of this section we describe the structure of each of the four meta-dimensions in terms of its dimensions and provide examples for each. Table 1 provides an overview of the taxonomy and illustrates the relationships discussed in the paper. All the possible example categories may not have been identified at these lower levels. Furthermore, it is clear that real studies fall into several categories. For example, many studies are done partly in the laboratory, partly in the field and both quantitative and qualitative data are collected using a number of different techniques. In later sections of the paper we discuss the implications of our taxonomy in relation to these mixed studies.

Table 1 Overview of the main dimensions of the taxonomy

1 STUDY GOAL

1.1 object of study
(i) products (e.g., requirements, specification, code, user interface, installed system ...)

(ii) processes (e.g., designing, testing, reading, inspection, evaluating, users interacting with systems, installing, maintenance ...)

1.2 focus of study
(i) quality (e.g., efficiency, complexity, reliability, correctness, modifiability, reusability, usability and adherence to plan, ...)

(ii) productivity (cost, effectiveness, ...)

1.3 purpose of study
(i) passive (e.g., to understand the object of study, to learn about the object of study, ...)

(ii) active (e.g., to manage, to motivate, to predict, to build, to control, to test or evaluate, ...)

1.4 viewpoint of study
(e.g., organization, manager, developer, customer, end-user, ...)

1.5 context of study
(e.g., additional processes, additional products, humans, methods and tools, ...)

2 STUDY PLAN

2.1 learning approach (i.e., paradigm)
(e.g., natural science, engineering, mathematics, ethnography, ...)

2.2 study design
(e.g., single project layout, replicated project layout, multi-project variation layout, blocked subject-project layout, ...)

2.3 study control
(e.g., influence intended and explicitly manifested, influence intended and implicitly manifested, influence unintended and explicitly manifested, influence unintended and implicitly manifested, ...)

2.4 study location
(e.g., field, laboratory, theoretical (e.g., specification, model), ...)

3 STUDY METHODS

3.1 who performs the study
(e.g., researcher, end user, development team, tester, maintainer, management, HCI specialist, ...)

3.2 what activities are done
(e.g., plan, collect data, validate data, analyze data, interpret results, communicate results, ...)

3.3 when is it done
(e.g., before development starts, requirements, design, interaction design, screen design, coding, testing, after completion, ...)

4 STUDY TECHNIQUES

4.1 nature of the data
(i) kind of data type (e.g., quantitative, qualitative, ...)

(ii) type of data representation (e.g., numbers, profiles, distributions, protocols, logs, diaries, opinions, ...)

(iii) degree of data validity (e.g., verified, independently validated, validated by collector, not validated, ...)

(iv) data granularity (e.g., macro, intermediate, micro, ...)

(v) information derived from the data (e.g., verbal report, written report, model, demonstration, ...)

4.2 data handling mechanisms
(i) data collection mechanism (e.g., automated logging, non-automated logging, forms, checklists, heuristics, interviews, direct observations, protocols - video, audio, interaction - , diaries, questionnaires, elicitation system, ...)

(ii) data validation mechanism (e.g., automated, independent read, redundant data, correlation between researchers, correlation between researcher(s) and user(s), ...)

(iii) data analysis mechanism (e.g., summarize, categorize, statistically analyze, model, ...)

(iv) data interpretation mechanism (e.g., expert system, researcher, researcher and user(s), researcher and designers, ...)

(v) feedback mechanism (e.g., on-line, verbal report, written document - standardized or non-standardized format - , demonstration, ...)

1 STUDY GOAL

Using Basili's GQM approach (3, 9, 11, 35) study goals can be characterized in terms of five dimensions:

1.1 object of study

Which product or process do I study?
Here the term 'product' can include any software document, the software system or just the interface of the software system. The term 'process' can include a process used to develop or use products.

(i) products
Some example categories in the case of products are: requirements, specification, design, code, user interfaces, and installed systems.

(ii) processes
Some example categories in the case of processes are: designing, testing, reading, inspecting, evaluating, using (i.e., users interacting with installed systems), installing and maintaining.

1.2 focus of the study

Which particular aspect of the object do I focus upon?
We may be interested in distinguishing between two example categories: the quality of the final product and the productivity of the development process.

(i) quality
Some example categories in the case of quality are: efficiency, complexity, reliability, correctness, modifiability, reusability, usability and adherence to plan. Usability, however, has been largely ignored by SE until recently, whereas it is of prime importance in HCI. Bennett [4] and Shackel [5] defined usability in operational terms in which they recognized the importance of trying to measure the following criteria: learnability, throughput, flexibility, user attitude, ease of use, and utility. Their definitions helped to provide the foundations for usability engineering [10]. More recently other measurement techniques have been developed including some based on use of heuristics [24, 26, 48], which provide relatively low cost ways of identifying usability problems, and situated studies in which users perform their own tasks in their own working environment rather than in laboratories [10, 22, 54]. As we shall show later, our taxonomy helps to make explicit and explain the inherent differences between these and other forms of usability evaluation.

(ii) productivity
Some example categories in the case of productivity are: cost and effectiveness. Keeping costs low while at the same time developing effective software requires efficient and effective development techniques. It is important not only to test the product in terms of aspects such as code correctness and usability but also to reduce development costs.

1.3 purpose of the study

What is the purpose of the study?
The purpose of the study may be: passive or active. Passive purposes are aimed at better understanding or visualizing existing software items without influencing them, whereas active ones are aimed at actually influencing them in some way.

(i) passive
Some example categories in the case of passive purposes are: to understand, to learn, to assess the product or process being studied.

(ii) active
Some example categories in the case of active purposes are: to manage, to motivate, to predict, to build, to control and to test or evaluate the object of study in order to improve it.

1.4 view point of study

From what perspective do I study?
Anybody interested or involved in the field of software reflects a potential measurement or study perspective. For instance, a manager may view work throughput quite differently from a user. Some example categories are: the organization, manager, developer (or more specifically: designer, tester, coder, evaluator, human factors specialist), customer, and end-user.

1.5 context of study

What contextual information relating to the object of study do I need to study it?
Each object under study has been affected by the context and environment in which it has been developed, maintained or used. In order to study that object it may be necessary to include closely related aspects of that context or environment such as other processes, products, usage scenarios, methods and tools. This is best understood with reference to a specific example as the aspects of interest will vary according to the object of study and the focus. If, for example, the goal is to improve usability, it may be necessary to study not only the user of the system in a usability laboratory or in the field, but also characteristics of the preceding development process in order to be able to improve the system's usability.

To illustrate the use of the study goal dimensions, we have analyzed two example studies below, which we will also analyze in terms of the dimensions of each of the other meta-dimensions.

Example study 1: The goal is to test the hypothesis that 'managers are not able to predict the maintenance workload for their staff during the first year after the system has been installed from a system's architectural design'.

Using our taxonomy this study goal would be described as:
Analyse the process of maintaining (object of study) in order to learn about adherence to plan (purpose of study) of maintenance cost (focus of study) from a manager's perspective (viewpoint of study) in the specific environment and context of the study (context of study).

Example study 2: The goal is to identify the usability problems that users experience when using a spreadsheet.

This study goal can be analysed as:
Analyze the spreadsheet technology (object of study) in order to understand (purpose of study) its usability (focus of study) from the users' perspectives (viewpoint of the study) in the particular environment and context of use (context of study).

2 STUDY PLAN:

The study plan consists of four dimensions: learning, study design, study control and study location.

2.1 learning paradigm

How do I plan on learning?
Learning is about gaining relevant knowledge about the object of study (i.e., developing some models of the object with some focus, for some purpose, from some point of view and within some context) so that more incisive studies can be carried out for building better models, which ultimately lead to modifying and improving the object of study.

There exist radically different learning paradigms. The differences in these conceptual frameworks result from fundamental differences in the traditional objects of study and differences in the beliefs of researchers about the appropriateness of different ways of learning.

The objective which usually underpins any study, either directly or indirectly, is ultimately to bring about improvement. As a scientific approach, 'improvement' involves either short term or long term iterative cycles of watching in order to learn about software phenomena, building new models of those phenomena, modifying the objects of study based on these new models and then watching them again to validate the new models and so on. Learning is achieved through the iteration of watching adherence to explicit or implicit models, revision of existing models or building new models or the application of new models. Two approaches reflecting such iterative improvement are Basili's Quality Improvement Paradigm (QIP) [3, 9, 11, 35, 55], which we described earlier and 'usability engineering' [10], which was also mentioned earlier.

Both QIP and usability engineering view software development as an experimental process. The implication is that we must learn from each development and reuse that experience to improve the current as well as future developments. It is assumed that this kind of learning requires a combination of the natural science and engineering paradigm. Each project is performed according to a sequence of steps: characterize, set improvement goals, select best development approach to satisfy goals, develop and collect data, analyze data, learn and feed back.

The plan/do/check/action approach, proposed by Deming [12], is similar in that it is based on the engineering learning paradigm. Each project is performed according to a sequence of steps: plan the project, do the project according to plan, check the project results against the targets set in the plan, and take corrective actions.

The paradigm adopted for a study has a pervasive influence, as we shall show later. It influences how the study is designed and controlled in terms of where it is located, the type of data that is collected and how that data is analysed. Example categories of learning paradigm in SE and HCI are: natural science, engineering, mathematics, and ethnography.

For the purposes of this discussion, *natural science* includes physics, chemistry, biology and so on, where (natural) phenomena exist as 'facts' in a system, which are created according to given laws. For example, gravity is a natural phenomenon in our solar system. The curve model of a flying ball (of a given weight, thrown with a given force and direction) is based on the law of gravity and cannot be changed. Learning, according to the natural science paradigm, is based on observing existing objects in their real-world environment, building models of their behaviour, and reapplying these new models to validate them.

Constructing new objects according to observed facts is *engineering*. For the purpose of discussion, engineering includes the traditional disciplines of electrical and mechanical engineering and so on, where objects are created by humans. Learning, according to the engineering paradigm, is based on stating hypotheses about the effects of some development process on characteristics of the resulting products, applying this development process, validating the hypotheses and potentially changing the hypotheses.

In engineering laws can change as we change the underlying engineering technology such as when using steel rather than wood. The conceptual framework of engineering assumes that learning is based on building objects and models based on known or hypothetical laws with anticipated implications, testing them, and collecting data to see

to what degree the anticipated implications are real. This, of course, is the over-riding paradigm for software engineering. During the late 70's and 80's it also became an important approach for commercial HCI designers [e.g., 4, 5, 10]. Usability engineering enabled human factors considerations to be taken into account within the overall framework of software development. By adopting this approach usability could be integrated into software development in a way that made it logistically feasible. The similarity of the approach to the already established software development paradigm also made the usability perspective acceptable to design teams.

The conceptual framework of *mathematics* assumes that learning is based on formally proving phenomena to be correct or incorrect within a formally defined closed system of axioms. The importance of formal methods have become acknowledged increasingly in recent years in SE [43, 50], and to a lesser extent in HCI [41, 49].

The *ethnographic* paradigm has its origins in anthropology and sociology and it has also been important in marketing and educational studies. The underlying tenet is the belief that imposing control on certain variables in order to examine the behaviour of others in known scientific conditions will change the very nature of the object of study. It is, therefore, becoming accepted that scientific and engineering approaches are inadequate for studying large complex systems with many inter-related variables [54, 56]. Instead, understanding the usability issues associated with such systems must be gained from observing natural usage within the context itself. Thus, data has meaning only within its context of origin; it cannot be treated objectively as an isolated entity. During the late 1980's some researchers and developers in HCI [e.g., 22, 42] adopted this paradigm for their work. SE, although more traditionally quantitative, was also acknowledging the importance of context in measurement studies [e.g., 3, 28].

The fact that we can actually observe computer science phenomena (e.g., how the qualifications of developers affect the quality of the resulting software) as well as change such phenomena (e.g., by assisting developers with better methods and tools) is a benefit and a problem at the same time. It is a benefit in that we can improve negative phenomena. It is a problem in that sometimes none of the above learning approaches in isolation is sufficient. It is, therefore, perhaps not surprising that some studies adopt different paradigms for different parts of the study. For example, a usability study may collect some data using very controlled laboratory testing (i.e., the scientific paradigm) within an overall development paradigm of usability engineering paradigm [10]. Ethnography may also be employed to try to establish what kinds of usability problems are important to users in their normal working lives. In describing such studies, however, it is quite common for only the dominant paradigm to be acknowledged explicitly [7], which gives a distorted view of the real events.

2.2 study design

How do I plan the lay out of the study?
All studies in the software domain involve teams of individuals applying some (set of) technologies to some objects. An example of such a study is a group of students performing some maintenance tasks on some source code modules, or a development team applying some testing techniques in different projects. In the former case, a maintenance process has been applied by different individuals to different objects (i.e., modules). In the latter case one team applied a testing technique across multiple objects (in different projects). In order to characterize the types of study layouts, Basili et al. [35] referred to a collection of multi-person teams engaged in different tasks as 'subjects' and the collection of separate problems or pieces of software to which these tasks are applied as 'projects'.

The plan of a study will depend on how many objects are studied and in how many project environments or experimental designs. It is obvious that the study of one object (e.g., architectural design and its implication on the maintainability of the final system) in one project (e.g., a specific software development project) requires a different study

plan than the study of a class of objects (e.g., object-oriented architectural designs) in a class of projects (e.g., all projects of organization Y). The categories that have been used to characterize study design in this paper are taken from Basili et al. [35] and they are:

Single project layout, which assumes the study of software (technology) by one subject in one project environment.

Replicated project layout, which assumes the study of software (technology) by multiple subjects in one project.

Multi-project variation layout, which assumes the study of software (technology) by one subject in multiple projects.

Blocked subject-project layout, which assumes the study of software (technology) by multiple subjects in multiple project environments [11].

In practice, not all combinations of study layout and study location categories are likely. Whereas single project and multi-project studies (commonly known as case studies) can be performed at acceptable cost in the field, replicated and blocked subject-project studies cannot. The latter, also referred to as controlled experiments, are typically performed in laboratories with smaller scale objects because of cost. This suggests that laboratory experiments are suited to derive significant results from studying 'small scale' objects in a controlled setting.

2.3 study control

To what degree do I plan to influence the object of study and its environment?
It is practically impossible to study an object (except in the case of static analysis of products) without some how influencing it. This influence can be intended (e.g., a controlled experiment where a particular phenomenon is studied, such as the number of times users select the correct icon) or unintended (e.g., data collection actually disrupts the natural work process of developers). Influence can be explicitly manifested in the object of study (e.g., by changing development criteria) or implicit (e.g., the attitude of developers is changed as a result of them being observed).

Theoretically the following four combinations of these two aspects give the following categories:

Influence intended and explicitly manifested. For example, influence occurs in the case of a controlled experiment to understand the problems of learning different programming languages. In this case, influence is intended. In order to relate observed differences to the difference in programming languages, all other factors (e.g., influence of different qualifications of testers, different design approaches, different degrees of error-proneness) need to be excluded.

Influence intended and implicitly manifested. For example, some contextually based usability studies unobtrusively video users who use the test system to do whatever they wish. Sections of the videotape in which users can be seen to experience difficulty are then shown to the development team and the testers, who discuss the nature of the problems and what kinds of improvements are needed.

Influence unintended and explicitly manifested. A study is often influenced unintentionally in, for example, the case of a measurement-based case study in which the purpose is to test the effectiveness of a testing technique in a project environment. Even though no influence is intended natural work flow may be disrupted because explicit guidelines have to be followed as to when and how data collection forms are to be completed.

Influence unintended and implicitly manifested. A study may be influenced when, for example, a video recording is made in order to better understand the problem solving approaches used by a design team. Of course, no influence is intended and no explicit changes to the design process are being prescribed. However, it is well-known, that if people know that they are being observed their behaviour may change - a phenomenon known as the Hawthorne effect [34].

2.4 study location

Where do I study the object of study?
The site of the study can have a large influence on the nature of the study. The choice of site is intimately related to the overall study approach and particularly the learning paradigm and the kinds of techniques and methods that are used. For instance, an ethnographic study would not be compatible with a controlled laboratory location. Examples of study location categories are:

Natural settings in which the product or process is normally located, which are often called field settings.

Laboratory settings are selected when controlled conditions are required, such as in a study to investigate the suitability of a set of icons for a particular system.

In the case of some theoretical studies a specification or model is used and the study location is of no relevance.

The following hypothetical studies illustrate how the 'study plan' meta-dimension can be applied. They build on the first two studies and show how the taxonomy can be used to progressively characterize the nature of the studies.

Example study 1: The intention of this study (described above in relation to the 'study goal') is to test the hypothesis that 'managers are not able to predict the maintenance workload for their staff from a system's architectural design during the first year after the system has been installed'. In order to achieve the study goal stated above, a combined natural science and engineering learning paradigm will be used (learning paradigm) to investigate 10 project sites (replicated project layout in a field study location). The researchers will attempt to influence the maintenance work as little as possible but some influence is inevitable (influence unintended and implicitly manifested).

Example study 2: The intention of this study (described above in relation to the 'study goal') is to understand how the spreadsheet technology is used in the day-to-day working practices in an office. This will be an ethnographic study (learning paradigm) of working practices in just one office (single project layout in a field study location). The researchers will try to be as unobtrusive as possible but it is realistic to think that they may have some small influence (influence unintended and implicitly manifested).

3 STUDY METHODS

Study methods can be characterized in terms of three dimensions: who, what is done and when it is done.

3.1 Who

Who performs the study?
Some example categories are: researcher, end-user, development team, tester, maintainer, management and HCI specialist. The value of multi-disciplinary teams is being recognised more and more as being essential for examining complex problems. Consequently, many teams will contain several of the above personnel. (This, of course, makes it all the more important that common frameworks and terminology, such as this one, are available to facilitate communication between the different members of the team.)

3.2 What is done
What study activities need to be performed?
Some example categories are: plan, collect data, validate data, analyze data, interpret data, and communicate results.

3.3 When is it done
When are the study activities performed in order to achieve the study goals (described in 1) following the study plan (described in 2)?
The following list of activities can be interpreted as a linear sequence of events or iteratively (e.g., as in user-centred design[25]). Some example categories are: before development starts, during project requirements phase, during design, during project testing phase, and after project completion.

By extending our two previously used examples we show how the 'study methods' meta-dimension can be applied to characterize them further.

Example study 1: In order to study the impact of 'object-oriented architectural designs' on maintenance (referred to above in relation to 'study goal' and 'study plan') in all the projects of an organization according to a replicated study layout, a study method is used that involves design and maintenance personnel from 10 projects and researchers (who). The researchers will plan the study details as well as validate, analyze and interpret the collected data, and the project personnel will collect data (what activities). Planning, analysis and interpretation will be performed after each project completion, and data collection and validation will be performed (what activities) throughout the projects (when).

Example study 2: In the second study (referred to above in relation to 'study goal' and 'study plan') the researchers (who) will work with the end-users (who) as closely and unobtrusively as possible. They will collect situated natural data which they will discuss and analyse (what activities) with the users. Both users and researchers will communicate (what activities) the results to management. The project will take place after one version of the technology has been in place in the office for a year and as part of requirements collection for an upgrade (when) of the product.

4 STUDY TECHNIQUES:
Each of the study activities described in the study method has to be supported by at least one study technique which can be characterized in terms of the following dimensions: the nature of the data and the mechanisms for acquiring, validating, analyzing, interpreting and presenting the findings from it.

4.1 nature of the data
What is the form of the data collected and what information is derived from it?
This dimension is concerned with different characteristics of the data *per se*. The example categories are:

(i) kind of data
What kind of data is to be collected?
Some example categories are: quantitative, qualitative

(ii) type of data representation
How is the data represented?
Often several types of data will be collected in one study. Example categories are: numbers, profiles, distributions, protocols which may be verbal, video sequences or interaction logs and diaries.

(iii) degree of data validity
How is the data validated?

Some example categories are: verified, independently validated, validated by collector, not validated.

(iv) data granularity
What is the general granularity of the data?
Some example categories for describing the data are: macro, micro, or intermediate in granularity. For example, at the micro end modelling techniques are used to analyze cognitive processes, whereas at the macro end a model may help to explain how technological changes alter the way a company operates.

(v) information derived from the data
In what form will the results of the study be communicated to others?
Some example categories are: verbal or written reports, models, demonstrations.

4.2 data handling mechanisms
How do I perform the technique?
Mechanisms are needed for collecting and processing the data so that they fulfil the requirements outlined in the 'study goal' and are operationalized through the 'study plan' and 'study methods' meta-dimensions. Some example categories are: collection, validation, analysis, interpretation and feedback.

(i) data collection mechanism
How is the data collected?
Some example categories are: automated and non-automated logging, forms, checklists, heuristics (i.e. noting specific important aspects), interviews, direct observation, video, audio or interaction protocols, diaries, questionnaires, various forms of elicitation [23].

(ii) data validation mechanism
How is the data validated?
Some example categories are: automated, independent read, redundant data, correlation between researchers, correlation between researcher(s) and user(s).

(iii) data analysis mechanism
How is the data analyzed?
Some example categories are: summarize, categorize, statistically analyse (e.g. find mean, standard deviation, analysis of variance, correlation analysis, regression analysis etc.), model.

(iv) data interpretation mechanism
How is the data interpreted?
Some example categories are: expert system, researcher, researcher and end user(s), researcher and designers.

(v) feedback mechanism
How are the results of the study fed-back to the interested parties?
Some example categories are: on-line, verbal presentation, written document in standardized or non-standardized format, demonstration of system, construction of prototype.

The last episodes in the two hypothetical studies (discussed in relation to the other three meta-dimensions) illustrate how the 'study techniques' meta-dimension can be added to the earlier studies.

Example study 1: Collecting complexity data [data type: quantitative, numbers (representation of data type), verified (validity), micro (granularity), produced as a written report (information derived) from lines of source code via a tool (collection and validation (automated mechanism)]. The data may then be partially analyzed by the tool and partially by researchers and designers who also interpret the findings and feed them

back to the development team as a verbal report supported by a written document in non-standardized format (mechanisms for analysis, interpretation and feedback).

Example study 2: Qualitative data are collected (kind of data) in the form of video protocols, diaries and opinions (data representation). The data are informally validated in discussion with users to establish shared understanding. The granularity of the data is macro. Videotape is the main data collection mechanism. The data are then summarised and categorized by the researchers and users (data analysis mechanism) and a report and presentation are prepared (feedback mechanism).

Notice that although most of the categories in the taxonomy are applied in the analysis of the examples some categories may be unnecessarily detailed for some examples. In the next two sections we show how this taxonomy can be used to analyze the underlying nature of some general measurement approaches in the two fields and some real SE and HCI studies. In section IV, we consider well known measurement approaches that are discussed in the literature of the two fields. Then, in section V, we apply the taxonomy to some specific studies in the two fields. All four of these studies have been reported in the literature and the authors have participated in three out of the four studies.

IV Characterization of example SE and HCI study approaches

The approaches discussed in this section are general in order to show how the taxonomy can be used at this high level for high-lighting key differences.

SE study approaches

Software quality measurement is driven by metrics for measuring various aspects of processes and their resulting products. A number of approaches to studying phenomena in the SE area have been developed [e.g., 1, 2, 3, 9, 11, 13, 35, 50, 52, 55, 61] which differ due to the different backgrounds of their authors and users, as well as their study goals, study layouts, methods and techniques. For illustration purposes, we briefly characterize just three approaches which have been suggested for measurement-based analysis of software related purposes in the SE field. We do not include the widely applied, but usually unsuccessful, attempts to 'measure without objective' approaches.

The *goal/question/measure approach* to measurement has been developed at the University of Maryland [11, 3, 9, 35]. It provides a general mechanism for identifying measures in a goal-directed manner and the context for sound interpretation of collected measurement data. Study goals are defined in a systematic way, refined into a set of quantifiable questions that in turn imply a specific set of measures and data to be collected. The goal/question/measure hierarchy provides the context for back-up data interpretation. It is assumed to be applied in the context of the Quality Improvement Paradigm discussed in section III.

Using our characterization scheme, the goal/question/measurement approach can be viewed as directly supporting the study plan and selection of study techniques (particularly analysis) stages within the measurement process. Any thinkable study goal can be defined. The study goal is used as the source for defining measures with any kind of data, level of granularity and any type of underlying data representation. Learning is based on a combination of natural science and engineering paradigms.

The *Software Quality Metrics (SQM) measurement approach* was originally proposed by Boehm et al. [1] and then refined later by McCall et al. [2]. It provides a mechanism for identifying measures for a variety of factors of the delivered product from a user's perspective. A list of factors of interest, together with their refinements into criteria and metrics, is provided from which to choose. It is assumed to be applied in the context of a closed model which defines all aspects of interest in a product from a user's perspective.

Using our characterisation scheme, the SQM approach can be viewed as also directly supporting the study plan and selection of study techniques (particularly analysis) stages within the measurement process. In its pure form it was intended only for study goals reflecting the users' views of the deliverable software system but its use is often broader. Learning is based on a theoretical paradigm.

The *Quality Function Deployment (QFD) measurement approach* was originally proposed by Kogure and Akao [45]. It provides a mechanism for identifying measures for a variety of factors of the delivered product from a user's perspective. It assumes that aspects and measures of the deliverable system are defined first which will then be traced back to aspects and measures of earlier products (e.g., design , specification, requirements). The idea is to provide a basis for quality control during the production process. This approach is assumed to be applied in the context of the Plan/Do/Act improvement approach.

Using our characterization scheme, the QFD approach can be viewed as directly supporting the study goals and study plan stages within the measurement process. In its pure form it was intended only for study goals reflecting the users' views of the deliverable software system. Learning is based on the engineering paradigm.

HCI study approaches

There have been a number of trends in usability measurement [e.g., 24, 26, 36, 38, 46, 48, 54, see also 7, 47, 57 and 58 for overviews]. In the 70's and early 80's traditional *laboratory testing* was predominant. Gradually, usability engineering [29] began to increase in importance and, as part of this process, another and less formal kind of laboratory testing was done, in which users' performance on benchmark tasks was measured. Usability engineering is a central part of commercial software development in some companies and techniques for improving its cost effectiveness [e.g., 26, 48] continue to be sought. Complimentary techniques are also being developed for collecting data about aspects of usability that cannot be tested in the laboratory and can only be understood by field studies [e.g., 36, 42, 54] in which users are observed working with systems in their natural work environments. In parallel, a body of work has developed in which the focus is on developing models of user interaction which can be used to predict the usability of systems at very early stages of development [e.g., 18, 19, 20, 21, 46]. In the remainder of this section we will characterize these approaches in terms of our own taxonomy. Although we consider each in turn it is quite common for techniques from the different approaches to be used in concert as we have already said.

Laboratory testing is valuable for examining and comparing differences in one or more independent variables. Issues such as which menu and command names are most memorable for users and whether or not menus should be broad or deep in particular systems can be tested using standard laboratory testing techniques [39, 40].

Using our characterization scheme, the object of study is usually how people perform a short, tightly controlled task. Quantitative data are collected which are usually analyzed statistically. The study tends to focus on just a small part of a resulting system. Learning is based on the scientific paradigm.

Usability engineering is a process whereby 'the usability of a product is specified quantitatively, and in advance. Then as the product is built, testing takes place to see whether the planned-for levels of usability have been achieved' [15]. Users' performance on specially designed benchmark tasks is recorded and analyzed in purpose-built usability laboratories [4, 5, 10, 15, 29]. Usability criteria are identified and levels of acceptance are defined. Development proceeds iteratively in cycles of 'design-test-redesign' and the usability of the product is monitored and recorded in a usability specification.

Using our characterization scheme, the object of study is usually people interacting with a computer system at the task level. The data collected are mostly quantitative. Although, in addition to benchmark testing, field observations are often carried out to provide information about how users actually use the system to do their normal work and some of this data is usually qualitative. The field data provides a valuable alternative perspective to that gained from the laboratory data and recognition of the importance of context of use in usability studies continues to gain momentum. Learning is based predominantly on the engineering paradigm but scientific and sometimes ethnographic paradigms also influence the study, through the use of benchmark testing and field observation respectively.

The term *heuristic evaluation* [24, 26, 48] came into use at the end of the 80's. Heuristics are provided to help evaluators, who may be HCI specialists or users, identify usability problems when inspecting a software system. Examples of the kinds of high level heuristics [24] that drive the approach are: be consistent, provide feedback, speak the user's language, provide good error messages, provide short-cuts, and so on. The main advantage of this approach is its low cost compared with other approaches [38], and coupled with this, the speed with which the most obvious problems can be identified and fixed.

Using our characterization scheme, the object of study is usually human-computer interaction. Quantitative data are collected of counts of different kinds of usability problems, which are presented in a written report. Learning is based on the engineering paradigm.

Suchman's [42] *ethnographic* studies did much to demonstrate that certain kinds of interaction can best be understood in relation to the context in which they are situated. At around the same time as Suchman's work in the mid and late 80's researchers and developers at Digital Equipment Corporation needed techniques which enabled them to do more than identify and fix usability problems, as in usability engineering. Furthermore, they needed techniques that were more appropriate for testing multi-user, multi-tasking, multi-media, distributed and co-operative systems and not just the single user systems of the 70's and early 80's. In response to this need, they developed contextual inquiry [10, 22]. The cornerstone of contextual inquiry is the collection of information about the usability of systems in normal working contexts [22]. Records may be kept in the form of notes or on video. Interpretation of this information, whether done immediately or *post hoc*, is a collaborative activity between users and researchers to develop a shared understanding of users' problems. Gradually a bank of relevant information is accumulated which can be fed back to the design team. Various other techniques have also been developed as ethnographic tools for understanding usability [23, 36, 37, 60].

Using our characterization, the object of study is people interacting with computers and other artefacts naturally in their normal environment and no control is intended. The data (i.e., observations) are qualitative. Learning is based on the ethnographic paradigm

Different forms of *cognitive modelling* have been popular for sometime because they hold the lure of predicting usability problems very early on in design from specifications, mock-ups or simple prototypes, which save development effort and money. The keystroke Level Analysis [16], GOMS [17] and Cognitive Complexity Theory [18, 19], which builds on GOMS are well known techniques for modelling users' cognitive and physical behaviour. Cognitive walkthroughs [46] and other techniques [20, 21], which have similar aims have also been developed.

Using our characterization the object of study is generally a specification or prototype. Learning is based on the mathematical paradigm in which a theoretical model is used.

At this high level of analysis the key dimension for distinguishing both SE and HCI approaches is the 'learning paradigm' which, as we said in section III, has a pervasive influence on other dimensions. This is made more explicit in the detailed studies described in the next section.

V Characterization of specific SE and HCI studies

The discussion and characterizations in section IV illustrated how the taxonomy can be used to analyse different measurement approaches from SE and HCI. In this section we apply the taxonomy to two studies from SE and two from HCI to show how it can be used to characterize specific studies. The authors were participants in three of these studies. From these experiences they became aware that communication among members of multi-disciplinary teams could be eased by concisely characterizing details of the studies using a commonly agreed taxonomy. Later it also became apparent that the taxonomy was a useful planning and steering device for projects.

All four examples [27, 28, 29, 30] are published and the characterizations are based on these published accounts. In some cases there is not sufficient information to allow a full characterization of all aspects of the study. Some dimensions simply do not feature in the accounts. In the case of the studies with which we have been involved it has been possible to add some of this extra detail but this could not be done for the other study [29].

SE 1 Complexity/Maintainability Study [28]

Study goal

One goal of this study was to analyze structural aspects of software systems (object of study) in order to understand and predict (purpose of study) their maintainability (focus of study) from a developer's perspective (view point of study) in a distributed operating system domain (context of study).

Study plan

The learning approach assumed a combination of the natural science and engineering learning paradigms as formulated in the TAME improvement paradigm [9] - assumed intended/explicit influence (study control), and assumed a blocked subject-project design layout (study design) to be performed in a laboratory environment (study location).

Study methods

Researchers (who) at the Universitaet Kaiserslautern planned maintenance experiments, in which a number of students had to perform a set of given maintenance tasks in which they validated and analyzed the experimental data, and presented the results [28] (what). The study occured during all stages of maintenance (when).

Study techniques

The students collected data related to maintenance effort (number of staff-hours), locality (number of components affected by a change), type of maintenance task (correction, adaptation, enhancement) and component complexity (fan-in/fan-out, McCabe's complexity, etc.). The data collected was quantitative (kind of data). Data was collected via forms and interviews (data collection mechanism), independently validated by the researchers (data validation mechanism), statistically analyzed using correlation statistics (data analysis mechanism), interpreted by the researchers (data interpretation mechanism) and presented as a written report [28] (feedback mechanism).

The results showed a significant correlation between certain kinds of component complexity and maintenance effort [28]. The study, therefore, indicated that measures of complexity may provide one promising predictor of maintenance effort during the component design stage of a development project.

HCI 1 Evaluation of command, menu and iconic interfaces [29]

Study goal
In this study the interfaces of seven different office systems (objects of study) were studied by designers and researchers (view point of study) in order to evaluate their ease of use (a specific aspect of usability study - purpose of study). The context of the study was probably part of an on-going series of usability studies, which were done to improve the developers understanding of interface design (long term purpose of study - passive) for future product development (focus of study).

Study plan
Benchmark tasks were used to assess the usability of the systems (engineering approach). A total of 76 users participated in the study and they were divided into three classes depending on their previous experience with computers. Each user performed a benchmark task that lasted approximately an hour. Exact details of the study design are not made clear in the paper but it was probably a blocked subject design in a usability laboratory (location). Control was intended and explicitly manifest.

Study Method
Human factors researchers and developers (who) studied representative end-users doing specified benchmark tasks. These researchers planned the task, collected and analyzed the data, interpreted it and reported the results (what). The paper referenced does not mention data validation nor when the study was conducted but the implication is that data were collected during the development of each product and then re-analysed some time later (when).

Study Techniques
Users' performance on benchmark tasks was measured (data collection mechanism). The data collected was quantitative (kind of data). The performance data was represented as a metric (information derived from the data) for rate of task completion (i.e., $S = 1/T \times P \times C$, where T is the time spent on the task, P is the percentage of the task completed, C is the arbitrary time-unit constant of 5 minutes and S is the user's performance score.) The data was then analyzed statistically to examine correlation between user performance and previous experience and then the results were tabulated and interpreted (data handling mechanisms).

SE 2 Testing Study [30]

Study goal
One goal of this study was to analyze different testing techniques - functional testing, structural testing, and stepwise abstraction code reading - (object of study) in order to understand their effectiveness and cost (purpose of study) from a developer's perspective (view point of study) in the NASA/SEL environment (context of study). The focus of the study was to improve the techniques for future use.

Study plan
The learning approach assumed a combination of the natural science and engineering learning paradigms as formulated in the TAME improvement paradigm [9] - assumed intended/explicit influence (study control) and assumed a blocked subject-project design layout (study design) to be performed in a laboratory environment (study location).

Study methods
Researchers at the University of Maryland planned an experiment in which students from the University of Maryland and professional developers from NASA's Software Engineering Laboratory (SEL) performed component testing using three different testing techniques (code reading, functional testing and structural testing), validated and analyzed the experimental data, and presented the results (what). The students and professional developers (who) collected data (what) during each application of a testing

technique involving both the detection of failures and the isolation of the faults that caused the detected failures (what).

Study techniques
For each testing task each individual collected data related to effort spent (number of staff-hours), effectiveness (number of failures/faults found) and effectiveness (estimates of percentages of faults found). The data collected was quantitative (kind of data). Data was collected via forms and interviews (data collection mechanisms), independently validated by the researchers (data validation mechanism), statistically analyzed (data analysis mechanism), interpreted by the researchers (data interpretation mechanism), and presented as verbal reports to the SEL and in written reports [30] (feedback mechanisms).

This study resulted in a number of models explaining the relative cost and effectiveness of the testing techniques studied. Stepwise abstraction code reading has been identified as being superior to the other techniques in terms of cost and effectiveness. These models have been validated for SEL environment.

HCI 2 study to evaluate the usability of different usability evaluation approaches and techniques [27]
The study goal
The goal of this study was to evaluate (purpose of study) how some established and some novel evaluation techniques (objects of study) captured the usability issues (focus of study) in seven prototype interfaces (indirect objects of study) in a small company (context of study). Researchers' and end-users' worked on this study together (view point of study).

The study plan
The learning approach was primarily ethnographic. The intention was to influence normal working practices as little as possible but some techniques did turn out to be fairly intrusive (control was unintended which turned out to be explicitly manifest). The study used a repeated subject-project layout in the company's offices (study location).

Study methods
The study was carried out by researchers and users (who), who collected and analysed a variety of different types of data including protocols, logs, users opinions etc. (what) while users did their own work. The study took place outside of normal product development (when).

Study techniques
The data collected included: quantitative performance measures from the keystroke and interaction logs; video protocols and users' opinions contributed qualitative data (kind of data). This data consisted of numbers, comments, video and audio recordings (data representation). The granularity of the data varied from individual key presses (micro) to users' opinions (macro). The data handling mechanisms included a number of data collection techniques (e.g. automatic software logging of interaction and key-presses, video recording and interviews). The data were informally validated by the researchers and users. Data analysis varied according to the type of data; protocol and interview transcripts were examined and the responses were categorized. Keystrokes were analyzed using a form of keystroke modelling. Researchers interpreted the data and the information derived from it was presented in a written report.

A comparison of the main characterizations of the two SE and HCI studies is summarized in Table 2. SE 1 (software complexity/maintainability) and HCI 1 (interaction styles) are studies of the characteristics of artefacts, while SE 2 (testing techniques) and HCI 2 (evaluation techniques) are studies of development processes.

Notice that there are many similarities between the studies as well as some notable differences.

One of the main similarities in the study goals is the focus on future development. Testing is done not just to rectify the immediate problems identified but also to feedback longer term learning for future improvements; an explicit philosophy of the GQM approach and usability engineering. This may seem an obvious thing to do but it has not always been recognized as such, as we pointed in section III.

The study plans of the two SE studies and the first HCI study are very similar and quite different to that of the third HCI study. The key difference between HCI 2 and the others is the learning paradigm from which all the other differences in this meta-dimension and the next two dimensions follow. While there has been increasing awareness of the importance of context in both fields, some HCI specialists are now looking to social science techniques to help to explain the complex interactions that occur between groups of people interacting with a variety of equipment, including computers, in natural environments. Although context is treated differently by the two fields, there is mutual recognition of its importance by some researchers in both fields.

Fundamental differences between the two disciplines do not appear in the study methods meta-dimension at the level of detail of this analysis. In these studies and in general, there is more emphasis on data validation in SE, often involving automated techniques.

There are many differences in the techniques used in the two fields and in the nature of the data obtained using them. Even usability engineering studies, such as HCI 1 which is comparatively quantitative compared with HCI 2, tend to use interviews and to collect video protocols. These practices are not used in SE. HCI must adopt these more qualitative approaches in order to adequately take account of the range of user behaviour and attitudes that prevail. It does not matter how well a system rates in terms of formal usability tests, if users do not like the system they will not use it.

Our research has benefited from the greater understanding that we have developed for each other's work and fields of study and the increasing ease with which we can communicate using our framework and terminology. In addition, the analyzes have provoked questions which otherwise would not have been asked.

Dimensions	SE 1 Software complexity & maintainability	HCI 1 Interaction style evaluation	SE 2 Testing techniques	HCI 2 Evaluation of techniques
Goal				
object	software systems	office systems	functional and structural testing & code reading	evaluation techniques
focus	maintenance	future development	future developments	future use: effectiveness of capturing usability issues
purpose	understand & predict	evaluate, understand & improve	understand effectiveness & cost	evaluation
viewpoint	developer	developers	developers	researchers and end-users
context	distributed operating system domain	on-going series of usability studies to improve product development	laboratory	small company
Plan				
learning	science & engineering	engineering	science & engineering	ethnographic
design	blocked	probably blocked - but not clear from paper	blocked	repeated subject
control	intended & explicit	intended & explicit	intended & explicit	unintended
location	laboratory	laboratory	laboratory	offices of small company
Methods				
who	researchers	researchers & developers	students & developers	researchers & end-users
what	plan experiments which students carry out	plan, collect, analyze, interpret, communicate	as for SE 1 but more emphasis on experiments	plan, collect, analyze, interpret and communicate
when	during all stages of maintenance	during development	outside of any product development	outside of any product development

Techniques				
data	quantitative	quantitative; typically supported by interviews but not mentioned in this study	quantitative & qualitative	mostly qualitative with a little qualitative data from keystroke & interaction logs
mechanism	GQM approach; forms, automated tools, correlation analysis; interpreted by researcher & written doc. produced	benchmark data collected, analyzed statistically, interpreted and communicated in written form	broadly the same as for SE 1 but interviews also conducted	audio, video, interaction protocols categorized, keystrokes modelling; interpreted and communicated in verbal and written reports

VI Using the taxonomy in research and practice

The aim of our taxonomy is to provide a common framework for describing SE and HCI measurement. This framework fulfils four key functions for research and practice. One function is to facilitate *post hoc* analysis of measurement studies to make the activities and associated underlying decisions explicit so that the knowledge can be reused in future studies. In this role the framework supports future research and practice by revealing patterns, strengths and weaknesses in previous work. This enables teams to build up and transfer knowledge and experience from one project to another. A second function is as a *planning aid* for steering and structuring future research by enabling researchers to plan and predict ahead of time. A third function is for *guiding* practice as the study is actually being done. A fourth function is to *facilitate communication* between researchers and within and between development teams by providing a common language and vocabulary for discourse. This fourth function is particularly important for large multi-disciplinary teams composed of people with different knowledge, skills and experience; many of whom will not have worked together before.

Reuse of Results

We have already stressed the important trend towards viewing measurement studies as a source of knowledge, which feeds back into a continually growing body of corporate knowledge. This knowledge and experience is then available for reuse in future developments so it is not necessary to start afresh each time. This concept is important in many forms of software engineering where, for example, code or modules are designed to be reused in this way. The results from measurement studies need not be treated differently. Indeed, usability specifications containing the results of tests over many generations of the product development cycle are an example of how this can work. However, as well as reusing the results of testing, which is what is recorded in usability specifications, it is possible to record much more detail about the actual testing process by using our taxonomy. This makes it easier to understand exactly what was done and why, so interpreting the results becomes more meaningful. We are also exploring ways of documenting our taxonomy so that testing information can be recorded in a standard form and these records can be searched easily. This would improve its usability; particularly for training new team members. It would also make it easier to spot testing problems and inadequacies.

Planning

The taxonomy is hierarchical and the main structure of the taxonomy comes from the four meta-dimensions (i.e. the study goal, the study plan, the study methods and the study techniques). The hierarchical structure coincides well with a number of structured design methods (e.g., JSD [31] and SSADM [32]). For researchers who

prefer to work in a top-down fashion the taxonomy provides a natural structure for thinking about and planning measurement studies. However, as is being increasingly recognised, although design practice tends to be portrayed as being top-down, in practice much design actually results from numerous migrations up and down the levels, with chunks of low-level design being fitted into a more global top-down framework. Planning measurement studies often follows a similar pattern and the taxonomy has a role in helping researchers to keep track of their overall aims. It also supports goal directed studies as advocated by Basili in the GQM methodology [35] rather than focusing too strongly on techniques with, in some cases, an over zealous desire to obtain metrics for their own sake.

The outline structure of the taxonomy presented in Table 1 can be used as an *aide memoire* by researchers to check that they have considered all the important elements when planning their studies. In this role it can be traversed up and down as necessary and used as a template to characterize and communicate study plans. With further development, additions could be made to the taxonomy, which would convert it into a more active guide. It could even be automated. Other factors which would need to be taken into account are the logistics which constrain studies. For example, if a laboratory environment is to be used to obtain user performance measures, personnel with appropriate experience of experimental and statistical techniques must be available to do the work. An automated planning guide of this type would also be valuable for guiding new and inexperienced team members.

Guiding practice

As well as guiding the planning of future studies the framework is useful for steering practice. It provides a way of checking that the actual study follows the plans set out. Any changes to the original plan can be documented within the taxonomy. For example, if a decision is made to change the interviewing practice from open-ended interviews to structured interviews in order to save time, a study that started off in the ethnographic paradigm will no longer fit within that paradigm. Similarly, if observational data is categorized and analyzed quantitatively the paradigm may be violated. Such departures from the original plan need to be acknowledged and the data analyzed and interpreted appropriately. There are also times when it is necessary to use a mix of learning paradigms and our taxonomy helps to make this explicit too.

Communication

The most valuable role that the taxonomy has played in our own work is to facilitate communication between researchers from different disciplines, developers and other software specialists. Using this taxonomy we have gained insights into each other's measurement practices as well as our own. This has enabled us to establish shared understanding across the disciplines, which has allowed us to talk about similarities and differences more effectively than previously. Many groups suffer from inter-disciplinary communication problems and these need to be overcome. It is clearly desirable and natural for SE and HCI specialists to work together as both are concerned with complimentary aspects of software design and development. Funding organizations actively encourage submission of collaborative proposals and there are efforts to integrate SE and HCI development methods. The fact that SE and HCI measurement studies can be analyzed using just one taxonomy, as we have done, illustrates that the two fields already have much in common and that they would each benefit from better communication with the other.

VII Conclusions: moving together

SE is concerned primarily with understanding quality and productivity aspects of all software products and processes and their interdependencies, whereas HCI is primarily concerned with understanding the usability of the final software product and the development processes which help contribute to achieving this. The methods and techniques for measuring software and usability have correspondingly developed in different ways. However, during recent years there have been some attempts to

integrate software and HCI practices [e.g., 51]. Our taxonomy will help further to bring software measurement and usability testing even closer together by providing a common framework which encourages communication, joint planning of projects and helps to guide practice. Being able to characterize SE and HCI measurement is desirable because it encourages clarity, conciseness and shared understanding between the two disciplines. This should ultimately lead to better testing methods, which will in turn produce more efficient and usable software.

In the previous sections we have illustrated how our taxonomy can be used to characterize both existing measurement approaches and specific studies performed according to these approaches in the SE and HCI communities. These analyses have indicated some differences and some commonalties between the way measurement is done in the two fields.

There are differences in the paradigms in which the two disciplines tend to operate, although this is changing. SE measurement studies are predominantly within the natural science and engineering paradigms and they tend to be quantitative and often employ formal methods and statistics. Although some HCI studies are firmly within the scientific paradigm (e.g. the early studies in which menu designs were compared) and the engineering paradigm (i.e. usability engineering), HCI lends itself less readily to strictly quantitative testing. While HCI researchers strive to be rigorous and scientific in their work, HCI is by nature less traditionally quantitative. One reason for this is that peoples' likes and dislikes when using a system are not generally expressed quantitatively, but these opinions are an essential component in assessing the usability of many systems. Another reason is that people do their jobs in different and often quite idiosyncratic ways which may not be predicted by software designers. Furthermore, SE not only lends itself more readily to quantification but it has also had more time to develop measurement techniques because it has been a recognised discipline of Computer Science for longer than HCI.

Another difference that has been mentioned already concerns the attitude of the two communities towards recognising the role of context of the studies. SE has paid little attention to the context of studies, including human influence. However, this is now changing with the introduction of methods such as GQM, in which study goals are stated in a context-dependent manner. HCI has tended to pay more attention to context than SE. The development of systems for computer supported collaborative working have heightened this interest further. The results of testing such systems are very limited, if not meaningless, unless the rich variety of activity in which the system is a part is taken into account. For this reason some HCI researchers are turning to the ethnomethodological techniques of anthropology and the social sciences in order to obtain more contextually based information. Although such techniques are not part of the software engineers' repertoire, it is interesting to note how both disciplines are acknowledging the importance of context and each adopting their own more contextually oriented approaches. By using a framework like ours, in which context and other human-oriented dimensions are made explicit along with SE parameters, a shared understanding of such issues is promoted. This is important for the development of good software.

For both disciplines rapid technological advances continue to present new challenges for those working in measurement. For example, with the development of new interaction styles, collaborative and distributed systems, virtual reality environments and so on, it becomes essential to consider what we need to known about both the functionality and usability of these systems. It is unlikely that existing measurement approaches can deliver all the information needed so both disciplines will need to find new and more appropriate ones. Using a common framework to discuss, record and communicate about the goals, plans, methods and techniques which are adopted will bring greater integration between SE and HCI, which should lead to the production of better products.

Acknowledgements
We would like to thank Bill Curtis, the editor responsible for our paper, Christiane Differding from the Universitaet Kaiserslautern, and three anonymous referees for their comments and suggestions for improving the first draft of this paper.

VIII References

[1] Boehm, B. W., Brown, J. R. & Lipow, M. (1976), Quantitative Evaluation of Software Quality, *Proc. 2nd International Conference on Software Engineering*, pp. 592-605.

[2] McCall, J. A., Richards, P. K. & Walters, G. F. (1977), Factors in Software Quality, *RADC TR-77-369*, Rome Air Development Centre, NY, USA.

[3] Basili, V. R. & Weiss, D. M. (1984), A Methodology for Collecting Valid Software Engineering Data, *IEEE Transactions on Software Engineering*, SE-10, 6, pp. 728-738.

[4] Bennett, J. L. (1984) Managing to Meet Usability Requirements: Establishing and Meeting Software Development Goals, in Bennett, J. L., Case, B., Sandelin, J. & Smith, M. (eds.) *Visual Display Terminals: Usability Issues & Health Concerns*, Prentice-Hall, pp. 161-184.

[5] Shackel, B. (1986) Ergonomics in Design for Usability, in Harrison, M. D. & Monk, A. (eds.) *People and Computers: Designing for Usability.* Proceedings of the Second British Computer Society Human-Computer Interaction Specialist Group Conference, York, pp. 45-64.

[6] Long, J. & Whitefield, A. (1986) *Evaluating the User Interface*, A Tutorial presented at HCI'86, York, September.

[7] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. & Carey, T. (1994) *Human-Computer Interaction*, Part IV, Interaction Design: Evaluation, pp. 595-708, Addison-Wesley, Woking, England.

[8] Long, J. & Dowell, J. (1989) Conceptions of the Discipline of HCI: Craft, Applied Science and Engineering, in Sutcliffe, A. & Macaulay, L. (eds.) *People and Computers V.* Proceedings of the Fifth British Computer Society Human-Computer Interaction Specialist Group Conference, Nottingham, pp. 9-32.

[9] Basili, V. R. & Rombach, H. D. (1988) The TAME Projects: Towards Improvement Oriented Environments, *IEEE TSE*, pp. 758-773.

[10] Whiteside, J., Bennett, J. & Holtzblatt, K. (1988) Usability engineering: Our experience and evolution, in Helander, M. (ed.) *Handbook of Human-Computer Interaction*, North Holland, Amsterdam, pp. 791-817.

[11] Basili, V. R. (1985) *Quantitative Evaluation of Software Engineering Methodology*, Technical Report, TR-1519, Dept. of Computer Science, University of Maryland, College Park.

[12] Deming, E. W. (1986) *Out of the Crisis*, Center for Advanced Engineering Study, MIT Press, Cambridge, MA.

[13] Sunazuka, T., Azuma, M. & Yamagishi, N., (1985) Software Quality Assessment Technology, *Proceedings of 8th International Conference on Software Engineering*, London, UK, pp. 142-148.

[14] Sunazuka. T. & Basili, V., R. (1989) *Integrating Automated Support for a Software Management Cycle into the TAME System*, Technical Report, CS Department (CS-TR-2289) and Umiacs (UMIACS-TR-89-75), University of Maryland, College Park, MD 20742, USA.

[15] Tyldesley, D. A. (1988) Employing usability engineering in the development of office products, *The Computer Journal* , 31, 5, pp. 431-436.

[16] Card, S. K., Moran, T. P. & Newell, A. (1980) The Keystroke Model of user performance time with interactive systems, *Communications of the ACM* , 23, 7, pp. 396-409.

[17] Card, S. K., Moran, T. P. & Newell, A. (1983) *The Psychology of Human-Computer Interaction*, Lawrence Earlbaum, Hillsdale, NY.

[18] Kieras, D. & Polson, P. G. (1985) An Approach to the Formal Analysis of User Complexity, *International Journal of Man-Machine Studies*, 22, pp. 365-394.

[19] Kieras, D. E. (1988) Towards a Practical GOMS Model Methodology for User Interface Design, in Helander, M. (ed.) *Handbook of Human-Computer Interaction*, North Holland, Amsterdam, pp. 135-157.

[20] Payne, S. J. & Green , T. R. G. (1986) Task-Action Grammars: A Model of the Mental Representation of Task Languages, *Human-Computer Interaction*, 2, 2, pp. 93-133.

[21] Howes, A. & Young, R. M. (1991) Predicting the Learnability of Task-Action Mappings, *Human Factors in Computing Systems CHI'91 Conference Proceedings,* ACM Press, New York, pp. 113-118.

[22] Wixon, D., Holtzblatt, K. & Knox, S. (1990) Contextual Design: An Emergent View of System Design, in Carrasco Chew, J. & Whiteside, J. (eds.) *Human Factors in Computing Systems CHI'90 Conference Proceedings,* ACM Press, New York, pp 329-336.

[23] Wellbank, M. (1990) An Overview of Knowledge Acquisition Methods, *Interacting with Computers: The Interdisciplinary Journal of Human-Computer Interaction*, 2, 1, pp. 83-91.

[24] Molich, R, & Nielsen, J. (1990) Improving a Human-Computer Dialogue, *Communications of the ACM*, 33, 3, pp. 339-348.

[25] Gould, J. D. & Lewis, C. (1985) Designing for usability; Key Principles and What Designers Think, *Communications of the ACM*,, 28, 3, pp. 300-311.

[26] Nielsen, J. (1989) Usability Engineering at a Discount, in Salvendy, G. & Smith, M. J. (eds.) *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, Elsevier Science Publisher B.V., Amsterdam, pp. 394-401.

[27] Crellin, J. & Preece, J. (1991) Case study of using different evaluation techniques in a small company, *Information and Software Technology*, 33, 5, pp. 366-382.

[28] Rombach, H. D. (1987) A Controlled Experiment on the Impact of Software Structure on Maintainability, *IEEE Transactions on Software Engineering*, SE-13, no.3, pp. 344-354.

[29] Whiteside, J., Jones, S., Levy, P. S. & Wixon, D. (1985) User Performance with, Command, Menu and Iconic Interfaces, *Human Factors in Computing Systems CHI'85 Conference Proceedings,* ACM Press, New York, pp. 185-191.

[30] Basili, V. R. & Selby, R. W. (1987) Comparing the Effectiveness of Software Testing Strategies, *IEEE Transactions on Software Engineering*, SE-13, 12, pp. 1278-1296.

[31] Jackson, M. (1983) *System Development*, Prentice-Hall, London.

[32] Downs, E., Clare, P. & Coe, I. (1988) *Structured Systems Analysis and Design Method: Application and Context*, Prentice-Hall, London.

[34] Brown, J. (1954) *The Social Psychology of Industry*, Penguin Books, Baltimore, MD, 1954.

[35] Basili, V. R., Selby, R. W. & Hutchens, D. (1986), Experimentation in Software Engineering, *IEEE Transactions on Software Engineering*, SE-12, 7, pp. 733-743.

[36] Crellin, J. M. (1990) PROTEUS: An Approach to Interface Evaluation, in Diaper, D., Gilmore, D., Cockton, G. & Shackel, B. (eds.) *Human-Computer Interaction INTERACT'90*, North-Holland, Amsterdam, pp. 389-394.

[37] Shaw, M. L. G. & Gaines, B. R. (1991) Supporting Personal Networking through Computer Networking, *Human Factors in Computing Systems CHI'91 Conference Proceedings,* ACM Press, New York, pp. 437-438.

[38] Jeffries, R., Miller, J. R. Wharton, C. & Uyeda, K. M. (1991) User Interface Evaluation in the Real World: A Comparison of Four Techniques, *Human Factors in Computing Systems CHI'92 Conference Proceedings,* ACM Press, New York, pp. 119-124.

[39] Paap, K. R. & Roske-Hofstrand, R. J. (1988) Design of Menus in Helander, M. (ed.) *Handbook of Human-Computer Interaction*, North-Holland, Amsterdam, pp. 205-235.

[40] Barnard, P. J. & Grudin, J. (1988) Command Names in Helander, M. (ed.) *Handbook of Human-Computer Interaction*, North-Holland, Amsterdam, pp. 237-255.

[41] Thimbleby, H. (1990) *User Interface Design*, Chapters 13 and 14, ACM Press, Addison-Wesley, Wokingham, England, pp. 287-344.

[42] Suchman, L. (1987) *Plans and Situated Actions - The Problem of Human-Machine Communication*, Cambridge University Press, Cambridge, England.

[43] Zelkowitz, M. V. (1990) A Functional Correctness Model of Program Verification, *IEEE Computer*, November, pp. 30-39.

[44] Shneiderman, B. (1991) An Agenda for Human-Computer Interaction: Science and Engineering Serving Human Needs, Report on Workshop sponsored by NSF, *SIGCHI Bulletin*, October.

[45] Kogure, M. & Akao, Y. (1983) Quality Function Deployment and CWQC in Japan, *Quality Progress*, October, pp. 25 - 29.

[46] Polson, P. G., Lewis, C., Rieman, J. & Wharton, C. (1992) Cognitive Walkthroughs: A Method for Theory-based Evaluation of User Interfaces, *International Journal of Man-Machine Studies*, 36, pp. 741-773.

[47] Hix, D. & Hartson, H. R. (1993), *Developing User Interfaces: Ensuring Usability Through Product and Process*, John Wiley & Sons, Inc., New York.

[48] Nielsen, J. (1992) Finding Usability Problems Through Heuristic Evaluation , in Bauersfield, P., Bennett, J. and Lynch, G. (eds.) *Human Factors in Computing Systems CHI'92 Conference Proceedings*, ACM Press, New York, pp. 373-380.

[49] Alexander, H. (1987) *Formally-Based Tools and Techniques for Human-Computer Dialogues*, Ellis Horwood, Chichester, UK.

[50] Hekmatpour, S. & Ince, D. (1988) *Software Prototyping, Formal Methods and VDM*, Addison-Wesley, Wokingham, UK.

[51] Curtis, B. & Hefley, B. (March, 1992) Defining a Place for Interface Engineering, *IEEE Software*, pp. 84- 86.

[52] Rozum, J. (1993) *Concepts on Measuring the Benefits of Software Process Improvement*, Technical Report CMU/SEI-93-TR-9, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[53] Florac, W. A. (1992) *Software Quality Measurement: A Framework for Counting Problems and Defects*, Technical Report CMU/SEI-92-TR-22, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

[54] Monk, A., Nardi, B., Gilbert, N., Mantei, M. & McCarthy, J. (1993) Mixing Oil and Water? Ethnography versus Experimental Psychology in the Study of Computer-Mediated Communication, *Human Factors in Computing Systems CHI'93 Conference Proceedings*, ACM Press, New York, pp. 3-6.

[55] Basili, V. R., Caldiera, G. & Rombach, H. D. Experience Factory (1994), in Marciniak, J. J. (ed.), *Encyclopedia of Software Engineering*, Vol. 1, John Wiley and Sons Inc., New York, pp. 469-476.

[56] Walsham, G. (1993) *Interpreting Information Systems in Organizations*, John Wiley, Chichester.

[57] Shneiderman, B. (1992) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 2nd Edition, Addison-Wesley, Reading, MA., chapter 13.

[58] Dix, A., Finlay, J., Abowd, G. and Beale, R. (1993) *Human-Computer Interaction*, Prentice-Hall, Hemel Hempstead, chapter 11.

[60] Greenbaum, J. & Kyng, M., eds., (1991) *Design at Work: Cooperative Design of Computer Systems*, Lawrence Erlbaum Associates, Hillsdale, NJ.

[61] Rombach, H. D. (1990) Design Measurement: Some Lessons Learned, *IEEE Software*, 7, 2, pp. 17-25.