APPLICATION OF A PROBLEM-SOLVING METHODOLOGY TO SOFTWARE USABILITY

GILBERT JAMES MANSELL

Master of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

February 1991

The University of Aston in Birmingham

Application of a Problem-Solving Methodology to Software Usability

Gilbert James Mansell

M.Phil.

1991

A real-world problem is analysed using the contingency theory of Checkland and Jackson & Keys. The problem was presented by a computer manufacturer, and is concerned with the usability of software products. The problem is taken to be soft rather than hard, and occurring in a complex multi-organisational system where the relationships between the systems agents, its customers and the problem-solvers is coercive. A methodology representative of a radical design paradigm is used to structure the problem. The suitability of the chosen methodology, and the radical design paradigm, for the problem context encountered in this research, is evaluated in this thesis. The Jackson & Keys framework, augmented by Jackson to cater for coercive problem contexts, is also evaluated.

SYSTEMS METHODOLOGY, SOFTWARE USABILITY, DESIGN METHODOLOGY, CRITICAL SYSTEMS THEORY.

ACKNOWLEDGEMENTS

During the course of this research the author was an investigator on Alvey/SERC project GR/D/72952 - MMI/143, along with Mrs. L. A. Macaulay and Dr. C. J. H. Fowler. No part of the research described in this thesis was performed in collaboration with these colleagues. The research was supervised by Mr. D. E. Avison.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

This thesis describes action research into the problem of software usability. Chapter 2 explains the nature of action research, which is a method of enquiry in the social sciences. The problems associated with action research are discussed, in particular the problem of client domination since this phenomenon arose in this research. An overview of alternative methodologies for action research is given, and a more detailed description of the tradition of radical design, since a methodology in this tradition was used in this research. Theory that enables a choice of methodology to be made is described, and the particular problem of methodology choice in a coercive context is discussed.

Chapter 3 provides the context for the particular situation studied in this research. It defines and categorises computer software, and describes the engineering process by which it is produced. Use of software by organisations that are building information systems is also described. Finally, the problems inherent in software production and use are defined.

Chapter 4 describes a particular project with which Mansell was associated, conducted on behalf of a computer manufacturer and sponsored by the Alvey Directorate. The project was to create a methodology that would improve the usability of the manufacturer's software. Mansell's reaction to the course that the project took was unfavourable, and triggered research into methodology for problem structuring, in particular in coercive situations. Use was made of the work of Jackson & Keys (1984)

and Jackson (1987a). A decision was made to use an Open University design methodology (devised for course T262) to analyse the problem.

Chapter 5 describes the application of the chosen methodology. A game called PIG (Problem Identification Game) was played to explore the problem structure, and then a number of design methods were applied to develop a solution. These methods included a 'user trip' which involved practical experience in the problem situation.

In Chapter 6 the application of the methodology is evaluated, and the suitability of the radical design tradition for action research is discussed. The Jackson & Keys framework for categorising methodologies is analysed and evaluated.

Chapter 7 summarises the work that has been done, and draws conclusions under the headings of software usability, radical design, contingency theory for methodology choice, and the Alvey project that sparked off the research.

CHAPTER 2 - ACTION RESEARCH

1. Introduction

This thesis describes an action research project. In this chapter the nature of action research is defined together with problems that may arise in its execution. As will be seen in Chapter 4 serious problems did indeed arise in this project. A number of methodologies exist for performing action research deriving from alternative paradigms. This chapter describes a particular approach described as 'radical design', since it was the one used in this project. Some theoretical considerations concerning choice of methodology are also presented.

2. The Nature of Action Research

Action research has been variously defined (Rapoport 1970, Foster 1972, Susman & Evered 1978), and the following themes emerge:

[1] Action research is intended to contribute both to scientific theory and to effective action in everyday life.

[2] Action research involves intervention in problematic situations, both by the researcher and other stakeholders, with an intention of bringing about change.

A theme that is sometimes stressed is that the change brought about by action research may be in the perceptions, values and attitudes of the participants in the research project. This orientation derives from the origin of action research in behavioural science, in particular the work of

Lewin (1947). From this perspective the purpose of action research is to bring about education or re-education of the participants. Action research from this perspective was pioneered in Britain by members of the Tavistock Institute (Rapoport 1970, Clark 1976). Another important contributory stream to action research, however, was operational research (OR), concentrating on apparently objective technical problems. Operational research was founded on the assumption (Churchman et al. 1957) that problems arise as a result of malfunctioning systems. Knowledge about systems behaviour was gained by mathematical modelling and experimentation. The socio-technical systems movement (Emery & Trist 1960) similarly relied on optimising the functioning of systems, but concentrated on mismatches between social and technical systems. More recent developments in action research methodology (Checkland 1981) reject the idea that human action is systemic, and treat the notion of system as epistemological rather than ontological. Writers on the methodology of action research vary in the stress they place on problem-solving or learning as the primary purpose of action research. Nevertheless, the expectation of many potential collaborators in action research will be that pressing practical problems will be solved.

The start-point of action research, then, is the identification of a real-world problem. An iterative cycle then ensues of action planning, action taking, evaluation and the specification of the learning that has taken place (Susman & Evered 1978). Real-world problem-solving is so-called to distinguish it from the laboratory problem-solving undertaken by natural scientists. Action research is a form of applied social science, and the researcher has little control over the definition of the research problem (although that definition may change as a result of the action

research process). Problem identification may be a prolonged process, and any particular project may proceed little beyond structuring the problem. For this reason the concept of problem-solving has been questioned by Checkland (1981), and rejected by Eden (1987).

Heller (1986) distinguishes between five different approaches to problem-solving. The traditional approach is that adopted in the natural sciences, and to some extent in the social sciences, particularly psychology. This approach uses laboratory research or carefully controlled field experiments to test or establish scientific theory. The problems to be solved are posed by the researchers, and no particular attention is given to practical application of the results. At the opposite end of the problem-solving spectrum is client-dominated consultancy. No explicit attempt is made to generate validated knowledge. Theory may be drawn upon but it may be heavily influenced by personal experience and 'common sense'. The problem to be solved is posed by the client and practical efficacy is paramount. In between these two extremes lies the opportunity for action research. Heller describes three possible situations. Firstly, there is the situation where the researcher is dominant but is willing to build bridges to users of the research findings. Secondly, there is the situation where the researcher and a client share responsibility for problem identification and research design. Thirdly, there is the situation where a client with a problem requests help from a researcher. Heller uses the term *research action* where the project has a significant emphasis on fact-finding, and *action research* where the project has a greater emphasis on implementation. The view of problem-solving as a spectrum of activity ranging from pure basic science to consultancy, with action research located somewhere between the two, is a valuable clarification of the status of this type of

research. Nevertheless it obscures the philosophical differences between pure scientists and action researchers.

The model of explanation central to traditional science has been formulated by Popper (1959). According to this account, laws of science explain phenomena. The laws may be validated by deducing from them, in conjunction with certain initial conditions, descriptions of events that may be observed. This 'covering-law' model is argued by some to apply to the social as well as the natural sciences. The model is acceptable only if there is the possibility of agreement between observers as to the nature of the events that have been observed. Disagreement about theory is possible using a covering-law model, but not disagreement about what has been observed.

There is a long history of rejection of the traditional approach of natural science by interpretive social scientists. According to this perspective, social phenomena are meaningful to the human beings who create them, whereas the events of the natural world are independent of subjective meaning. The meanings understood by social actors cannot be observed. Two different actors may interpret the same action, in which they are both involved, differently. Understanding of social phenomena necessitates a hermeneutic rather than an empirical process, therefore.

Action researchers are committed to producing knowledge in the service of action (practical knowledge), not knowledge for its own sake. The covering-law model is only partly adequate for this purpose. In addition to being accurate, practical knowledge must be usable by the participants in some problem situation. Human action is undertaken for a purpose, and practical knowledge must include knowledge of ways of establishing and clarifying purpose. Practical reasoning is concerned with ends

as well as means, unlike conventional scientific reasoning that excludes value judgements. Stakeholders in a problem situation will possess some practical knowledge before conducting any form of research. This knowledge may be tacit, however, rather than explicitly defined. Methodologies for action research provide explicit generalised knowledge about how to take action, including means of deciding on what purposes are desirable. A number of alternative *methodologies* are available (see section 3 of this Chapter) that differ in their theoretical assumptions and practical applicability. Contingency theory exists that guides choice between methodologies. A particular theory, that of Jackson & Keys (1984), is discussed in section 3 of this Chapter.

An action research project may produce practical knowledge of use to stakeholders in a problem situation. Alternatively, a project may add to knowledge about the methodology used to conduct it, or the contingency theory used to choose the methodology.

3. Problems Associated with Action Research

Action research has not lacked criticism as a mode of enquiry, and difficulties may arise when it is used. Firstly, action research may be accused of being 'unscientific' because it works in only one setting at a time, and relies on local knowledge rather than application of general laws to specific cases. The covering-law model may be inadequate for action research, but it is not irrelevant, the criticism runs. In fact, the equivalent of a scientific theory in action research is a *methodology*, which is a theory of action. If successive uses of a methodology in a number of different projects yield satisfactory results then the theory is to some extent verified. Secondly, action research does not generate

'data' as in positivist science, but rather interpretations of situations. An interpretation is more or less persuasive rather than undeniably the case, and an alternative interpretation may be possible. The hermeneutic process for validating interpretation cannot break out of what is known as the 'hermeneutic circle' i.e. an interpretation can be validated only by comparison with another interpretation. One answer to this type of criticism is to re-examine the nature of conventional science, and argue that the story about 'laws of nature' misrepresents the social nature of scientific activity. According to Kuhn (1962) members of a scientific community share a *paradigm*, or set of assumptions about what problems are important, and how one might go about solving them. Members of the community engage in 'normal science' - a problem-solving activity that confirms the shared paradigm. When normal science leads to anomalies, the science in question enters a period of crisis, in which one paradigm may be replaced by another. Kuhn claims that paradigms are 'incommensurable'. Those who adopt different paradigms see a different world, and will not be able to agree about the significance of observed events. From this perspective action research is not weakened by the interpretive nature of its data, because conventional science also relies on an unacknowledged process of interpretation. As will be explained in section 3, the action researcher appears to have a choice between a number of alternative paradigms.

Another criticism of action research derives from concern over the role of the client in the client-researcher relationship. The client may be paying for the research, and seeking considerable control over its course. It may be difficult or embarrassing for the researcher to disagree with or criticise the client, or to publish findings that the client wishes to

suppress. There may be stake-holders in the problem-situation other than the client, and the researcher may wish to take their point of view into account. Berry et al. (1986) report on research in the National Coal Board (NCB) during a period of intense conflict between the union and the management. The researchers wished to publicly disclose that the NCB accounts were an inadequate basis for the justification of plant closure decisions. The response of the NCB was to foreclose further research and to refuse to engage in a review of the researcher's arguments and those of other academics. The researchers took a stand on moral grounds, considering that their duty as academics overrode the need to maintain good relationships with the client. The issue for action research, in this case, is the use of power to deny the application of the insights of research in the interests of the community rather than sectional interest. The management of the NCB, backed by central government, wished their point of view to prevail. Other stakeholders, with different interests, were the union, the miners and the general public. The academics concerned were also stakeholders, with an interest in communicating their interpretation of the situation.

It is clear from the preceding discussion about the role of the client that action research raises ethical questions. Academics must add to knowledge, but action researchers claim to produce practical knowledge rather than knowledge for its own sake. Practical knowledge can be used to take action but action is purposeful. The action researcher is concerned with ends as well as means and must choose whose ends to serve. Decisions about desirable ends can be taken only in the context of a value system. Argyris et al. (1985) stress the importance of *competence* and *justice* both of which they see as deriving from the need to pursue

rationality. The action researcher should seek to enhance human competence - that is to enhance the likelihood of achieving intended consequences. Competent action must be preceded by rational determination of purpose, and justice demands that all points of view be equally taken into account. This value system is essentially democratic. The idea that injustice is irrational is also to be found in the work of Ulrich (1983). Democratic determination of purpose is fraught with difficulty. Stakeholders in a problem situation are not equally powerful, articulate or informed. It has been suggested by proponents of what is known as 'critical theory' (Geuss 1981) that stakeholders may not clearly understand their own true interest, because their ideas have been formed in a coercive society. Habermas (1979) argues that true interests can be defined only in conditions of complete freedom of discussion, and these conditions may be absent.

4. Methodologies for Action Research

A number of methodologies exist for performing action research, for example OR (Churchman et al. 1957, Rivett & Ackoff 1963, Ackoff & Sasieni 1968), socio-technical systems design (Emery & Trist 1960), soft systems methodology (Checkland 1981, Checkland & Scholes 1990), and design methodology (Jones 1980). What these methodologies have in common is that they provide a number of recommended methods and techniques for identifying problems, planning action, taking action and evaluating the results. Where they differ is in the ontological and epistemological assumptions they make. OR, for example, assumes that problems arise as a result of malfunctioning systems. Knowledge about systems behaviour is gained by mathematical modelling and

experimentation with a model. Socio-technical systems design assumes that problems arise because of a mismatch between social and technical systems. Technical systems are those studied by operational researchers, such as stock control, production and distribution systems, and can be optimised by the methods of OR. Social systems, however, are systems of human beings performing roles under the influence of sets of values that lead them to seek particular objectives. Knowledge of social systems will in part be based on interpretation rather than empirical data capture. Soft systems methodology (SSM) does not presume the systemic nature of a problem context, and transfers the notion of systemicity to the process of enquiry. It is intended for contexts characterised by complexity, lack of agreement about purpose, or divergences in world-view. Use of SSM is intended to bring about learning on the part of the participants. Design methodology has its origins in engineering and architecture, but has progressively widened the sphere of application of its methods to systems, including abstract systems. Its distinctive contribution to problem-solving is the stress it places on generating ideas for *new* entities as opposed to the study and optimisation of existing systems. March (1984) suggests that the two conventionally accepted forms of reasoning, induction and deduction, cannot give rise to new entities. Central to design is the hypothesizing of what *may* be as distinct from what *must* be or actually *is*. March names this type of cognitive activity 'abductive' reasoning. Design methodology has been influenced by a 'radical tendency' of designers who reject many aspects of contemporary society and seek to use design methods to re-design society.

5. Radical Design

The methodology used in the research described in this thesis derived from the radical design movement. The term radical design is used in this thesis to describe design methods and approaches that are:

[1] general-purpose, ie not specific to a particular branch of engineering.

[2] multi-level, ie suitable for designing at several different levels of abstraction.

[3] accessible, ie intended for use by lay-people not technical experts.

The distinction between [1] and [2] above is that general-purpose methods are suitable for designing different types of object at the same level of abstraction (e.g. aircraft, bridges or houses), whereas multi-level methods are suitable for designing hierarchies of objects, the higher levels of which are abstractions (e.g. roads, transport systems and economies). An example of a radical approach to design is that taken by Jones (1980). Jones defines design as *the initiation of change in man-made things* and by this definition widens the scope of design from engineering, architecture and fashion to include the activities of planners, managers, politicians and organised bodies concerned to bring about change in society. Jones defines four types of entity amenable to design, *components, products, systems* and *communities*. Traditionally designers have been concerned with products or components. Jones wishes to see systems and communities treated as 'designable' ie under conscious human control. This implies

> the power to continuously remodel the whole fabric of industrial society from top to bottom. (Jones 1980 p. 32).

Jones (1980) describes 35 design methods, including methods of

searching for ideas, methods of exploring problem structure, and methods of evaluation. Jones also provides a framework that suggests which methods are appropriate at various stages of a design project. The first edition of Jones' book was published in 1970 and, as the first Professor of Design at the Open University, his ideas influenced the development of undergraduate courses. The OU course T262, for example, first delivered in 1975, contained a design methodology that drew upon Jones' set of design methods but augmented it, and provided a more explicit contingency framework for choice of method. This framework recommended methods as suitable for either *products* or *systems* (the more ambitious aim of designing communities was not directly addressed). The other element of contingency in selecting a method was whether the project was at the exploration stage, or at the stage of generating ideas, or at the stage of selecting promising ideas. The stages of a design project were envisaged as:

- Problem identification
- Exploration
- Generation of ideas
- Selection of ideas

The methodology was not intended to produce detailed designs for any particular product or system, and its use provides a means of problem structuring. This is because of the intimate relationship between what is perceived as a problem and what is perceived as a solution. Cross (1982a) describes the problem-solving activity of a designer as 'solution-focused'. He argues that designers face ill-defined problems that lack definitive formulation. Any formulation of an ill-defined problem is bound to contain puzzles and inconsistencies. Furthermore, formulations of the

problem are solution-dependent, ie each formulation refers implicitly or explicitly to a solution. In these circumstances a designer who works on a solution is simultaneously defining a problem. Checkland (1981) refers to ill-defined problems as 'soft', and defines a soft problem as one that cannot be solved by choosing suitable means for pre-defined ends. Checkland's SSM concentrates on clarifying ends and postpones decisions about means. Clarification of ends in SSM is a sophisticated social process, involving debate among stakeholders. The T262 methodology does not treat the clarification of ends as a social process but rather as a psychological one. The designer becomes clearer in his or her own mind about the nature of the problem, and does so by generating ideas to solve it.

Another distinctive feature of radical design, and of T262 as an example of it, is the stress on making design methods accessible. Cross (1982b) argues that it is necessary to do more than merely allow users to participate in design. The distinction between user and designer must be eradicated, and knowledge of, and facility in, methods of design spread widely through the community. The reason that Cross seeks this end lies in his attitude to modern technology as pernicious, or having unacceptable side-effects. Such technology, moreover, is unthinkingly accepted as inevitable by users and consumers, due to lack of understanding of the design process as something amenable to social control. The T262 course, of which Cross (the current Professor of Design at the Open University) was co-author, was intended to disseminate knowledge of design methods.

Two issues arise from the application of a radical design approach in action research. Firstly, since design must have as its focus the creation of some new object (albeit an abstraction), a design paradigm will tend to

re-ify human activity. There will perhaps be a greater emphasis on social structure as a determinant of human activity, and less attention paid to intersubjective negotiation depending on interpretation of meaning. Secondly, any approach that is *radical* in intent must have adequate theory that enables coercive forces in society to be overcome. The mission of radical design, to eliminate the distinction between designer and user, may be unachievable without such theory. These issues are discussed in Section 3 of Chapter 6, drawing upon the learning that resulted from the project described in this thesis.

6. Choice of Methodology.

Given the wide variety of possible approaches to action research, the researcher is faced with the problem of choosing between alternatives. Apparently working on the assumption that different methodologies might be good for different purposes Jackson & Keys (1984) categorise the contexts in which problems may be perceived in terms of variations in the complexity of the systems in which the problem is located, and variations in the relationships among the participants in the situation. Simple systems consist of a small number of elements with few or regular interactions. A system may be experienced as complex for a variety of reasons including the large number of its components and their inter-relationships, whether it is difficult to observe the operation of the whole system, whether it is difficult to construct a quantitative model of the system and whether the system changes its structure dynamically. Jackson & Keys distinguish between mechanical problem contexts which contain simple systems manifesting relatively easy problems, and systemic problem contexts which contain complex systems manifesting difficult problems.

The system in which the problem exists is one factor in determining the character of the problem context, and the other is the nature of the relationships among the decision-makers in the system. If the decision-makers agree on a common set of goals for the system and make decisions in accordance with those goals then the problem context is described as unitary. If the decision-makers cannot agree on a common set of goals and make decisions which are in accordance with differing objectives, then the problem context is described as pluralist.

As a result of their analysis Jackson & Keys classify problem contexts into four different types:

[1] mechanical-unitary

[2] systemic-unitary

[3] mechanical-pluralist

[4] systemic-pluralist

They then suggest suitable methodologies for tackling problems in each of the four contexts. The techniques of classical OR are thought to be suitable for *mechanical-unitary* contexts. In *systemic-unitary* contexts the systems of concern have many elements in close inter-relationship and exhibit behaviour which is difficult to predict. There is, however, full agreement about the goals of the system(s). In these circumstances the use of management cybernetics (Beer 1985) is recommended. It seems likely that socio-technical systems design, described earlier, also makes assumptions consistent with systemic-unitary contexts. Jackson & Keys define a *mechanical-pluralist* problem-context as one in which the pluralism concerns differences amongst decision-makers (of whom the problem-solver may be one) *outside* the system, because the component parts of mechanical systems are passive, not purposeful. If the

disagreements among decision-makers can be resolved then the problems remaining can be solved using OR. The approach of Churchman (1979) is recommended for resolving disagreements. In a *systemic-pluralist* context the systems components are purposeful e.g. conscious actors or organised bodies thereof, and are not in agreement as to the goals to be pursued. In this context Jackson & Keys recommend a soft-systems approach, for example use of SSM.

7. Methodology Choice in a Coercive Context

Jackson (1987a) and Keys (1988) have separately elaborated the contingency theory for methodology selection, imparting a different orientation in each case. Jackson extends the contingency framework by redefining pluralist contexts as those in which the participants have to some extent differing objectives, but a genuine compromise can be reached upon which all agree (because their fundamental interests are not irreconcilable). He then identifies a new type of relationship that may exist between participants in a problem-context - a coercive relationship, in which 'any consensus that exists is only achieved through the exercise of power and by domination (overt or more or less concealed) of one or more groups of participants over others'. According to the logic of the Jackson & Keys framework, coercive situations may be either mechanical or systemic. Jackson tentatively assigns Ulrich's Critical Systems Heuristics (Ulrich 1983) to the mechanical-coercive category and suggests 'an approach based upon radical-structuralism is more apt in systemic-coercive contexts.'

Jackson (1982) has rejected SSM for use in coercive contexts because it provides no way of equalising inequalities in intellectual and linguistic

resources and power imbalances among the stakeholders. The result of these inequalities may result in use of SSM reproducing the existing social order rather than facilitating radical action. Jackson accuses SSM of being culpably 'regulative' because it is prepared to accept for implementation changes emerging from a false consensus produced by distorted communication.

> 'To have any claim to neutrality the methodology would have to incorporate a prior commitment to establishing the condition for unconstrained discussion' and 'challenge those social arrangements which produce distorted communication.' (Jackson 1982).

Checkland's reply (Checkland 1982) to Jackson proceeds as follows. Jackson tries to establish his case by argument alone, without appeal to real-world evidence of a testable kind. SSM was developed by experience in the world. Jackson takes as given an objective social reality characterised by structures that put constraints on groups. He assumes that there is a valid distinction to be made between a true and a false consensus. Jackson's requirement that a prior commitment be made to equalise the intellectual and power resources of individuals that wish to use SSM would ensure that the methodology could never be used. In principle use of SSM could be either:

> 'conservative/regulatory or radical/emancipatory depending upon the readiness to modify *Weltanschauungen* in the particular situation in which the methodology is used.'

In practice Checkland concedes that

> 'defining changes which are "culturally feasible" has lead to rather conservative use of the methodology.'

The essence of the disagreement between Checkland and Jackson lies in the issue of whether or not *Weltanschauungen* can be changed. Checkland believes that they have been changed by use of SSM. Jackson agrees that they could in principle be changed but believes that ideas derive from society -

> 'The ideas which flourish are likely to be those which support the dominant political and economic groups in the system.' Moreover 'the ideas of some participants in social systems may be ideological and may conceal the "real" nature of the social organisation.'

Jackson concludes therefore that use of SSM is unlikely to bring about radical change in coercive situations. Jackson denies that changing men's ideas is liable to be in general fruitful in bringing about radical change -

> 'political action rather than action at the level of ideas may be the best means of removing the major structural barriers lying in the way of communicative competence.'

Willmott (1989) reviews the debate between Jackson and Checkland and finds weaknesses in both positions. In the case of SSM his judgement is that it 'simply lacks a *social theory* capable of accounting for why particular sets of perceptions of reality emerge, and why some perceptions are found to be more plausible than others.' Willmott finds Jackson's presentation of his case inadequate, however, because 'it lacks any justification for privileging the radical paradigm of analysis'.

Jackson & Key's framework has been applied in a number of action research projects (Carter et al. 1987, Jackson 1987c, Keys 1987), and has influenced the work of a number of researchers (Banathy 1988, Flood 1989, Oliga 1988). Unresolved issues and areas for further research associated with the framework are:

[1] The location of methodologies *not* examined by Jackson & Keys (e.g. design methodology); in particular the discovery or development of approaches suitable for coercive contexts.

[2] Doubt about the appropriateness of location of methodologies that *are* examined by Jackson & Keys.

[3] Doubt about the philosophical basis of the framework e.g. whether the nature of a problem context can be objectively established, or whether the categories are ideal-types.

[4] The problem of paradigm incommensurability.

This last problem derives from questioning the rationality of choosing among methodologies deriving from alternative paradigms that make contradictory assumptions. Resolution of this problem resolves the doubt expressed in [3] above. Further discussion of these issues will be found in Section 4 of Chapter 6.

CHAPTER 3 - THE PRODUCTION & USE OF SOFTWARE

1. Introduction

This thesis analyses the problem of software usability. In this chapter the term 'software' will be defined and the processes by which it is produced and consumed will be described. Software is an engineered product that is sold to organisations to enable them to construct computer-based information systems. The production process is known as 'software engineering' and the process that consumes the product within organisations is known as 'information systems development'. The concept of 'usability' is vague but is associated with notions of fitness for purpose and ease of use. Long (1986), for example, states that a system must be usable:

[1] to perform specific tasks;

[2] by specific users; and

[3] in a specific physical and social environment.

Study of the literature, reveals a number of concerns about the software engineering and information systems development processes. These concerns are reported in this chapter. The purpose of this chapter is to provide a context for the specific problem notified to the researchers in this project. The client in this case was a computer manufacturer that provided software with its machine ranges.

2. Software

Computers are general-purpose machines that perform specific functions only when loaded with a program of instructions. Many different programs may be available for a particular type of computer. The behaviour of the computer varies depending on the program currently executing. The term 'software' is sometimes used to refer to computer programs in general, in contrast to 'hardware' - the physical machinery. Computer programs are not physical entities although they may be encoded on a physical medium such as magnetic disc or semi-conductor memory. Computer programs are abstract structures expressed as coded statements that can activate a defined machine. The term 'software' also has a more specific meaning, when used to refer to programs available for sale (or available free in the case of public domain software), as distinct from programs produced by a single computer user for private use. Pressman (1987), for example, categorizes software as follows:

SYSTEMS SOFTWARE - for example compilers, editors, operating systems, file management utilities, and telecommunication monitors.

REAL-TIME SOFTWARE - for example software that monitors and controls a system in its environment by responding rapidly to external events with control signals.

BUSINESS SOFTWARE - for example applications packages that process business transactions and support common business operations with information.

ENGINEERING AND SCIENTIFIC SOFTWARE - for example software that performs numerical analysis, simulates the operation of systems or provides computer-aided design facilities.

EMBEDDED SOFTWARE - for example instructions encoded in read-

only memory that replace conventional electronics within larger systems.

PERSONAL COMPUTER SOFTWARE - software that could include items from any other category, but designed to run on a low-cost machine for a particular market-place.

ARTIFICIAL INTELLIGENCE SOFTWARE - for example expert system shells.

This categorisation defines a software market-place and represents programs that the purchaser of a computer might expect to be available with the machine, thus obviating the need to produce them by in-house programming. Macro & Buxton (1987), taking the point of view of a provider of programming services, distinguish between software *projects*, which produce programs for a single identified end-client, and software *products*, which are programs written for a multiplicity of, as yet, unsecured (and maybe unknown) clients. The problem of software usability presented by the computer manufacturer in this research was a problem to do with software products.

The main business interest of a computer manufacturer is in the manufacture and supply of computer equipment. Since computers are useless without programs to direct their operation, however, computer manufacturers also supply software products with their machines. All computer manufacturers supply systems software and may supply other categories of software depending on their view of the market for their equipment. A major computer manufacturer, such as IBM or ICL, attempts to be represented in as many markets as possible, and will produce software in all of Pressman's categories. Computer manufacturers produce software *products* and do not typically involve themselves in software *projects* as a major line of business. Software products are

general-purpose - that is the manufacturer intends them to be used by as many customers as possible, in order to spread the development costs widely. Software projects are initiated by the users of computer equipment and they may be performed entirely in-house or by establishing contracts with specialist software houses.

A computer manufacturer's customers will always use *some* standard software products, for example the systems software. In other cases the customer has a choice between using the manufacturer's software and initiating a software project. This would be true, for example, of business software. In some cases software products may be available from third-party software vendors that rivals the manufacturers native products. A customer, therefore, assesses the software products available for the machines it uses, and decides whether to acquire them and/or to initiate software projects. A computer manufacturer experiences a marketing failure every time a customer decides not to use one of its products, and purchases a rival product, or initiates a software project to provide the facilities required. A customer decides whether to buy software products or to develop its own applications on the grounds of cost and utility. Software products are liable to be cheaper than in-house development because the software vendor can spread its development costs over the market. The marginal cost of producing one more copy of a software product is low. General-purpose products, however, may contain some facilities that are unwanted by any particular prospective user and yet be lacking in others. The functionality of a software product is a compromise between the possibly conflicting demands of customers. In-house development of software may be preferable to using a product ill-suited to user requirements.

3. Software Engineering

Software engineering is a relatively junior engineering discipline. Software products first became available in the 1950's and software reliability has remained a problem since then. Typically, early versions of a software product contain many design and implementation errors and a customer may have to wait years before the product is error-free. Software maintenance represents a considerable cost to software vendors and the detection and correction of errors may greatly inconvenience users. Software errors are known in the industry as 'bugs', as if to imply an external agent causing the software to malfunction. In fact the 'bugs' are introduced by failure in design and implementation. In the late 1960's practitioners and users started referring to a 'software crisis' (Sommerville 1982). Research to address the crisis has taken the form of developing techniques for coping with systems complexity, managing cooperating groups of designers and computer programmers, and measuring the quality of software (Lamb 1988). Determination of user requirements remains a largely unsolved problem.

Software engineering consists of a, possibly repeated, cycle of four phases:

[1] Requirements Analysis.

[2] Systems Design.

[3] Systems Implementation.

[4] Systems Maintenance.

Each phase can be broken down into further sub-phases. Descriptions of this 'software life-cycle' in the literature assume a software *project* with an identified end-client. For example Cohen, Harwood & Jackson (1986)

describe a 'contractual model' found useful in Standard Telecommunication Laboratories Ltd. (a member of the STC plc group). Here each phase of software development is regarded as the subject of a contract between two parties, called the customer and the supplier. The completion of each phase is signalled by the customer acknowledging that the item delivered to him by the supplier satisfies the terms of the contract between them. The existence of a customer simplifies the elicitation of user requirements and the validation of designs and implementations. The process starts with receipt of a customer's statement of requirements:

> 'a class of document infamously incomplete, ambiguous, inconsistent and generally unsatisfactory.' (Cohen et al. 1986).

Requirements analysis consists of removing these defects by respecifying requirements in a formal (mathematical) language. Once a formal specification has been created there is no further uncertainty in software design and implementation. Proofs of correctness are constructed for each mapping from specification to design to implementation. The account of Cohen et al. represents the most advanced thinking in software engineering, and yet it gives only limited assistance to the producers of software products, who do not have captive customers. Harker & Eason (1984) point out that establishing requirements for software products is problematic because of the distance of the supplier in time and space from its customers. The distance in time arises because the products are to meet needs some time in the future. Information technology creates new ways of working and possibly new types of job. A software producer cannot expect to respond passively to clearly expressed user requirements, but must in some cases create requirements by innovative design. The distance in space arises because software products are intended for a mass market. The software producer cannot negotiate requirements in such a way as to

produce an unambiguous specification with many hundreds or thousands of customers. Eason and Harker (1988) discuss the options open to software vendors hoping to develop successful software products. These include the following:

[1] Building formalised models of generic users to be used as a frame of reference for the design team.

[2] Employing people from user organisations.

[3] Bringing typical users into the supplier organisation on a temporary basis.

[4] Forging a relationship with a 'favoured customer'.

[5] Studying appropriate user groups within the supplier organisation.

None of these options is a completely acceptable alternative to producing software for a client on a contractual basis.

The specification and implementation characteristics of software, the so-called S-type, P-type or E-type features defined by Lehman (1980), also influence the likelihood of securely establishing requirements. In S-type (specifiable) systems the requirement can be precisely specified, is invariant with time, and a provable implementation can be achieved. Such systems are small and within the capabilities of a single person to achieve within a reasonable timescale. Much computer science research into formal specification of requirements is implicitly concerned with S-type systems. In the case of P-type (programmable) systems a complete and precise specification of requirements can be given, but a provably correct implementation cannot be derived from them. E-type (evolutionary) systems are those whose requirements will change with time, either because the environment of the system changes exogenously or because the system

itself changes its environment. Specifications of E-type systems rapidly become obsolete, and new versions need to be continually produced. The great majority of software products are E-type systems.

4. Information Systems Development in Organisations

Organisations rely on information systems (IS) to co-ordinate and control their activities (see for example the account of Davis & Olsen, 1985). Examples of IS are accounting systems, production planning and control systems, and sales analysis and forecasting systems. This approach to categorising IS concentrates on the support they provide to organisational *functions*. Another approach to categorising IS concentrates on the support they provide to *hierarchical levels* in an organisation. Using this principle, examples of IS would be transaction processing systems, management information systems and decision support systems. IS can be analysed and designed at a level of abstraction above the technology used to implement them. This principle has characterised the work of Langefors (1973). Langefors defines an information system as a system for the collection, storage, processing and distribution of information. The information system is part of a wider system known as the object system. The information stored and processed in an information system derives from the activities of the object system, and may be used to co-ordinate and control those activities. Langefors describes a model of an information system as *infological* if the model is free from implementation considerations. Information systems may be constructed by mapping an infological model to a *datalogical* model that presumes a certain type of technology to support the processing. Langefors develops a model of required data-flow in an information system by means of *precedence analysis* - a

process that defines the output requirements of the system, and reasons about the logical precedents of the output data in terms of stored data, input data and intermediate processes. Langefors' ideas have influenced the information systems development methodology known as ISAC (Lundeberg, Goldkuhl and Nilsson 1981). ISAC models an object system as a network of functions consuming input and producing output, then selects those functions that transform data and produces an infological model by precedence analysis. The infological model is then mapped to a datalogical model.

Ideas similar to those of Langefors are to be found in the work of Ross (1977) who describes a Structured Analysis Language that models systems by top-down decomposition of function, and shows the inputs to, and outputs from, these functions. These ideas are embodied in the information systems development methodology SADT, described by Ross & Schoman (1977). Structured Analysis creates the equivalent of Langefors' infological model. The data-flow diagrams produced by Structured Analysis may be converted to executable computer-based systems by the techniques of Structured Design described by Yourdon & Constantine (1979). Integrated accounts of structured analysis and design of information systems are provided by Gane and Sarson (1979) and De Marco (1979).

Distinction between infological and datalogical considerations is also important in modelling the structure of data, as distinct from the processes that operate on it. Some authorities argue (Howe, 1983, Avison, 1985) that data analysis should precede functional analysis because it may be possible to share data among applications and thus avoid data duplication. Data models, for example based on the relational model of Codd

(1970) or the entity/attribute/relationship model of Chen (1976), are implementation-free models of the structure of the data required by organisational IS.

It is during the process of mapping from an infological model of an IS to a datalogical model that use of software products will be considered. For example to map a data model to a database design requires knowledge of the database management system (DBMS) to be used. A DBMS is a software product that may be purchased from a computer manufacturer or a third-party software vendor. It may be possible to implement some of the functions shown on a data-flow diagram by applications package. This decision must be made before further design work continues that results in the specification of programs to be developed in-house. In general, software products form ready-made sub-systems that may be used to implement all or part of IS requirements. An organisation necessarily chooses to use certain software products when it chooses a hardware supplier. This is because computers are useless without systems software, and organisations very rarely attempt to develop their own.

The process of IS development in organisations has been criticised for its neglect of human factors and social issues, for example by Lyytinen (1987):

> 'The IS community faces a paradox: despite impressive advances in technology, problems are more abundant than solutions: organisations experience rising costs instead of cost reduction, IS misuse and rejection are more frequent than acceptance and use.'

Bjorn-Anderson (1988) diagnoses failures in workplace ergonomics, organisational ergonomics and societal ergonomics. 'Workplace ergonomics' is concerned with problems of fatigue and discomfort in using information technology. 'Organisational ergonomics' is concerned with the organisational purpose of information systems, and the necessity of

providing satisfying jobs with adequate professional content and opportunities for social contact. 'Societal ergonomics' is concerned with pursuing desirable social objectives, such as full employment. Failures in each of these areas has resulted from

> 'the narrow engineering approach to the area of human/machine communication, the instrumental approach to communication and its purpose, and finally the naive assumption that our technology is neutral.' (Bjorn-Anderson 1988).

Lyytinen focuses on the inadequacy of information systems development methodologies, particularly those described as *technical* design-oriented methodologies. Floyd and Keil (1983) describe the *problem of reduction* in such methodologies and point out the following dangers:

[1] Equating people with things (people as data sources or receivers) or with computer programs (people as information processors).

[2] Reducing objects to data about the objects.

[3] Reducing goal-oriented action carried out by people to the processing of symbolic information.

A number of methodologies exist that promise to deliver more effective information systems, for example ETHICS from Mumford (1983), an adapted version of Checkland's SSM from Wilson (1984), and MULTIVIEW from Wood-Harper, Antill & Avison (1985). Mumford stresses the socio-technical rather than narrowly technical nature of information systems, and urges user participation in systems design. Use of SSM encourages the exploration of issues from many points of view prior to the selection of a primary task to be supported by an information system. MULTIVIEW eclectically combines techniques from SSM, socio-technical systems design, data analysis and functional analysis. None of these approaches is widely used in practice, and the prevailing paradigm in organisational information systems development is technical design.

5. Software Production and Consumption

Software production and consumption is a system beset with problems. The software engineering paradigm necessitates the unambiguous specification of user requirements. Software vendors find it difficult to establish user requirements. Software design cannot proceed without some model of requirements, so software producers use strategies like designing a product for a 'favoured customer' or gaining access to 'typical' users. These strategies do not guarantee a comprehensive definition of user requirements. Software products often contain many errors when released. The mapping of designs from formal specifications may reduce programming errors. Not all software producers yet use formal specification, however, perhaps because of a shortage of skilled practitioners. Although formally specified software is amenable to proof of correctness, it is still possible to introduce errors into the specification, particularly when converting a vaguely specified user requirement into the language of mathematics. Some aspects of a software product are very difficult to specify formally, for example the user interface. Successful software products have typically been through many iterations of a cycle of release, use, feedback of criticisms and error reports, and re-release. There is little evidence that this situation will change. Since most software products are E-type systems, the possibility of eliminating the maintenance phase of the software life-cycle seems remote.

The nature of IS development in organisations also poses problems. The infological modelling that is conducted is at a level of abstraction above implementation, and this causes human characteristics and requirements to be ignored. Decisions to automate work systems are taken on the grounds of cost and efficiency, not on whether jobs of adequate quality

and quantity should exist. People are often treated as a means of implementing information systems (ie as information processors), rather than as clients to be served by them.

--

CHAPTER 4 - THE USTM PROJECT

1. Background

A research project entitled 'User Skills and Task Match (Methodologies for Matching IT Products to User Needs)' was approved for funding by the Alvey Directorate towards the end of 1985. The collaborators in the project were a computer manufacturer (who provided the project manager), and Huddersfield Polytechnic. The Polytechnic was represented by three academics, a computer scientist, a psychologist and Mansell, supposedly a specialist in methodology. The proposed research was stated in the submission document to have three main objectives:

[1] To provide a means of producing more valid and rigorous software product specifications.

[2] To enable user's requirements to be more fully incorporated into the product design process.

[3] To improve communication between marketeers and designers.

The product of the research was envisaged as a design methodology.

The background to the research proposal was as follows. At the time, Government policy towards information technology research in the United Kingdom was to encourage academic collaboration with industry and to channel funding through the Alvey Directorate. Alvey-approved projects necessarily involved industrial collaboration. Alvey categorised research into the areas of Intelligent Knowledge Based Systems (IKBS), Software Engineering, Very Large Scale Integration (VLSI), and Man-

Machine Interface (MMI). The USTM proposal was accepted as an MMI project.

The manager of the USTM project had been appointed by the computer manufacturer to oversee MMI initiatives within the software producing division of the company. The USTM project was only one of his concerns, therefore, and his brief overall was to raise the profile of human factors issues within software production. The computer manufacturer was aware that software usability was becoming an issue in the industry. The initiative in seeking Alvey funding for the project lay with the project manager, and the academics were in a sense 'recruited' by him to help in implementing his ideas. The motives of the project manager appeared to be:

[1] To be be seen to be taking vigorous action in the discharging of new responsibilities.

[2] To gain another Alvey award for his company (such awards were prestigious).

The project manager characterised the problem to be investigated as follows. Designers lacked information about the prospective users of software and the tasks that they performed. This could give rise to products that were difficult to use, but also to products that exhibited inappropriate functionality. In conditions of uncertainty designers tended to provide too many functions rather than too few, and this complexity might worsen usability, as well as being uneconomic from the vendor's point of view. The project manager considered that the problem was in part caused by a communication gap between marketeers and designers. Marketeers were familiar with the market for software products, and to some extent understood user and task characteristics. Their software

'specifications' were informal, however, and did not provide sufficient guidance to software designers. The designers were good technicians who lacked market awareness and often operated under conditions of uncertainty about the ultimate use of the products they were designing. As pointed out in Chapter 3 software vendors in general have the problem of not being able to thoroughly analyse user and task characteristics prior to design, however competent the marketing staff. This is because software products must be sold to many end-user organisations, and the characteristics of each one cannot be analysed in detail. This is in contrast to in-house systems development where the users are 'captive'. Existing software engineering methodologies assume that user requirements can be accurately captured. A software vendor hoping to sell to a wide market cannot be totally confident that requirements are known prior to design of the product.

The analysis of the problem was provided by the project manager and so was the solution. This was to take the form of a methodology. The Alvey proposal implied that this methodology would be used by marketeers to improve their software requirements specifications and thus reduce designer uncertainty. In practice this is not how the methodology came to be used. The physical form of the methodology was specified as a training manual, to be supplemented by training courses. The structure of the proposed methodology was described as follows in the Alvey proposal. It was to have five distinct 'levels' - the *market level*, the *user level*, the *object level* the *task level* and the *dialogue level*. The term 'level' seemed to correspond to the idea of a phase or step, in that it was implied that the user of the methodology would complete tasks at the market level before starting on the user level and so forth. Iteration

around the levels was anticipated, however. The term 'phase' was used with a different meaning, however, in the description of the methodology. The methodology was described as having three 'phases' - *Market Review, Product Requirements and Outline Design,* and *Product Development.* To complete the Market Review phase the user would proceed through the five levels, and then proceed through them again to complete the Outline Design phase. So the methodology consisted initially of five phases repeated twice as a block of five. Each block of five, however, might involve iteration before it could be regarded as finished. The Product Development Phase consisted of prototyping a product, and the details of the phase were not to be addressed in the Alvey funded research (neither was the Dialogue level). The conceptual architecture of the methodology did not survive long after the start of the research. In a sense it was a facade constructed to impress the grant-awarding body. It was a professional piece of 'methodology-speak' that included references to levels, phases, iteration, users, objects, tasks and dialogue, but meant very little. In terms of action research it represented a commitment to action with no clear theory informing the action, or enabling sense to be made of the problem situation. The proposal was accepted, however.

The proposal stated that the methodology would be evaluated by testing it in two of the computer manufacturer's market areas. It is indeed difficult to imagine how a software requirements specification methodology could be evaluated in any other way but by trying it out. Even if the methodology had been 'successfully' applied, there might well be uncertainty about whether the success was due to the inherent worth of the methodology, or chance variations in the skills of the practitioners, or chance developments in the market-place for the product. Nevertheless, as

it turned out the methodology was not evaluated in this way at all.

After acceptance of the research proposal by Alvey two of the academics, the computer scientist and the psychologist, were seconded to the computer manufacturer to prepare the ground for the project and familiarise themselves with company procedures. During this secondment the project took a turn that caused it to deviate sharply from the proposal. At the instigation of the project manager a series of three-day workshops were run called *Developing a Product Opportunity* (DPO). These were attended by company staff. A typical workshop consisted of two teams each comprising a marketeer, designers and a technical author. Each team was responsible for developing a particular software product. Lectures on the five levels were given to them, and team members worked on tasks associated with each level. The DPO workshop programme seemed premature because an untested methodology was being taught to company staff. From the point of view of the participants, however, there were benefits to be gained by withdrawing from a pressurised work environment and spending time discussing their product. To some extent it did not matter that details of the methodology were obscure, unclear, arbitrary and lacking in theoretical justification. The 'levels' provided a framework for group discussion.

Plans were made to evaluate the project in its present form. The methodology was to be evaluated in three ways:

[1] An evaluation of the workshops.

[2] Evaluation by case-study.

[3] Evaluation by historical analysis.

The rationale for evaluation by workshop was that designers and marketeers were educated and skilled people, and if they thought the content

of the workshop was satisfactory then it had survived an important test. An end-of-workshop questionnaire was, therefore, provided. Hutt et al. (1987) report a favourable reaction to this evaluation, in particular the participants liked the team-building nature of the exercise. The acid test, however, was whether they subsequently used the methodology in their work, and whether it gave rise to better specifications. Note that the methodology was now being taught to marketeers, designers and technical authors, not just marketeers. It could be the case that software products will improve simply by causing marketeers and design staff to come together periodically for group discussions. In other words the specific content of the USTM methodology might be irrelevant.

Evaluation by workshop could not conclusively prove the worth of the USTM methodology. Such evaluation could be provided only by attempting to use the methodology to specify the requirements for a piece of software. This was planned to take place but, due to the exigencies of organisational life, never did. Thus the project failed in its most important piece of validation. It seems extraordinary that the authors of a methodology could teach its use to others without having used it themselves, with varying degrees of success under varying circumstances, the methodology gradually being refined in use. Hutt et al. (1987) compare USTM with Checkland's SSM and Harker & Eason's (1985) Open-Systems Approach, in that 'all three are concerned with getting the requirements right'. This remark reveals hubris.

Evaluation of the methodology by historical analysis was to take the following form. An existing product was to be chosen. Its dysfunction was to be predicted by comparing its actual specification with the specification that would have been produced if USTM had been used. Finally

its actual dysfunction would be established by field research and compared with the predicted dysfunction. This is a difficult but not unacceptable way of proceeding, and needs many repetitions to guarantee validity. It is no substitute for using the methodology and refining it in use. It was stated earlier that the conceptual architecture of the methodology did not survive. DPO corresponded to this architecture, and was the equivalent of the Market Review phase. The equivalent of the Product Requirements and Outline Design phase was retitled *High Value Solution (HVS)*, and the equivalent of the Product Development phase was retitled *Completing Product Requirements (CPR)*. The five levels were abandoned in specifying the latter two phases. Changes to a methodology are a sign of health if these derive from practical use, and learning from use. In this case the revised phases were to be developed in the same way as DPO - by workshop. The essence of the work to be done in HVS was as follows. For each high level task to be performed in some market area, a partitioning was effected between man and computer according to specified principles. The composition of a set of automated tasks comprised a software product. CPR was to be concerned with planning the implementation and delivery of the product.

2. Problems Associated with the Project

After a year's work on the USTM project Mansell was experiencing considerable frustration with it. The burden of his criticism was as follows:

[1] A methodology for product specification is being developed not by using it and refining it as a result of reflection on use, but by teaching it to prospective users. Some improvement of the methodology

results from this, but it seems an unsatisfactory way of proceeding.

[2] The content of the methodology seems arbitrary and garbled in places.

[3] Validation of the methodology is not taken sufficiently seriously.

[4] The project is subject to 'industrial' project management with tight time schedules, and no opportunity for thought or reflection.

[5] Academics are being used as 'trainers' in the delivery of course material, rather than contributors of ideas. The role of the academic seems to be to add legitimacy to the material - to give the impression that the courses derive from academic research.

[6] There has been no real analysis of the problem situation, and yet rapid moves are being made in delivering a problem 'solution'.

Mansell was particularly concerned with clarifying the nature of the problem and finding out more about the problem situation. The remainder of his time on the project was, therefore, spent in problem structuring. An issue that had to be faced was that of selection of methodology for performing this work. Guidance was sought in the work of Checkland (1981) and Jackson (1987a).

3. Categorisation of Problem

Checkland's research has been concerned with problems defined as follows:

> 'A problem relating to real-world manifestations of human activity systems is a condition characterised by a sense of mismatch, which eludes precise definition, between what is perceived to be actuality and what is perceived might become actuality.' (Checkland 1981).

Underlying this definition is a distinction between laboratory problems and real-world problems, and between hard problems and soft problems.

A laboratory problem is of the type worked on by natural scientists, and is defined by the investigator. A real-world problem is suffered by people in the world, and may be perceived differently by different people. The investigator cannot control the definition of the problem. Real-world problems raise the issue of subjectivity in definition. A hard problem is posed as a need to define suitable means for an agreed end. A problem is described as soft if there is confusion or disagreement over desirable ends. The issue of subjectivity in problem definition raised by Checkland has caused a shift of emphasis in SSM from 'problems' to 'problem situations', defined as:

> 'a nexus of real-world events and ideas which at least one person perceives as problematic.' (Checkland 1981).

The issue of subjectivity raises the question of whose viewpoints should be taken into account during problem analysis. One way of answering this question is to pre-define roles considered important in problem situations, and consider the situation from the point of view of these roles. SSM gives prominence to the roles of *problem-owner* and *problem-solver*.

> Checkland defines a problem-owner as
> 'The person or persons taken by an investigator to be those likely to gain most from achieved improvement in a problem situation.' (Checkland 1981).

Potential problem-owners in the situation studied in this research are software marketeers, software designers and software users. The concern of the marketeers is the development of a portfolio of attractive, profitable products. The concern of the designers is clarity in requirements specification so that professional skills can be employed in producing effective and economical solutions. Marketeers and designers come from different cultures. Marketeers have a superficial grasp of technicalities and strongly developed social skills, whereas designers are technicians

with an orientation towards technological virtuosity rather than human requirements. Both marketeers and designers have a common motive in desiring the commercial success of the software products with which they are associated. Software users are not a homogenous group. Direct end-users have to make use of software products during their work. Frequent users may have different requirements from casual users. Indirect users of software rely on intermediaries to operate the software and provide the results. Other interested parties in user organisations are those who manage staff who make use of information technology, and those responsible for recommending purchase of I.T. products. There is a conflict of interest between software vendors and their customers that derives from the nature of commercial activity - one party is trying to make money out of the other. The commercial relationship is not transient, however, because a software product may be in use for a number of years and may be enhanced during this period. The customer may have a maintenance contract with the vendor, and may be able to put pressure on the vendor because of the customer's economic power, or via user groups. The interests of all the employees of a user organisation are not uniform. Managers strive for productivity and lowering of costs whereas direct end-users need to have jobs that can be performed without undue stress or difficulty. Harker and Eason (1984) state that user problems with information technology are common, and report on software products that use confusing terminology, function according to complex and rigid rules, and do not support user tasks well or require that tasks be performed in some unnatural way.

In summary, the problem situation contains problem-owners who do not share a uniform point of view, and whose interests may be in

conflict. These factors are sufficient for the problem to be deemed soft rather than hard. There is further reason, however, deriving from the motivations of the would-be problem-solvers. The project was a collaboration between an industrial and an academic partner. The problem-solvers were not a homogenous body with respect to their goals and world-view. In principle, commercial and industrial firms are concerned with profit, and academic institutions are concerned with learning. The two opposed motives can result in alternative and conflicting views of what to take to be a relevant system. A further complicating factor is the coercive element in the problem context.

4. Degree of Consensus in the Problem Context

Jackson (1987a) considers that the relationship between participants greatly affects the character of a problem-context. The critical dimension of the relationship is the degree of agreement that exists over objectives. This can range from complete agreement (a unitary context), through some measure of disagreement that is nevertheless rèconcilable (a pluralist context), to contexts where power is exercised by some participants over others (coercive contexts). The exercise of power may be overt or more or less concealed, involving deception or distortion of the truth. Jackson (1987b) states that:

> 'the exercise of power in the social process can prevent the open and free discussion necessary for the success of interaction'

and refers to the 'emancipatory interest' that human beings have in freeing themselves from the constraints imposed by power relations (in addition to their technical interest and practical interest).

Two aspects of the problem-context were considered to be coercive. Firstly, employers may use information technology to reduce dependence

on their employees and software vendors collaborate in this process. Changes to working practices and use of technology are often forced on employees. Secondly the relationship between the problem-solvers in the project was coercive because of the industrial collaborator's control over the project. Academic values were under threat from commercial values.

The research of Child (1987) is evidence of the coercive nature of the introduction of information technology. Child claims that the following objectives feature prominently in managerial intentions when introducing new technology:

[1] Reducing operating costs and improving efficiency.

[2] Increasing flexibility.

[3] Raising the quality and consistency of production.

[4] Increasing control over operations.

Effects on the labour force include reductions in manpower, breaking down of traditional task boundaries, and subjection to greater degrees of monitoring and control. Child highlights the degradation of jobs that is made possible by the application of information technology and states that:

> 'Of all the developments discussed in this paper, the degradation of jobs can be the most confidently identified as a managerial strategy - it has a long history, has been widely discussed and practised, and for many years found a place in managerial, engineering and even personnel literature (though never without its critics).'

By job degradation Child means reducing the skill required to do a job and increasing managerial control over task performance. He gives examples in the areas of numerical control of machine tools, newspaper production, electronic point-of-sale systems and cash dispensing systems. Evidence that attempts to introduce new technology are coercive and generate conflict are provided by Willcocks and Mason (1987). The following

cases illustrate this point.

In 1982 the Department of Health and Social Security devised a 15 year 'operational strategy' to computerise the recording, assessment, calculation and payment of all UK social security and welfare benefits. The major objective was a £700 million saving on operating costs by 1995 resulting from the displacement of 2500 employees. A seven month strike took place at the Newcastle computer centre in 1984 and incurred costs of £150 million.

The National Coal Board introduced the Mine Operating System (MINOS) in selected collieries from the early 1980's. It provided centralised computer control and detailed monitoring of colliery activities. Pits without the system were deemed uneconomic and 41000 redundancies were achieved between 1981 and 1984. The miner's strike over pit closures and redundancies was the most bitter industrial relations conflict of the 1980's.

The newspaper industry has been revolutionised in the 1980's by the introduction of computer systems that permit direct input and composition of text by journalists, editors and tele-add staff. The traditional job of typesetting has been eliminated. When News International moved its newspaper production from Fleet Street to Wapping 4500 workers were dismissed and bitter conflict ensued.

The other aspect of coercion relevant to this research derives from the fact that the problem-solvers were 'managed' in their work by a representative of the computer manufacturer. All Alvey-funded research took the form of collaboration between academics and industrialists. Academics were not free therefore, to choose areas of research unconstrained by commercial considerations. Alvey submissions were constrained not only by the requirement to link academics with industrialists, but also by the themes defined as suitable for funded research. These themes were chosen because they were seen by government as relevant to British competitiveness in the world information technology market. HCI was seen as an 'enabling technology' that would give rise to better products and greater commercial success. The USTM methodology was funded as an example of collaborative HCI research. The general Alvey framework was coercive in the sense that academic freedom was constrained, but a greater degree of coercion resulted from subjecting research to direction by commercial interests. The project manager in this

research treated the academics as employees whose function was to design and deliver training courses for company staff. Commercial firms cannot be expected to sponsor and direct unbiased research. In this case it was inevitable that a corporate point of view would be taken that assumed that the problem-owner was the computer manufacturer. More fundamental, however, was a lack of understanding of what counted as research. Action research necessarily takes place in the real-world where academic values are perhaps not understood or admired. An action researcher has to face this issue and make appropriate accommodation. It is essential, however, that academics retain control over what counts as research otherwise truth becomes the servant of sectional interest.

5. Choice of Methodology

As explained in Chapter 2, little guidance is provided by Jackson in how to proceed in coercive situations, although his work alerts the action researcher to the possibility of their existence. In this project Mansell decided to use a radical design methodology, partly because of its emancipatory philosophy, and partly to test the validity of Jackson's ideas and his understanding of them. The methodology was used after a year of frustration with the USTM project, at a time when pressure from the coercive nature of the situation was intense. Use of a design methodology in structuring soft problems raises the issue of the ontological status of human activity systems. A designer conceives of new objects or systems. Whether human activity can be 'designed' in the same way that cars or transport systems can, is a contentious matter. This issue will be discussed in Chapter 6. The next Chapter describes the application of the chosen methodology.

CHAPTER 5 - USE OF DESIGN METHODOLOGY

1. Introduction

The OU design methodology devised for course T262 (Jacques & Talbot 1975, Cross & Roy 1975) was used in the course of this research. Chapter 4 of this thesis has explained the background to the choice of methodology. Firstly, the problem situation was perceived as 'soft', due to the multiplicity of possible problem-owners and diversity of world-views. The 'obvious' choice of methodology was use of SSM, therefore. Secondly. however, the problem situation was perceived by Mansell to be coercive, and the research of Jackson (or from Checkland's point of view the unsupported assertion of Jackson) indicated that SSM was not suitable for coercive situations. It is Mansell's belief that the project manager in this case would not have tolerated use of SSM because the project plan had been made and the desired solution specified prior to the start of the project. Doubts about the solution and the course of the project were vigorously brushed aside by the project manager. Little guidance on how to proceed in coercive situations was available from the literature, however.

Use of the T262 methodology might appear quixotic. Despite its origins in the work of a generation of radical designers it has no recent history of success in tackling soft problems in coercive situations. Moreover it is open to the criticism of naivety about the ontology and epistemology of human activity systems. It could be argued that any design

methodology, if applied to human activity, will make the naive assumption that human activity systems exist as designable entities. This naivety has been dispelled by Checkland in his replacement of a hard with a soft systems approach. Nevertheless it seemed possible that application of a solution focus to a problem situation (the distinctive contribution of the design paradigm in contrast to the interpretive approach of SSM) might yield valuable clarification, particularly when the constraints placed on the project by the client were removed. In this case, however, there are limits to the rational case for choice of methodology. T262 was applied partly in desperation to allow Mansell to clarify his own thoughts about the situation. The reason why such an apparently obscure approach was adopted was because Mansell had used the methodology before, and found it led to personal enlightenment about a problem situation.

The structure of a T262 project is as follows. The methodology starts with a problem identification process during which a game called PIG (Problem Identification Game) is played (Jacques & Talbot 1975). The problem chosen is then explored using a number of methods from a Design Methods Manual (Cross & Roy 1975). There is no prescribed sequence for use of these methods, or any rigid prescription of which ones to be employed. There is, however, a loose contingency structure for use of the methods. Firstly, the design project might be either at the exploration stage, or at the stage of generation of ideas, or at the stage of selection of promising ideas. Different methods are on the whole more suitable for one stage rather than another. Secondly, the focus of design might be either a system, a product or a component. Particular methods are suitable for each of these possibilities. After having applied any particular method, guidance is available on which method to use next.

2. The Problem Identification Process

Problem Identification consists of phases of preparation, exploration, formulation and determination of project potential. In the preparation phase a problem statement is selected or prepared. The exploration phase analyses the problem situation in terms of pairs of elements from the situation that have a problematic relationship to each other. In the formulation phase an initial analysis is made of what is unsatisfactory in the problem situation, and an attempt is made to define the core of the problem and its ramifications. Finally, a potential project to solve the problem is defined. The objective existence of 'problems' is not questioned by the methodology, and the *weltanschauung* of the definer of the problem is not enquired into. Alternative points of view of the problem situation are not sought. Use of PIG however, encourages creative thought, and this itself may give rise to multiple alternative views of the problem.

3. Preparation Phase

The preparation phase consists of selection of a starting problem statement. The statement chosen represented what was problematic about the project, rather than being an expression of the problem to be solved. Selection of a problem statement expressing unease about the project itself is an indication that the problem context was coercive. The following statement by Nissen (1984), was chosen as input to PIG, with no clear idea of the likely consequences.

> *During the short history of information systems research most studies have centred around producing knowledge on which to base methods of design and implementation of such systems. The implicitly intended knowers of such knowledge were mainly persons specializing in information systems analysis and design. The values they supported in their work were predominantly those of their employers. The research methods applied were fetched from natural science and from objectively explaining*

social sciences.

This statement seemed to diagnose what was wrong with the approach being taken in the Alvey-funded research. The project was concerned with the generation of knowledge for use by professionals in the interests of their employer (the computer manufacturer), and had a scientific facade (Alvey-funded research). The academic discipline expected to make a major contribution was psychology - an 'objectively explaining' social science. From the point of view of the computer manufacturer, the function of the academics in the project team was to provide legitimacy to the knowledge being offered. The consumers of the knowledge were intended to believe that they were consuming the fruits of objective research. The academics believed that what they were doing could possibly be regarded as research if the work could be scientifically validated. The computer manufacturer did not take this validation seriously. The nature of the collaboration caused the academics to behave like employees of the computer manufacturer.

4. The Exploration Phase

The exploration phase consists of the generation of a 'key problem pair' and the subsequent generation of a network of problem pairs. A problem pair consists of two elements from the problem situation that have a problematic relationship to each other. An asterisk is used to indicate the problematic relationship. The key problem pair chosen was:

Research Methods * Systems Development Methods.

Choosing this as the key pair implied that the generation of knowledge about system development methods posed a research problem.

The generation of a network of problem pairs is the aspect of

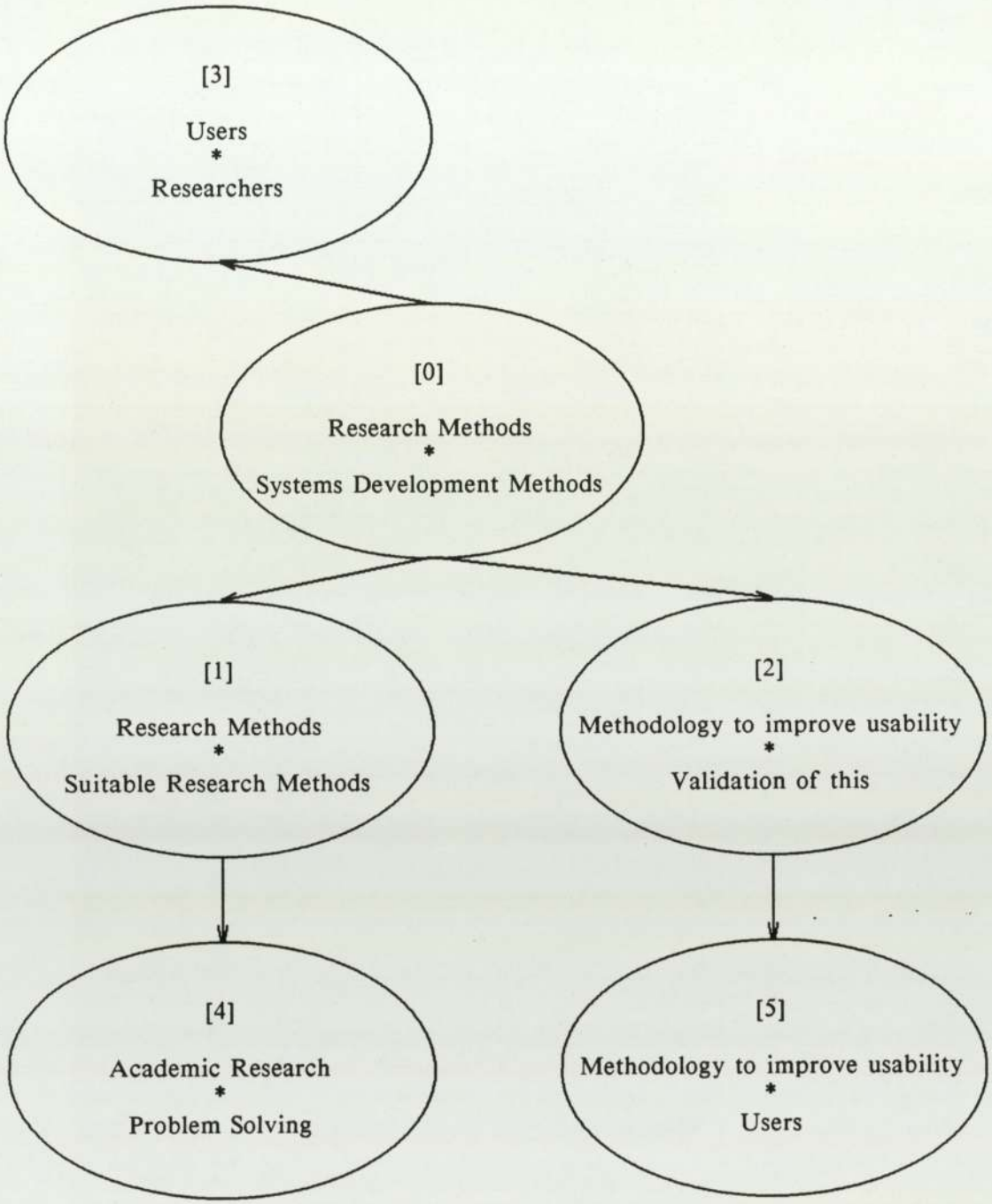

FIGURE 1 - Problem Pair Network

Problem Identification that is most like a game. It is played as a board game using throws of a die to move round the board. Each time a move is made certain operations are carried out that result in the generation of a new problem pair. Playing the game causes one's thinking about, and knowledge of, a situation to be formalized in a semantic network. The 'game' aspect encourages creative thought.

The problem pair network that resulted is shown in Figure 1. The train of thought behind this network developed as follows.

0 to 1

The problem being analyzed is that of finding suitable research methods to generate knowledge about systems development methods.

0 to 2

This general problem is exemplified in the project described in this thesis.

0 to 3

Here a revulsion against the idea of 'research' and 'researchers' occurred. Knowledge should not belong to a privileged group to be doled out as it sees fit. Users of systems should be empowered to enhance their own knowledge about how to change them. This thought was brought about by PIG requesting that a complete contradiction of the original problem statement be drafted. Here is the antithesis of Nissen's statement that was produced.

> *During the long history of information systems research most studies have centred around methods for incrementally revising such systems by the users of the systems themselves. The knowledge is not intended for technical specialists but for anyone who needs or wishes to provide information or receive it. The values informing this work have derived from hostility to capitalism and centralized bureaucracy. The research methods employed have derived from phenomenology.*

1 to 4

Methods to generate knowledge about methods should not have to count as scientific research. The model of disinterested, objective, scientific enquiry is not appropriate for problem solving in systems of human activity. The values of scientific academic research have a problematic relationship with the requirements of problem solving.

2 to 5

The computer manufacturer's view of a methodology to improve software usability does not involve users. It takes the form of workshops attended by professional designers and marketeers. The user is the person who should assess software usability, and unless the methodology incorporates user research it cannot be successfully validated.

5. The Formulation Phase

The Formulation Phase of PIG subdivides into the following Stages:

[1] Stage 1 - Review

[2] Stage 2 - First Formulation

[3] Stage 3 - Intervention Potential

5.1. Stage 1 - Review

At this stage of PIG a succint statement is made that provides the latest view of the problem. The following statement was made.

> *System development methods are suitable only for application by experts, not by users of the designed products. Academic research into systems development methods itself poses a methodological problem - is a science paradigm appropriate? Perhaps researchers should become both designers and users. Users of information technology products need to be given the means of altering/redesigning the products to meet their needs. These users may be ignorant of appropriate methods. There should be*

a 'science of method' that allows people to choose appropriate design methods. There may be a problem for vendors of information technology of judging exactly how much of the product design should be 'hard' (difficult/impossible to change), and how much 'soft' (amenable to change by the users). A design that is too hard is inflexible, and design mistakes are difficult, slow and costly to correct. A design that is too soft requires too much user effort to make it usable.

5.2. Stage 2 - First Formulation

The purpose of this stage is to formulate a systematic and explicit statement of the problem, its causes and ramifications.

The following analysis was made of what was wrong, unsatisfactory or undesirable about the situation.

Vendors waste money producing software products that are not usable, or usable with difficulty. These products are not commercial successes.

Users suffer the frustration of using badly designed and documented software. They may give up using the software or use only a fraction of its potential. The purchaser wastes money.

Software designers feel frustrated at the time they waste working on failed products.

The core of the problem is the gap between designers and software users.

The following analysis was made of the causal factors.

The educational system is elitist. Designers are well-educated experts, whereas many users will be less well-educated or lack an understanding of technology.

There is a dichotomy between technological and social imperatives. Software is a form of technology that can automate human information-handling tasks, and radically alter the way in which people must work. What is good for 'efficiency' may not be good for people.

The designer may be ignorant of the context of use of software. Designers very rarely use their own products. Designers serve the interests of their employers, not those of users.

Ramifications of the problem were specified as follows.

Scarce design talent is misused.

User frustration and fatigue could lead to health problems or

accidents. In the future a major disaster could occur as a result of poor software usability, for example in military applications, air traffic control or the nuclear industry.

5.3. Stage 3 - Intervention Potential

At this stage the possibilities for change in the present situation are identified. The analysis identifies firstly the people who would benefit from change or be penalized by it, and those who have the power to sponsor change or stifle it. Secondly, aspects of the situation that are easy to change, and those aspects that might be change-resistant, are identified.

Interested parties who might be expected to benefit from attempts to improve the situation were identified as follows.

Direct end-users of software should be able to work more effectively and make fewer mistakes.

IT vendors should have fewer failed products.

Designers might benefit in the long run as they enhance their skills and improve their job satisfaction.

Interested parties whose best interests might be served by leaving the situation as it stands were identified as follows.

Designers in the short-run lose autonomy and need retraining.

User organisations need to acquire expertise in specifying requirements and tailoring software. This may be very expensive, particularly for small organisations.

Interested parties who might have the power to initiate, sponsor or support change were identified as follows.

IT vendors.

Government.

The position of designers was ambivalent; clearly they could resist but might welcome change.

Moving on from the interested parties to the problem situation itself, it is evident that the problem situation is a socio-technical system. The

specification and creation of user-revisable software is a problem for software engineering (a technical problem). The technical problem can be solved only if the requirements can be clarified. The information technology supply and demand system is a social system. Bringing about change in this system is difficult because nobody owns the system. There is no point in solving the technical problem unless the solution relates appropriately to the surrounding social system.

6. The Project Potential Phase

The Project Potential Phase of PIG consists of the following Stages:

[1] Stage 1 - Project Matrix

[2] Stage 2 - Final Formulation

6.1. Stage 1 - Project Matrix

By this stage of PIG the player should have constructed a well-formulated problem statement consisting of a 'core', supported by some assessment of causes, effects and likely responses to intervention. In this stage three types of intervention are considered. Firstly, ACTION involves taking direct action in the real world to solve the problem. Secondly, PLAN involves the devising of a plan for action at some future date. Thirdly, RESEARCH involves an enquiry, the results of which might contribute to future action. Each type of intervention can occur at one of two points of intervention. Firstly, CORE involves changing some aspect of the problem core. Secondly, PERIPHERY involves changing some aspect of the environment of the problem situation. The following analysis was made at this stage.

ACTION ON THE CORE

Design, implement, use and evaluate a piece of software that is intended for easy revision by users.

ACTION ON THE PERIPHERY

Give training courses to designers that encourage them to think about human factors.

PLAN FOR INTERVENTION ON THE CORE

Specify an enhanced methodology for software design that is intended for both designers and users.

RESEARCH ON THE CORE

Study software in use and analyse its usability.

RESEARCH ON THE PERIPHERY

Get more information on designers and users (educational background, organisational role, nature of tasks performed etc.)

6.2. Stage 2 - Final Formulation

In this stage one of the suggestions for intervention is chosen and a design project specified in the following form:

[1] A short summary of the problem core.

[2] Proposed action.

[3] Expected outcome.

[4] Doubts about successful completion.

The following formulation was made.

CHOICE OF ACTION

Specify an enhanced methodology for software design that is intended for both designers and users.

PROBLEM CORE

Usability problems in professionally designed software - gap between designers and users.

ACTION

Concentrate on the deferring of design decisions - which decisions to defer, and how to give the user the power to take them.

OUTCOME

Uncertain. Some feel for the feasibility of the idea of deferring design decisions.

DOUBTS

Are there any generalized principles or would each application area yield unique criteria? It may be difficult to generate generalizable knowledge.

7. Design Briefing

The second major component of the OU design methodology consists of the application of a number of design methods. The objective of this work is the generation of possible solutions to the problem identified earlier. The outcome of this work corresponds to a design brief - a preliminary statement of what needs to be designed. Much more detailed work would be necessary to implement feasible designs.

Use of the Design Methods Manual necessitates the formulation of a strategy. This is because it consists of a collection of fifteen design methods from which appropriate methods must be selected and combined in an overall design process. Thus the designer is responsible for providing the structure for use of the methods, taking account of contingent factors. One contingent circumstance is the action that the designer is seeking to take at a particular time, either to explore a problem area, generate solutions or select appropriate solutions. The other contingency is concerned with the nature of the thing to be designed, whether it is a system, a product or a component of a product.

At the outset it seemed that most emphasis should be put on exploring the problem situation, since inadequacies had been perceived in this area. Moreover the focus of design was clearly a system, albeit an abstract system. So the strategy was adopted of following these methods in this sequence:

[1] Objectives Tree

[2] Counterplanning

[3] User Trip

Each of these methods was recommended for systems design at the

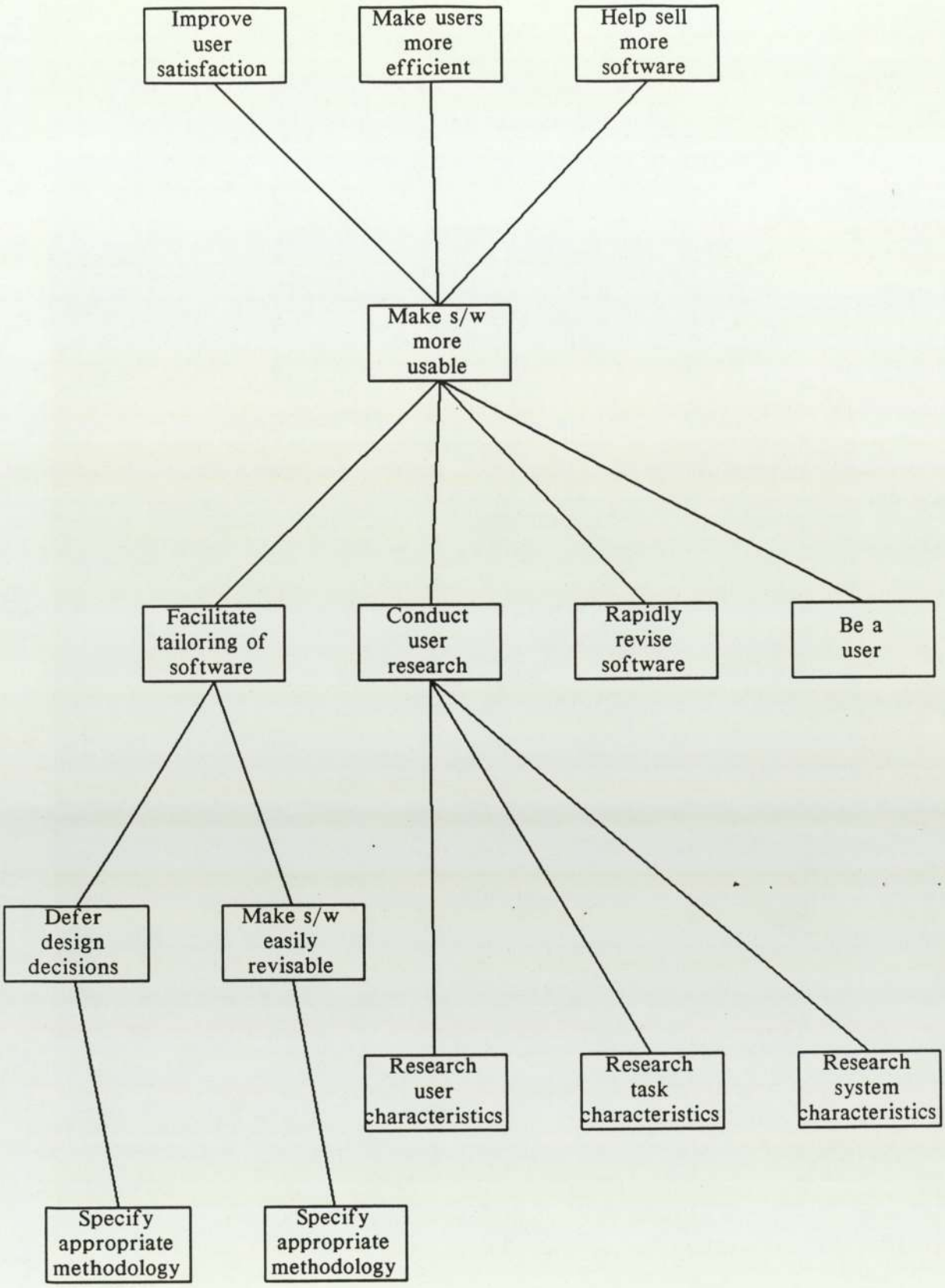

FIGURE 2 - Objectives Tree

problem exploration phase. An Objectives Tree is a method of defining the design objectives and sub-objectives in a project. Counterplanning is a method of examining the assumptions underpinning a design proposal, and by considering conflicting assumptions coming to a changed perception of the problem. A User Trip is a method for finding problems, insights and ideas, based on the careful and deliberate use of an existing product or system. The User Trip was conducted while working in a user department of the computer manufacturer. The OU design methods were supplemented in this project by SSM-style root definitions and conceptual models. For example, as a result of difficulties encountered in constructing an objectives tree, root definitions and conceptual models were also used. They also proved useful during Counterplanning.

8. Objectives Tree

The procedure to be followed when using this method is:

[1] List the known objectives for the project.

[2] Expand this list of objectives into sets of both higher-level and lower-level objectives.

[3] Represent the hierarchy of objectives diagrammatically.

The objectives hierarchy produced is shown in Figure 2. The project is concerned with making software more usable. Higher level objectives to which this contributes are:

[1] Improve user satisfaction.

[2] Make users more efficient.

[3] Help computer manufacturer sell more software.

Software is made more usable by achieving the following objectives:

[1] Facilitate the tailoring of software by users.

[2] Conduct user research.

[3] Rapidly revise software when requested by users.

[4] Send designers on user trips.

Facilitating the tailoring of software by users requires the following objectives to be achieved:

[1] Defer appropriate design decisions.

[2] Make software easily revisable.

User research takes the form of:

[1] Research user characteristics.

[2] Research task characteristics.

[3] Research system characteristics.

The 'enhanced methodology for software design' referred to in the Final Formulation Stage of PIG would contribute to the objectives of deferring appropriate design decisions, and making software easily revisable.

After constructing the objectives tree, the following conclusions were reached. The proposed action for solving the identified problem was only one possibility among many. There was considerable uncertainty that any particular action would produce the right effect. In short, a loss of confidence in the proposed solution was experienced. In addition an objectives tree was felt to be too constraining a model - a network rather than a hierarchy was felt to be more suitable. The idea of using an SSM conceptual model presented itself so the design strategy was modified to try this out.

9. Conceptual Model

At this point an attempt was made to construct a conceptual model of a human activity system relevant to the problem being worked on. The attempt was not successful but is reported here to show the learning that took place. Confusion was experienced during the exercise as to whether a model was being produced of what exists in the world or of some ideal-type system. Another attempt was made later in the project. First of all a number of alternative root definitions were produced.

> *A system that ascertains user requirements for software and produces such software subject to the constraint that any particular product must have a viable number of customers.*
>
> *A system that determines what software is necessary for a given range of computers, and persuades, educates or trains customers to use it.*
>
> *A system that determines what software is suitable for meeting organisational needs and acquires and installs it.*
>
> *A system that monitors use of third party software, reports errors and receives corrections and makes suggestions for improvement.*

The first two of these (embryonic) definitions are from the point of view of the software vendor, and the second two from the point of view of the user. The first definition is of a requirements-driven system that seeks the greatest happiness of the greatest number. In real-world terms the first definition has a certain ring of truth and would apply to certain products in certain markets. A computer manufacturer, however, does not simply respond passively to requirements, but creates requirements by a marketing and training effort. This aspect is captured by the second definition that places the onus of requirements generation with the vendor. Historically the bulk of computer software has been produced in this way, because of user ignorance of new technology. Neither definition captures the full richness of the software business.

The third and fourth definitions describe systems that might, or

arguably ought to exist in user organisations. The software market is competitive and users can often choose between competing products providing similar features. User organisations may also have a choice between developing applications in-house and using standard software packages. Software users have an on-going relationship with software vendors. Errors are reported and eventually corrected, and new releases of software are made containing enhancements. Users have some opportunity to influence future product development by making comments and criticisms.

A fifth root definition was produced.

A system of software producers and software users, linked by flows of software, information about software, error reports and information about requirements.

This root definition is unacceptable in SSM terms because it refers to activities that take place in the real-world, rather than defining an ideal-type human activity system. It is a market-place that has no owner and is not under central control. Software vendors cannot control the organisational activities of their customers, so they cannot ensure use of their products. Users cannot require vendors to produce exactly what they want.

If the system is compared with Checkland's 'formal systems model' then the following observations can be made:

[1] The system has no ongoing purpose or mission. It exists as a self-maintaining causal network.

[2] No measures of performance are meaningful since no objectives are being pursued.

[3] The system contains no decision-taking process. The system components have local decision-taking processes but there is no overall

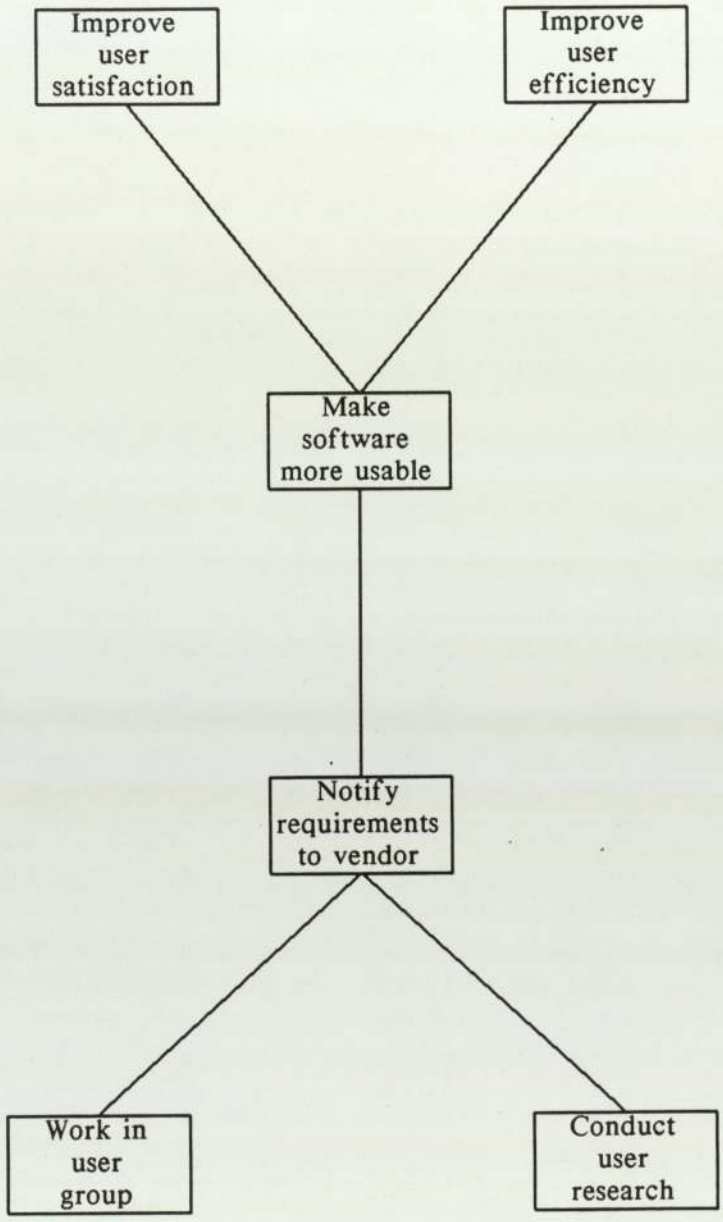

FIGURE 3 - Objectives Tree
(from user point of view)

co-ordination.

This state of affairs is typical of economic activity in a market-place with no planning and control at a level above organisational activity. Competition between computer manufacturers and software vendors may give rise to inefficiency, for example too many different incompatible models of computer. Scarce software design skills may be wasted replicating software for incompatible machines. Competing hardware and software vendors may not be able to afford the research necessary to thoroughly establish user requirements. User requirements may vary arbitrarily because of lack of planning in the user industries.

The conceptual modelling exercise was discontinued at this point because of the intractable nature of the system. One lesson that was learned, however, was that the objectives tree (Figure 2) produced earlier was biased towards the point of view of a software vendor. A similar tree was constructed for a user organisation, based around the objective of making software more usable (see Figure 3). A user organisation concerned with this objective would conduct user research and work in a user group, with the aim of communicating requirements to the vendor. Given the current state of the art, user organisations would not normally set about revising third party software themselves. If they were profoundly dissatisfied with commercially available general-purpose software they might commission special-purpose software. However, it seemed very unlikely that user organisations would be concerned with usability as distinct from functionality. Any system set up to liaise with vendors would deal equally with either aspect of software.

It had become clear during these exercises that an objectives tree is a special case of an SSM conceptual model. Each activity on a conceptual

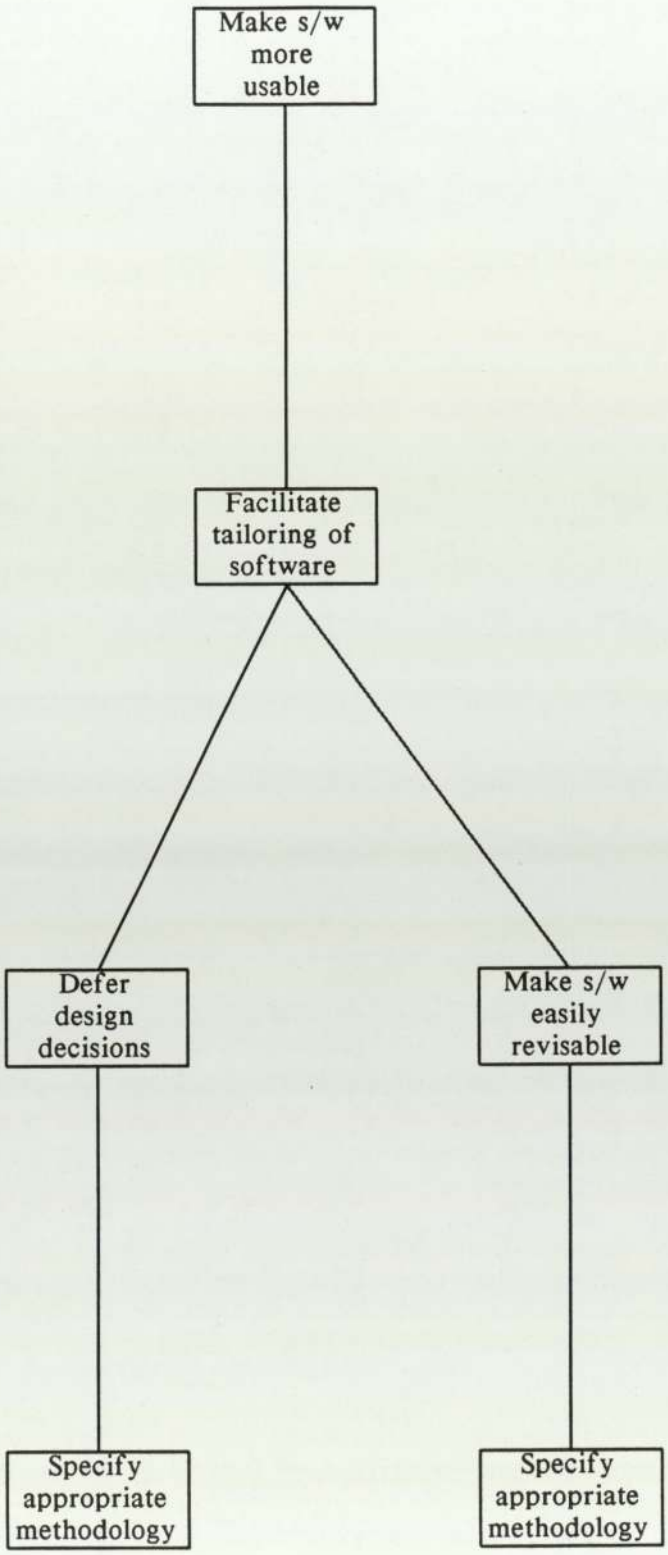

FIGURE 4 - Plan 1

model seeks to achieve an objective. Each activity can itself be modelled as an activity system, so the concept of hierarchy is incorporated. An objectives tree is an insufficiently systemic model, however, since it is constrained by its hierarchic structure. Moreover, the SSM root definition and conceptual model explicitly evinces a point of view. At this point in the application of T262 Mansell felt he was beginning to understand the difference between a design paradigm and the interpretive approach of SSM.

10. Counterplanning

The Counterplanning procedure starts with an existing plan, proposal, design or decision. The basic assumptions underlying the plan or proposal are identified. An alternative set of counter assumptions is generated. Using these counter assumptions a counter plan is generated that is deliberately the 'deadly enemy' of the original plan. By considering the plan and counter plan together a new plan is synthesized.

The existing plan (Plan 1) is shown in Figure 4. The focus of the proposed action or solution is the specification of design methods that make it possible for designers to defer certain critical decisions about their product. Users will then be able to take these decisions themselves, but to do so will require the software product to be easily revisable. The feature of easy revision requires the application of appropriate design methods. The assumptions underlying this plan were specified as follows:

[1] User tailoring of software will improve usability.

[2] User tailoring of software is organisationally feasible.

[3] Contexts of use for software products vary substantially (i.e. there is a need for user tailoring in specific task or organisational contexts.)

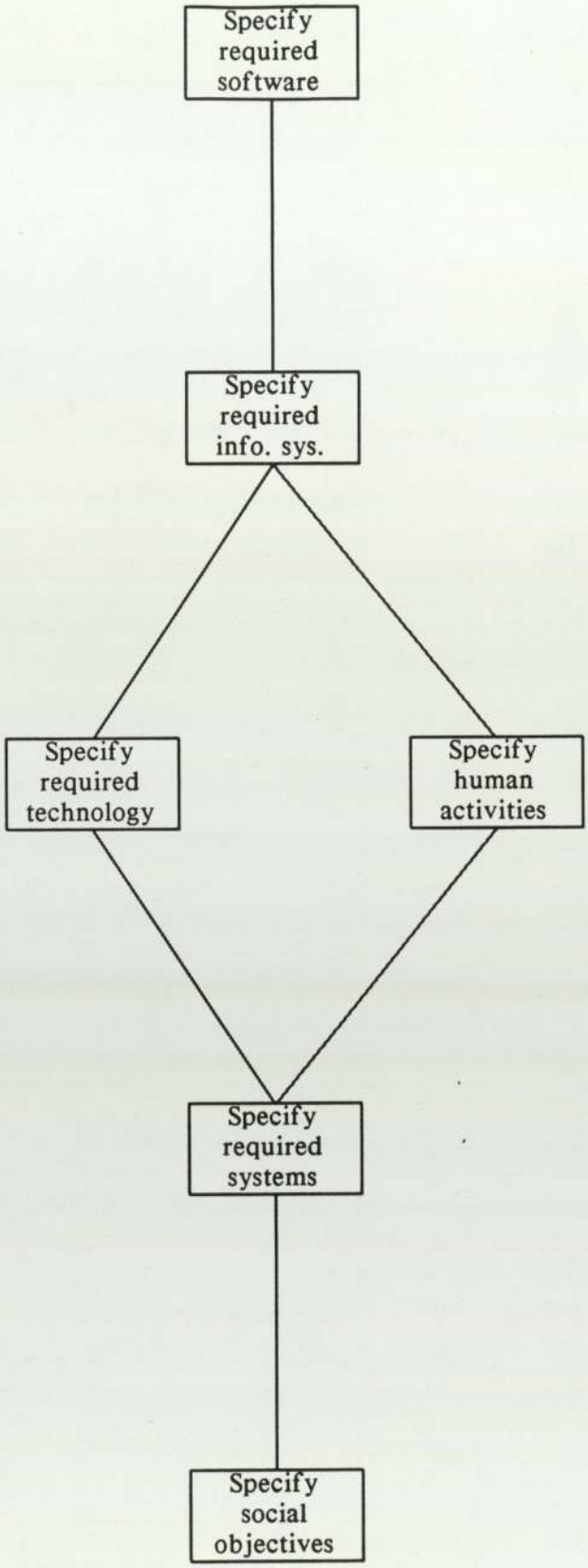

FIGURE 5 - Plan 2

[4] It is more desirable to facilitate user tailoring of software than all the other things that could be done to improve usability.

[5] The current structure of the software business cannot be changed.

[6] Usability is a clear, sharply focused concept.

[7] Usability problems are worth solving.

The following counter assumptions were made:

[1] Variations in contexts of use for software products are arbitrary and undesirable.

[2] The software business is badly organized and can be changed.

[3] Usability is not a clear, sharply focused concept.

[4] Usability problems with software are relatively trivial.

The essence of the counter assumptions is that the usability problem points to a more substantial structural problem in the software business.

The plan (Plan 2) based on these counter assumptions is shown in Figure 5. The orientation of this plan is towards the specification of required software at a national level. Some state planning agency would specify social and economic objectives to be pursued. Systems required to pursue these objectives would be specified. The implementation of these systems would require appropriate technology, and the specification of appropriate patterns of human activity. To perform the specified task would necessitate the construction of information systems, and these would require the specification of software systems. This utopian (or in the wrong hands totalitarian) plan presumes the feasibility of planning economic and social activity at a level above organisations. Organisations are brought into existence to implement rational plans. Use of technology and specification of human activity is similarly derived from rational

planning.

The relationship of this plan to what was previously conceived of as a software usability problem is as follows. In society at present there is no effective planning of economic and social activity. Organisations have considerable autonomy, and human activity and information systems evince arbitrary and irrational variability. Competing software vendors have to sell products in a market with this characteristic of irrational variability. The attempts of the software vendor to divine the characteristics of the market-place, and compromise between conflicting arbitrary requirements, produces products with what are perceived as usability problems. If, for example, there was specified one standard national payroll package, then its performance could be optimized on all relevant dimensions. The current situation permits the existence of many alternative payroll packages for many alternative types of computer. The probability of any particular package being optimal is much lower in the existing situation.

The Counterplanning exercise terminates with a plan that is a synthesis resulting from dialectical tension between the original plan and the counter plan. It had become apparent that counterplanning requires a shift in world-view. So any new plan would have to be made from a particular point of view. Reluctantly the decision was taken to adopt the point of view of the computer manufacturer. Plan 1 was feasible but the likelihood of improving the situation was uncertain. Plan 2 was utopian and infeasible. Plan 1 involved the design of methods to achieve a particular purpose. Plan 2 involved the re-design of most of the economy. The synthesis of the previous two plans, Plan 3, was an attempt to design a system that would deliver required software. Plan 3 was more ambitious than

Plan 1 but far less so than Plan 2. SSM conceptual modelling was used to sketch out a design. The first attempt at a root definition was as follows.

RD 1

A system owned by a computer manufacturer that establishes requirements for, and designs, implements and maintains general purpose software systems intended for sale to the manufacturer's customers.

One problem with this definition is that organisations do not have requirements for general purpose software systems. They have requirements for information systems and wish these to be met in the most economic way. This may involve the use of general purpose software. The other problem is that a computer manufacturer does not simply establish requirements and then satisfy them. It will only do so if this can be done profitably, and this implies the existence of a large enough market.

An attempt to meet these objections gave rise to a second attempt at a root definition.

RD 2

A system owned by a computer manufacturer that establishes requirements for information processing in organisations, and conceives of, designs, implements and maintains general purpose software systems that are intended to meet those requirements in a sufficiently large market to warrant investment in the software.

This root definition posed problems over conceptualizing the system as a transformation. In RD 1 the transformation was thought of as *Requirements* into *Satisfied Requirements*. This could not be true of RD 2 because not all requirements were to be satisfied by this system. Requirements that could be serviced profitably would be met by selling a

product. The system in question, however, was not concerned with selling, simply with making products available for sale. A transformation of requirements (which are abstract) into products (which are concrete) was also unsatisfactory.

A third and final root definition was drafted as follows.

RD 3

A system owned by a computer manufacturer that determines markets for general purpose software systems, conceives of possible software products to sell profitably in these markets and designs these products.

The **CUSTOMERS** of this system would be the implementors of the designs (i.e computer programmers and technical authors). They would be responsible for converting abstract specifications into working systems. The **ACTORS** in the system would be marketeers and systems designers. The **TRANSFORMATION** performed by the system would be of a Product Opportunity into a Design for a Product. The existence of a product opportunity can only be guessed at, until a profitable sale of a product is made. When such an event occurs it proves the prior existence of a product opportunity. A product opportunity partly derives from a prospective purchaser's objective requirement for some service. Opportunities can be to some extent created, however, by skillful marketing. The transformation performed by the system is problematic, therefore, in that there cannot be total certainty that is has taken place. The **WELTANSCHAUUNG** of the root definition is that software must be sold in a market-place and its production must be profitable; software production is liable to be profitable, that is, markets exist. The **OWNERS** of the system are senior managers in the computer manufacturer. The **ENVIRONMENT** of the system includes the company sales-force, prospective customers,

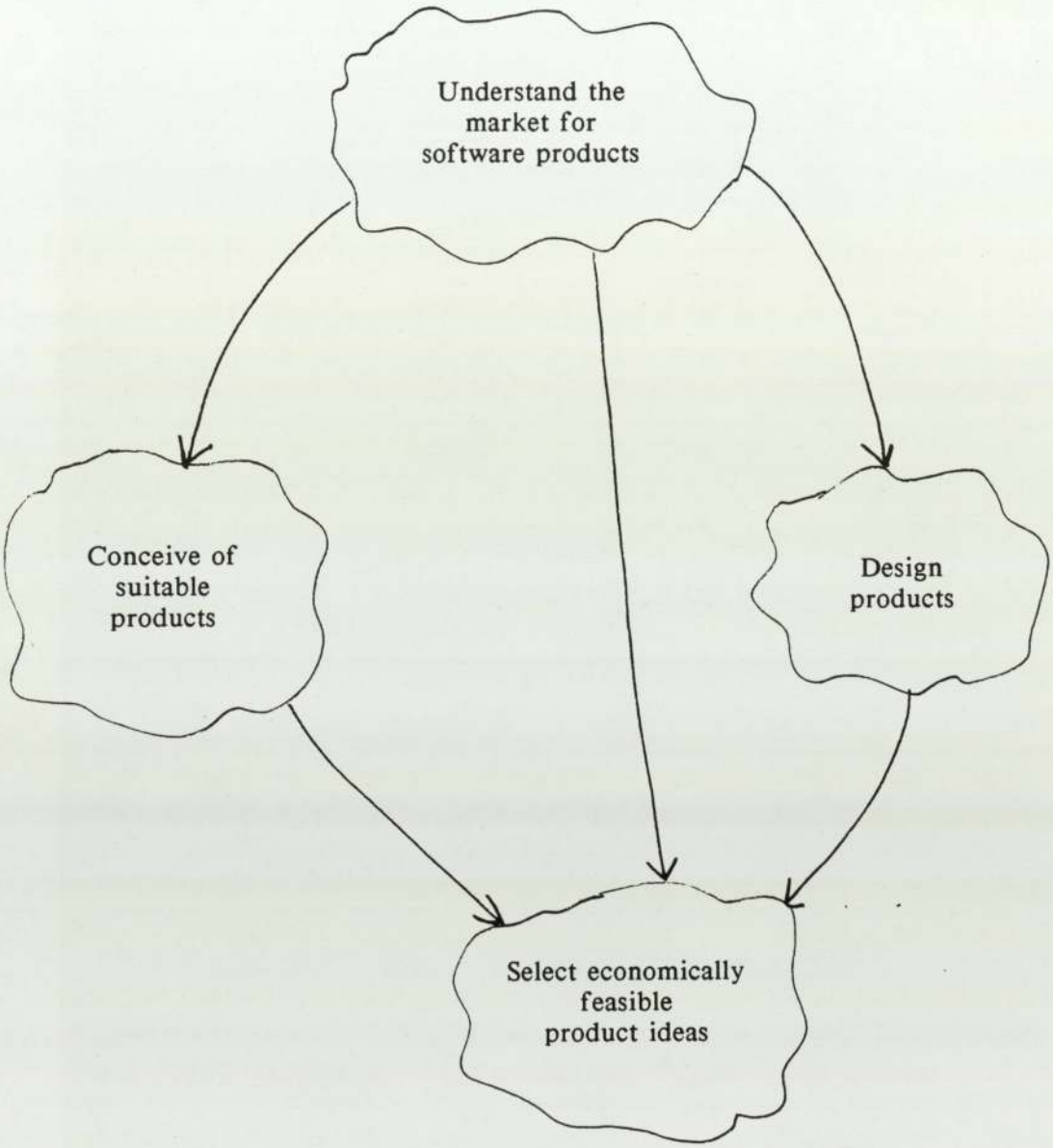

FIGURE 6 - Conceptual Model of RD3

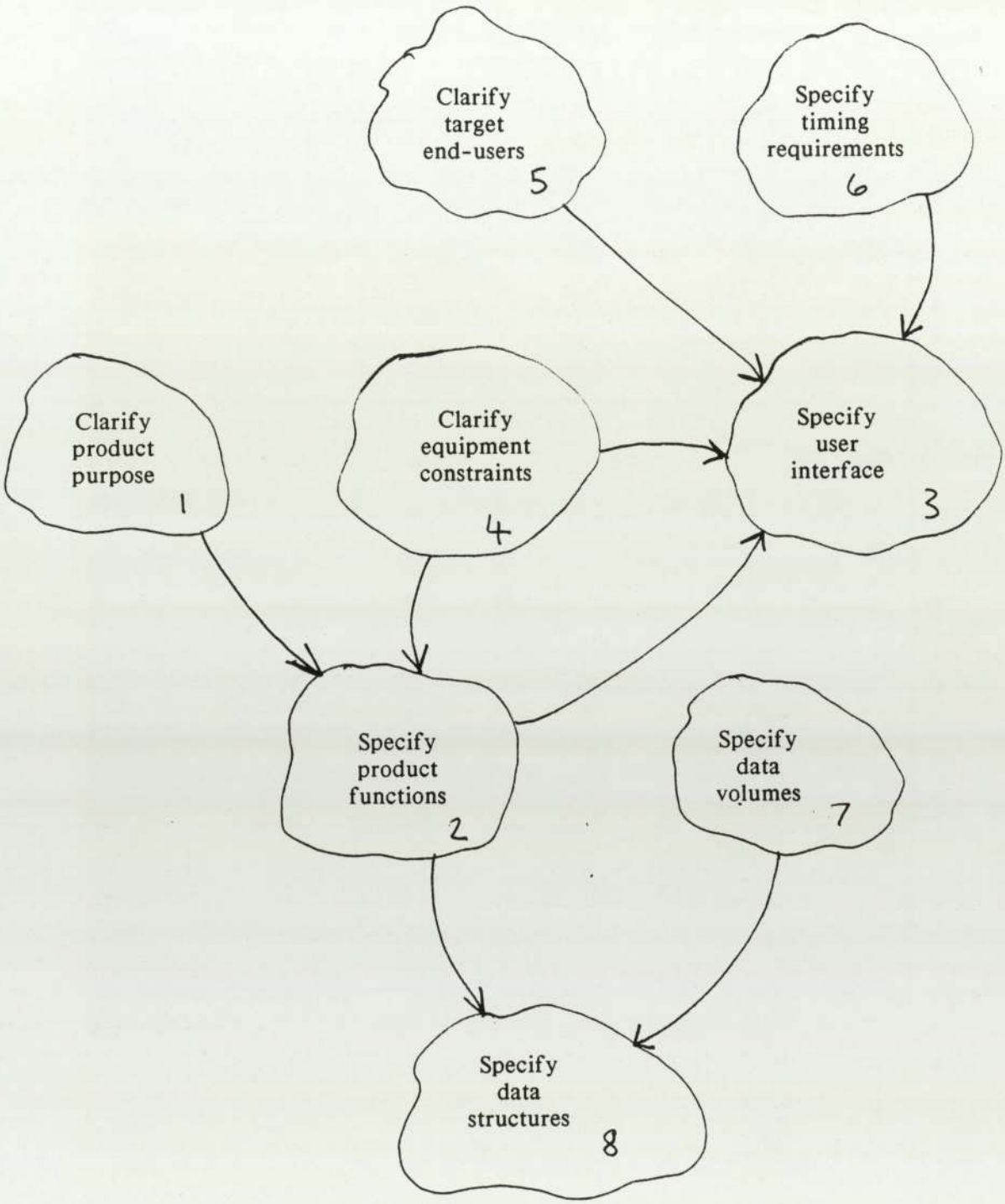

FIGURE 7 - Conceptual Model of "Design Products"

prospective end-users of products, and systems development staff in the customer organisations.

The conceptual model of the system is shown in Figure 6. The Design Products activity was selected for more detailed analysis. It was taken to be 'A system that converts ideas for software products into detailed designs suitable for implementation'. **CUSTOMERS** of the system are computer programmers who would have to code from the design specification and technical authors who produce documentation for the product in parallel with the programming activity. **ACTORS** of the system were systems designers. The **TRANSFORMATION** performed by the system was of an outline specification of a product into a detailed design specification. The **WELTANSCHAUUNG** of the definition of the system is that it is possible and desirable to produce an abstract representation of a software product prior to its implementation. The **OWNERS** of the system were project managers and quality control staff concerned with standardizing design methods and specifying documentation standards. The **ENVIRONMENT** of the system included the marketeers who generated the ideas for products and would be concerned about their realization. The conceptual model of the system is shown in Figure 7.

The rationale for this model with respect to the root definition is that an 'idea for a software product' is taken to be a specification of the purpose of the product and its target end-users, together with a specification of the equipment on which the product is to run. These specifications may be vague, (because an idea for a product is not a design) and so the Design activity must clarify them (Activities 1, 4 and 5). A 'detailed design' of a software product is assumed to consist of a specification of product functions, user interface, timing requirements, data volumes and

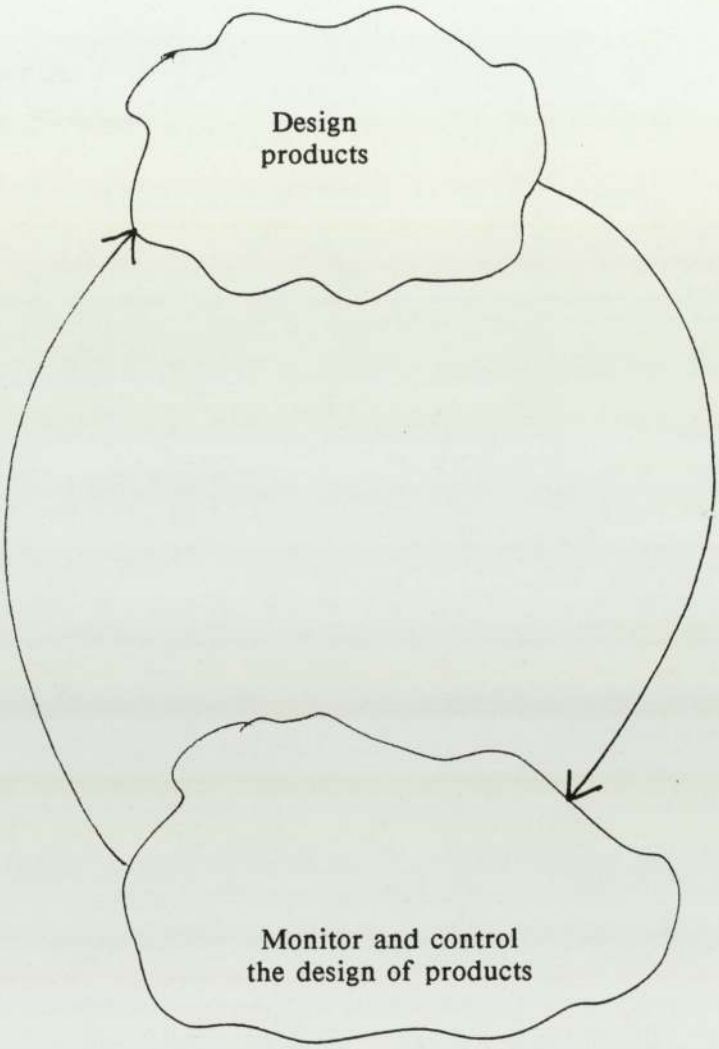

FIGURE 8 - Monitoring & Control of "Design Products"

data structures (Activities 2, 3, 6, 7 and 8).

Attention was now directed to the monitoring and control of the Design Products sub-system, using the model shown in Figure 8. The *effectiveness, efficacy* and *efficiency* of a Design Products system was reviewed. Effectiveness of the activity was apparently determined by the market research and product conception activities in the surrounding system. Marketing failure could give rise to well-designed products that were commercial failures. Efficacy of the activity was problematic. Since ideas for products were necessarily vague there was a problem in deciding whether a design did or did not meet requirements, and if so how well. Efficiency was not thought to be a problem - this was a question of motivating and managing talented professional staff. To ensure efficacy in design the monitoring and control activity had to provide a means of validating designs that checked that suitable means were being pursued for known ends. The ends to be pursued, however, were necessarily imprecisely expressed. To some extent desirable ends became clear only after suitable means had been provided. This is the central paradox embodied in the activity of design that has given rise to the concept of solution focusing. The paradox was thought to be resolvable by the inclusion of prototyping in the monitoring and control sub-system. Designs would be realized in a working prototype, exposed to live use to clarify requirements and test solutions, and the results of the exercise used to revise the designs. The relevance of prototyping was explored in the next phase of the project.

11. User Trip

A User Trip is a method for finding problems, insights and ideas based on the use of an existing product or system. The procedure followed is to decide which user's point of view is to be adopted, decide the limits and variations to the user trip and carry out the trip, recording actions, impressions, thoughts and ideas. The implicit assumption in the method is that the product or system to be used is the one with which problems are associated. Since the problem to be solved in this project was concerned with the usability of software products, the decision was taken to use a particular product in its real operational setting.

11.1. The Product

The product in question is known as Quickbuild. The product consists of:

[1] A methodology for information systems analysis and design.

[2] A suite of software components.

[3] Documentation in the form of a User Guide and on-screen Help.

[4] Training.

The software components consist of:

[1] Quickbuild Pathway (a series of menus that guide the user through the analysis and design stages).

[2] A Data Dictionary that is used to store documentation associated with analysis and design.

[3] A Database Management System (IDMS).

[4] A Transaction Processing System to permit multiple interactive access to stored operational data.

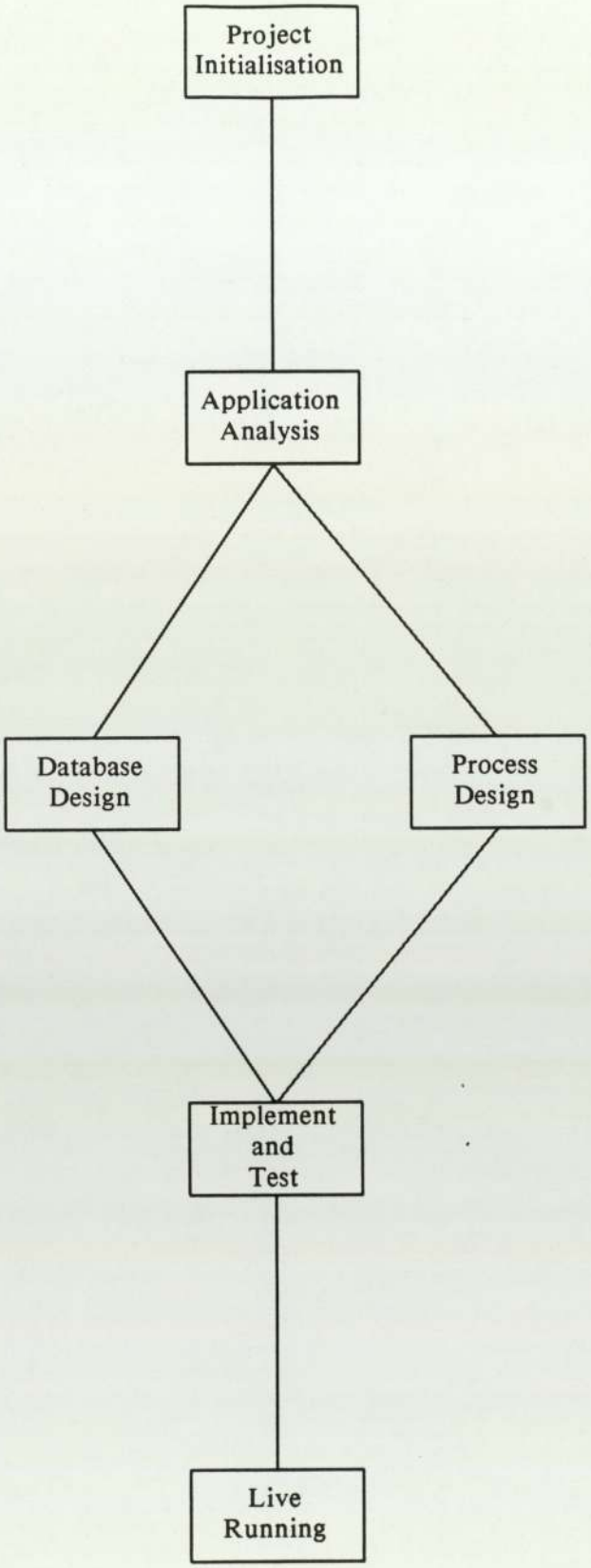

FIGURE 9 - Quickbuild Methodology

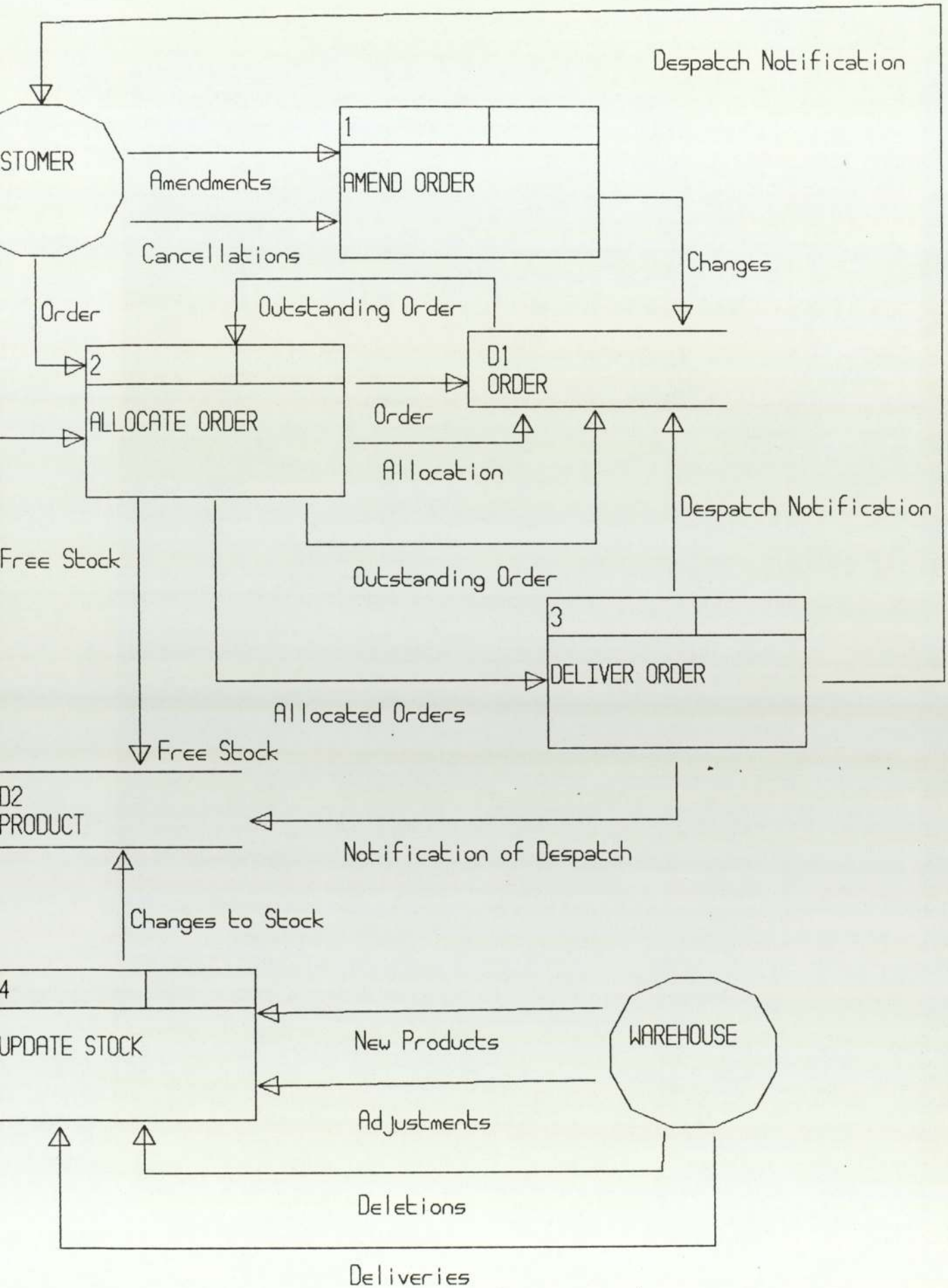

FIGURE 10 - Example Data-flow Diagram

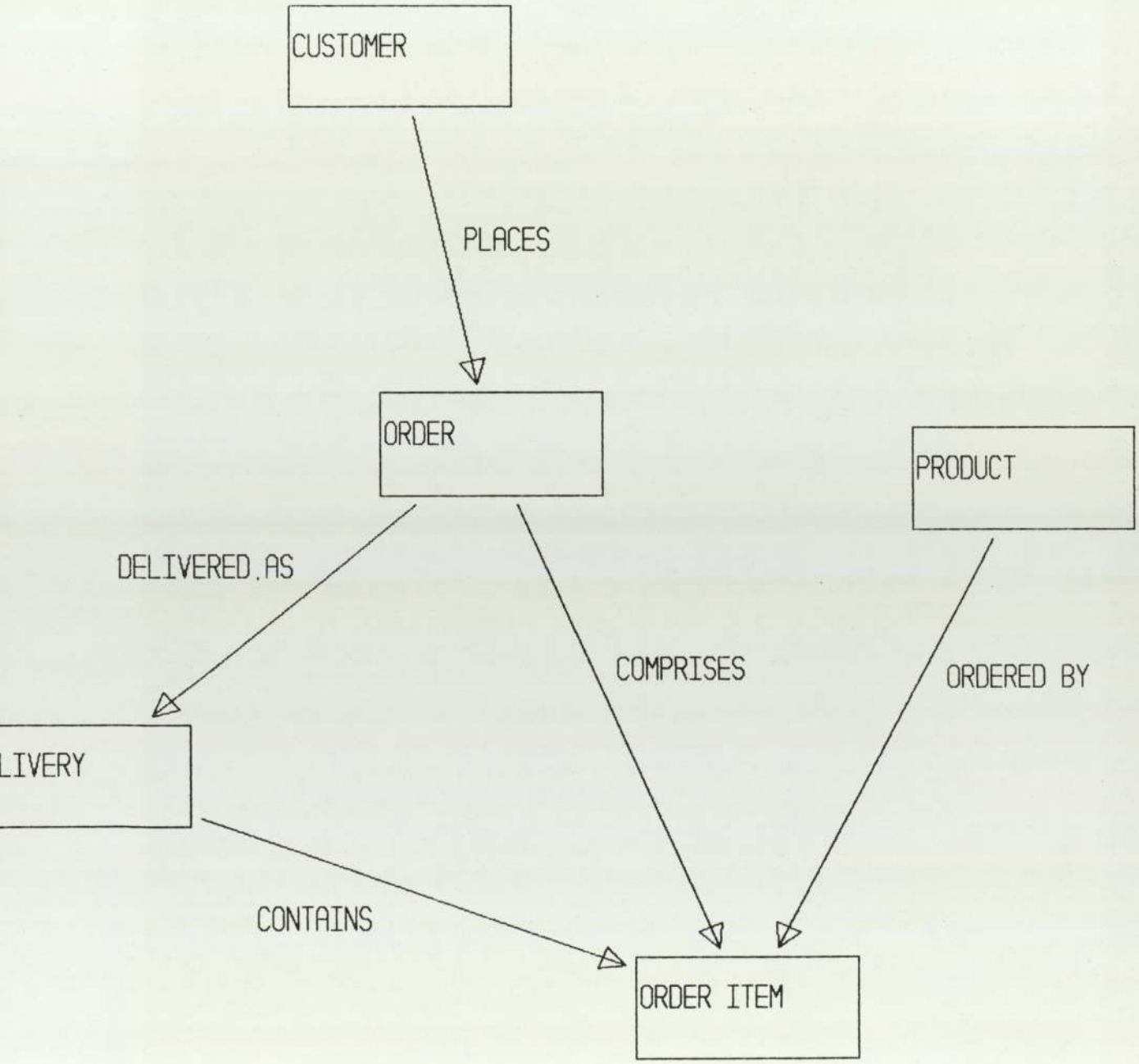

FIGURE 11 - Example Entity Model

[5] A fourth generation language product set that permits applications to be implemented that update, report on or query stored data.

The product is intended to be used by relative newcomers to data processing, for example non-technical business users.

11.2. The Test System

The Quickbuild methodology is shown diagrammatically in Figure 9. The Application Analysis phase creates a data-flow diagram of a required information system and an entity/relationship data model. In Database Design the entity/relationship model is converted into an IDMS schema design. In Process Design the required application is specified using a fourth generation language set. The user interface is also designed in this phase. Implementation consists of generating an operational database and compiling application programs.

For purposes of a User Trip, the Application Analysis and Database Design phases of the methodology were applied to a simple system. Quickbuild Pathway was used to enter the documentation of the system into the Data Dictionary. The data-flow diagram and the entity model of the example system are shown in Figure 10 and Figure 11. The logic of the system is as follows.

Allocate Order Function

Customers place orders for products and these orders are recorded. The quantity ordered by the customer is compared with the quantity available, and if sufficient is available then the order is allocated. Allocation causes the free stock level to be adjusted and a message to be sent to the Deliver Order function. Outstanding orders are regularly checked to see if they can be satisfied.

Deliver Order Function

On notification of order allocation the required goods are selected and despatched. The order is deleted from the Order Data Store and the quantity despatched is subtracted from the quantity in stock recorded in the Product Data Store.

Amend Order Function

Outstanding orders may be amended by the customer. They may be cancelled completely, or the quantity ordered may be adjusted. In either case the Order Data Store is updated.

Update Stock Function

New products are from time to time introduced to the product range and existing products withdrawn. Deliveries of products are made to replenish existing stocks. The Products Data Store is updated to reflect these changes.

11.3. Evaluation of User Trip

The information gained during the Trip was structured according to Eason's Causal Framework of Usability (Eason 1984). This framework assumes that the user reaction to a system is determined by *system functions* (task match, ease of use and ease of learning), *task characteristics* (frequency and openness) and *user characteristics* (knowledge, discretion and motivation).

System Functions - Task Match

If a system offers a good task match then it supports the user in the execution of a task in a beneficial way. The following problems were encountered during the user trip.

[1] Ideally it should have been possible to draw data-flow diagrams and entity models directly on the screen, rather than convert the diagrams to form-filling responses as required by Quickbuild Pathway. The form-filling, however was an improvement over the command-driven interface to the dictionary that preceded Quickbuild Pathway.

[2] A very serious problem was encountered over the representation of data-flow diagrams. They were supposedly part of the Quickbuild methodology and yet Quickbuild Pathway did not allow for the specification of data-flows and data-stores - just operations and events. The reason for this deficiency appeared to be that Quickbuild Pathway was compatible with an obsolete dictionary architecture i.e a version of the dictionary that did not support data-flow diagrams. The feature was only marginally usable.

[3] There was no way at all of specifying which functions on the data-flow diagram were to be automated and which were not - an activity known as 'setting the automation boundary' in structured systems analysis.

System Functions - Ease of Use

The ease of use dimension is a reflection of the effort that the user has to make to use the system efficiently. The following problems were encountered.

[1] Assuming that use of the product was split over several sessions,there was no convenient way of recording how far had been reached in any session. A 'Produce a report telling me everything I have done so far' function was required. The only way to document the work of each session seemed to be to interrogate the data

dictionary directly, using dictionary command language. Use of this language required expertise beyond the level of the assumed user.

[2] There seemed no easy way of correcting errors that had been made. Attempts to repeat steps that had gone wrong generated further error messages, because the software did not allow duplicate entries to be made in the dictionary. The software, however, would report an error condition only after making an erroneous entry in the dictionary. Errors had to be corrected using dictionary command language.

[3] The Database Design task had undesirable consequences. The product had a feature of automatic generation of an IDMS schema (records, sets and data-items) from the previously specified entity model. When this feature was initiated the software picked up every single entity, attribute and relationship in an entire shared data dictionary, and generated an enormous spurious database specification. This damage took a considerable amount of time to repair using data dictionary control language. The authors of the documentation for the product seemed to have assumed that the user would have a data dictionary for his sole use - no mention was made of the possibility of shared use, which is the commercial norm.

System Functions - Ease of Learning

Quickbuild Pathway was well provided with help facilities and learning proceeded at a reasonable pace. However it was thought that the user would have very considerable problems if he had not already learned the techniques of data-flow diagramming and entity modelling, and did not have some grasp of physical systems design. In other words the product did not teach the methodology that had to be used. The methodology was described in the supporting manual but needed prior knowledge to

understand it. The associated training course also assumed knowledge of the methodology.

User Characteristics - Knowledge

To use the Quickbuild product effectively the user would have to know how to use the techniques of data-flow diagramming and entity modelling. Knowledge of the underlying dictionary architecture and command language also proved essential. If any other than a naive translation of entity model to IDMS schema was required, then the user would need knowledge of IDMS. Although use of the 4GL product set was not included in the trip, some limited use of the dialogue generator, Application Master, was made. This required knowledge of IDMS database navigation, and interactive screen design. Overall a reasonable degree of technical competence was required to use Quickbuild. This degree of competence was more than the target users could be expected to possess.

User Characteristics - Discretion

This variable refers to the power of the user to choose not to use the product if he finds it unacceptable. Quickbuild Pathway is clearly a discretionary item that the user could ignore if he chose to use dictionary command language instead. No particular systems analysis and design methodology can be imposed on a computer manufacturer's customers, so the user could choose not to apply the techniques of data-flow diagramming and entity modelling. Even if these techniques were used the results need not be entered in the dictionary. Use of the dictionary itself is mandatory if IDMS and the 4GL product set is to be used. Customer organisations would have some policy on which software products were to be used, but none of the Quickbuild products were absolutely essential to building information systems. Conventional files were an alternative to IDMS.

COBOL was an alternative to the 4GL product set. Typical mainframe customers used IDMS and the dictionary, and made some use of the 4GL's whilst retaining COBOL. They would use Quickbuild Pathway if there were cogent reasons for doing so, but its use was likely to be discretionary.

User Characteristics - Motivation

A poorly motivated direct end-user may use what discretionary power he possesses to avoid using the product. Motivation may be affected by many other factors than those directly related to the product. Nevertheless some information technology products take on an aura of mystery and enchantment, that captivates and motivates their users. Examples of such products are the Apple Macintosh and the UNIX operating system. A certain amount of glamour is associated with the Quickbuild 4GL's, and a menu-driven front-end to the dictionary has novelty value at least. Nevertheless a subjective judgement says that Quickbuild lacks that special factor that motivates users in its own right.

Task Characteristics - Frequency

It is thought that a frequently performed task supported by software requires a different style of interface from a task that is infrequently performed. In the former case a command-driven interface is often preferred, giving maximum power to what is likely to be a skilled user. In the latter case a menu-driven form-filling interface with rich help facilities is thought to be required. Quickbuild Pathway falls into the latter category and its purpose is clearly to enable the infrequent, less skilled user to build information systems in the same way, and using the same tools, that a skilled, professional systems designer does. End users of information could possibly build their own systems using Quickbuild,

eliminating the need for professional systems designers and computer programmers. As currently constituted the product seems to fail in achieving this purpose.

Task Characteristics - Openness

Openness refers to the task which the user is performing. The more open the task the more the user has flexibility in deciding precisely how to perform it, the next action to take, and the information required to perform it. Quickbuild is attempting to support the user task of information systems analysis and design. Quickbuild assumes a closed task in that it imposes a fixed sequence of development phases on the user, assumes fixed outputs for each phase that must be entered in the dictionary, and specifies the analysis and design techniques to be used. It is arguable that the task of information systems development should be treated as more open than this. Development sequence should not be rigid, but should encourage creative thinking and prototyping approaches. Specific techniques should not be mandatory, but a wide range of optional techniques supported. On the other hand the relatively naive user would perhaps welcome process structure, gradually requiring more flexibility as skill and competence grow.

11.4. Operational Context of User Trip

The User Trip was conducted in a realistic operational context. This was the Group Information Systems Division of the computer manufacturer. GIS had the responsibility of developing internal company information systems. It did not produce software for sale to customers. It had the role of a customer, therefore, with respect to the software producing division of the company. The major respect in which GIS was unlike a

customer was that it did not, and presumably would not have been able to, use competitor's products. The way in which it used software products, however, was the same as the company's commercial customers.

GIS had many years experience of developing information systems using company software. It used VME mainframes, IDMS, DDS, and COBOL, and some use was being made of Quickbuild 4GL's, principally Application Master. The Quickbuild Pathway product was being prototyped in GIS prior to general release. The User Trip described in this thesis formed part of this prototyping process.

It was evident that GIS was not an ideal prototyping environment for the product. The GIS staff were proficient dictionary users at command language level. Quickbuild Pathway was a distraction to them. This accords with the principle that skilled users prefer command languages and are irritated by menu systems, whereas naive users need the structure provided by menus and form-filling. GIS made use of a shared dictionary and shared databases. Quickbuild Pathway ignored this possibility.

11.5. Conclusions

The User Trip confirmed the existence of a prototyping activity used to validate software designs within the company. This activity was in-house, however, and conducted in a less than ideal environment in the case of the product investigated. The User Trip by no means gave exhaustive experience of the company's product line, or of its software validation activity. Nevertheless, valuable learning took place. It became evident that software products were far more than code. Documentation, training and support all formed part of the product set, and products were linked into systems of products. Usability was a complex

phenomenon and was inextricably bound up with functionality. Whether or not a product was used was not solely to do with usability (or functionality for that matter). To a certain extent the computer manufacturer could enforce use of software by linking items together in a way that ensured that if one product was used, then others had to be. Informal conversations with the Quickbuild design team made clear the risks of designers being cut off from users of their products. Project schedules did not permit effective prototyping, or even informal contact with customers. Designers were not permitted contact with prospective customers. This was left to marketeers.

CHAPTER 6 - EVALUATION

1. Introduction

This Chapter performs evaluation in three areas. Firstly, the application of the OU Design Methodology will be evaluated. Secondly, the paradigm of Radical Design will be evaluated with respect to its suitability for use in systems of human activity and for bringing about change in coercive situations. Finally, the contingency framework used in this research to select a problem-solving methodology will be evaluated.

2. Evaluation of Methodology Application

The contention of Cross (1982a) and Darke (1979) that design is a solution-focused activity has been confirmed in this research. The solution focus changed several times during the project, however. At the end of the problem identification process the focus was on a methodology for designing user-revisable software. The input to problem identification had been Nissen's (1984) statement describing information systems research as positivist, and performed on behalf of employers to 'improve' the design and implementation of information systems. This statement was felt to characterise the 'official' Alvey-funded project of which the author was nominally a part. The 'solution' that emerged from problem identification was not directly concerned with positivist-orientated employer-dominated research, but rather with the employer's initial problem statement concerning software usability. The connection between the input to, and the

output from, the problem identification process, is that applied externally-funded research of any sort is bound to contain coercive elements (whether acknowledged or not) due to the nature of society, and the usability of products by consumers or employees is bound to be affected by the same contradictions. Current methods of production often fail to produce products that best suit the interests of consumers or end-users. It is contradictory for a producer of products to sponsor research on behalf of the users of its products. The proposed solution, of facilitating the revision of unsatisfactory products by the users themselves, may be applicable in some circumstances, and is certainly technically feasible in the case of software. It may be impossible, however, to divorce a software product from its context of use on some task system, and both systems may need to be revisable to bring about any improvement in the user's lot.

A loss of confidence in the proposed solution was experienced during the design briefing process that followed problem identification. The construction of objective trees and SSM-style conceptual models encouraged a more holistic view of the problem context, and developed the idea that the problem was structural, rather than local to one producer of a product. The counterplanning exercise generated several shifts of solution focus, and this is the intended purpose of the method. During counterplanning the following solution foci arose:

[1] A national system for software production and use.

[2] A system to produce software rationally, from the point of view of the computer manufacturer.

[3] A manufacturer-owned system to produce designs for marketable products.

[4] A manufacturer-owned system to monitor and control the production of designs.

The focus of design had become increasingly conservative during counterplanning. The process is recommended to be self-consciously dialectical, with the tension between plan and counter-plan generating an appropriate synthesis. The plan in this case may be characterised as radical humanist; the counter-plan was radical structuralist, and yet the synthesis was conservative. The conclusion reached at the end of counterplanning was that the problem to be solved lay somewhere inside an internal company system. While this conclusion is disappointingly conservative, it was perhaps necessary to believe it while conducting a user trip *inside* the computer manufacturer. The manufacturer had already defined the problem as owned by itself, and it it was difficult to resist this idea. Experience gained on the user trip increased subjective understanding of software usability, and for the first time crystallized ideas as to the nature of usability. It became clear that a software product was not a single entity, but a complex interaction between code, hardware, documentation and training course, all of which were components of the product. Usability was also a complex phenomenon resulting from an interaction between a product, a user and a task. Attempts to improve software usability would have to take into account the complex nature of these systems.

Intrinsic to design activity, according to Cross (1982a), is the use of codes to refer to objects of design. Engineering drawing is an example of the use of such an object code. Systems designers use flow diagrams, and diagrams representing relationships between the structural components of a system. The OU Design Methods Manual did not recommend any

particular notation for design. This was because the methods were intended to be independent of any particular application area, such as engineering or architecture, and intended to be equally useful when designing artefacts or systems. The absence of an object language or design notation proved to be a problem when applying the design methods. Initially, an objectives tree was thought of as a design language, but this proved to be confusing and restrictive. Confusion arose between the objectives of the project being undertaken, and the objectives of the system being designed, and the diagrams produced were sometimes a mixture of both. The objectives tree was too restrictive a model for systems design, because a tree is a hierarchy and systems are networks. At the stage of drawing objective trees it was not clear what to take to be a system relevant to solving the problem being worked on. For this reason, and also because of its richer modelling language, SSM root definitions and conceptual models were used. The SSM conceptual model seems to be a satisfactory language for general purpose systems design. Checkland (1981) disassociates use of SSM from design, possibly because design is associated with hard systems engineering. SSM conceptual models are not supposed to be designs, but ideal-type models of systems defined from a particular point of view. Their purpose is for use in comparison with real-world activity, and the initiation of a debate among interested parties. It is nevertheless clear that an SSM conceptual model is a flexible and powerful design notation. The issue to be resolved is whether a design paradigm is appropriate for systems of human activity, or whether an interpretive paradigm, such as underpins SSM, is necessary. The issue of notation is trivial in comparison with the philosophical issues at stake. These will be addressed in the next section of this Chapter.

Before leaving the evaluation of the application of the OU methodology, however, some attention will be given to whether it 'worked' - that is whether it produced a design for something that if implemented would alleviate the problem being researched. The answer is clearly 'no'. Before this could be established in any project, however, it would be necessary to implement the design and evaluate the implementation. Evaluation of systems implementation is not a trivial exercise, and there is a tenuous link between the success of a system and the methodology used to specify and design it. Even if a system can be credited with some degree of success and some measure of this can reasonably be attributed to methodology, then it is still not clear whether application of the same methodology by different people would not have given a different result, or whether another methodology might have done even better. So at the level of systems design it is difficult to prove that any methodology objectively 'worked'. Subjectively it can be reported that this methodology is fun to use and generates creative thinking and valuable learning. At the end of the process of using the methodology the author had formed a richer picture of the problem situation. The sequence of activities to be followed when using the methodology is somewhat incoherent. The discontinuity between Problem Identification and Design Briefing is artificial. The Project Potential Phase would be more logical after Design Briefing rather than at the end of Problem Identification.

The methodology was chosen because of its origins in the radical design movement, and because the problem context was felt to be coercive. Coercion did not lessen as a result of using the methodology, and it contains no specific methods for dealing with it. The subjective discomfort of being subjected to coercion lessened, however, because use of the

methodology took the form of a protest. From the point of view of other members of the research team though, this member was regarded as a drop-out and a trouble-maker. The lesson learned here was that individual acts of protest are ineffective in the face of structural coercion, but preferable to supine submission.

3. Evaluation of Radical Design

The answer to the question of whether a design paradigm is suitable for intervention in systems of human activity, is that it depends on who is the designer. This is compatible with the stance of Checkland (1981) who finds the deficiency of hard systems approaches to be the assumption that a system is required to meet agreed objectives. Where this is true it perhaps does not matter who produces the design. If the boundary of design is widened to include the setting of objectives, however, and if a principled objection is raised to primacy being given to the viewpoint of a systems designer, in comparison with the viewpoints of the systems agents, customers or victims, then a *conventional* design paradigm must be rejected. Before a *radical* design paradigm can be embraced, however, the issue of whether it is appropriate to regard human activity as systemic must be addressed. Designers conceive of *objects*, whether physical or abstract, and if human activity *systems* do not exist then a design paradigm is not appropriate for intervention in human affairs. Checkland takes the view that an interpretive account of human activity is preferable to one that stresses the existence of systems in the world. This is because human activity systems are not material objects but abstractions used to explain, justify or plan human action. Each participant in human activity has his own view of its purpose, and of suitable means to achieve it.

Successful collaborative activity implies negotiation about ends and means, and interpretation of the meaningful actions of others. Human activity has structure as a result of the ongoing prosecution of purpose accompanied by ongoing discourse about desirable action. Structure is created by process and is not a determinant of it. Lilienfeld (1978) has pointed out the ideological function of systems theory in legitimizing particular social structures, and criticised systems-based accounts of human activity as being inherently conservative. Silverman (1970) denies the relevance of the biological analogy that he sees underlying systems-based accounts. Beer's (1972) application of systems theory to organisations, for example, was originally heavily dependent on biological analogy, as indicated by the title of his book 'The Brain of the Firm'. Beer's later work (Beer 1985) in developing the viable systems model (VSM) avoids reductionist analogy. Bowey (1980) points out that a social system should more appropriately be compared to a species than an organism. The crucial difference between an organism and a species is the presence of cybernetic control in the organism. The idea that cybernetic control by negative feedback loop explains the persistence of organised human activity underpins Beer's VSM. Other organisational theorists have made use of a cybernetic analogy, for example Argyris and Schon (1978) in discussing *organisational learning*, suggest that it may be regarded as *single-loop* (revision of a theory of action about how to achieve organisational goals), or *double-loop* (revision of organisational goals). A cybernetic model of system control applied to human beings is conservative because it provides no theory to explain or justify radical change, and also ideologically regulatory in that it can be used to justify inequalities in power relations between people. Checkland retains the concept of cybernetic control to

refine ideal-type models of human activity, but rejects the idea of structural determination of human activity. The systems paradigm is epistemological as far as Checkland is concerned, not ontological. Reification of human activity systems is a linguistic device that may have ideological function, or may simply be a shorthand way of referring to complex processes. Clearly such systems do not have material existence and, according to some writers of the soft systems tendency can be only subjectively defined. Carter, Martin, Mayblin & Munday (1984), for example, define a *soft system* as:

> 'A system depending largely on non-routinised human actions, so that human capacity for free choice, and the agent's limited access to the subjective values, beliefs and wishes of the participants means that wholly objective description or quantitative modelling are not appropriate.'

The philosophy underpinning this position is objective idealism - human activity systems exist but objective knowledge of them is unobtainable. Ideas about the system determine its nature, and the ideas of others can never be securely known. Checkland's position corresponds to philosophical idealism. He defines the word 'system' (Checkland 1981) as 'a model of a whole entity' and says that 'an observer may choose to relate this model to real-world activity.' Different observers might choose to apply different models dependent on their world-view. So 'systems' belong entirely to the realm of ideas not to the world of things.

The objection to an idealistic account of human activity - that it can be only subjectively interpreted and that ideas determine structure - is that human activity may be shaped by forces and historical circumstances of which all the participants are unconscious. Actions may have unintended consequences, and participants accounts of intention and purpose may be shaped by ideology. In some cases the ideas of the participants in human activity are crucial to defining what goes on - for example, two

actors pretending to quarrel in a play are not quarreling, however realistic their performance. In other cases human beings are swept along by social forces that they barely understand, for example in war or revolution, and an objective specification of events is valuable independent of individual interpretation.

The argument about the relevance of the design paradigm to human activity has proceeded as follows. Since designers need to design objects that are capable of objective existence (albeit as abstractions), then if systems of human activity do not exist the design paradigm is irrelevant. Some critics of systems theory applied to human activity concentrate on the type of system perceived (cybernetic models are rejected), and some concentrate on problems of subjectivity in identifying systems whose components are abstractions. These latter problems can seem to justify the espousal of some form of idealism. Idealism brings problems associated with erosion of the concept of truth - it provides no way of distinguishing between an ideologically distorted account, and one that corresponds to facts that remain true whatever the ideas of individuals may be. An acceptable ontology for social enquiry and intervention, therefore, must allow for the objective existence of systems and systems-generating mechanisms. This corresponds to the philosophical position known as realism. The conclusion must be that a design paradigm could in principle be relevant to intervening in social systems.

Simply because the objects for a hypothetical design process exist is not sufficient guarantee that such objects are designable. Many objects exist that are not thought to be suitable objects for design (for example living creatures). Man can intervene in the natural world but not fundamentally redesign it, the argument might run. Successful interventions in

the natural world are underpinned by powerful and well-tested scientific theory. The social world might be 'natural' like the objects of study of physics and chemistry. As in physical science there would be 'laws' of social science that no social engineer or systems designer could ignore. To the extent that social action was consonant with those laws, then it would be a consequence of them and not of freely chosen action. Action not consonant with the governing laws would inevitably fail to achieve its purpose. Structuralist accounts of social action follow this model. So, for example, any attempt to organise human activity that ignored the principles underpinning the VSM would fail, assuming that structure does indeed determine human action, and that Beer has correctly defined relevant structural principles. The issue is whether human activity systems exist because of the determination of people (perhaps designers) that they should, or whether they exist because of structure generating mechanisms in society. If the latter is true then it will be crucial to gain understanding of those mechanisms, and the limits they place on human action. It is possible that the process of design as a human activity is the outcome of some structuration process. The issue then is whether design can produce feedback to the process that generates it. The distinction between natural and designed systems has as its heart a methodological dualism that demands that a qualitative distinction be maintained between the natural and the social sciences. If man is securely part of the natural world, then his design activity is a natural phenomenon, as is all human activity.

Location of design activity within a structuration process enables appropriate consideration to be given to radical as distinct from conventional design. The latter may be compared with systems engineering,

convincingly ascribed by Checkland (1981) to a hard-systems paradigm. Conventional design assumes agreement over objectives and places design in the hands of experts in particular technologies, such as engineering or architecture. People described as *systems* designers are liable to be systems engineers, computer scientists or management scientists. The mission of radical design was to reduce the need for professional designers by defining general design methods suitable for clarifying requirements in any application area, and teaching these methods in a widely accessible way. Open University courses, for example, impose no entry qualifications and can be studied by those unable to attend a conventional educational institution. The OU T262 course was radical to the extent that it was devised by people dissatisfied with conventional technology and the direction taken by modern industrial societies. The OU T262 course was also radical in its intention to empower students to act to bring about change in society, by providing a problem-solving methodology. The course signalled problems to students and provided tools to help in their solution. The weakness of the course is the absence of a coherent theory of what gives rise to exploitative and alienating technology, and how fundamental social change may be brought about. Cross (1982b) describes the role of radical design in bringing about a more desirable future. He views design as a process that can be influenced and controlled to produce different products. Engineers can be educated to resist the socially irresponsible demands of industry. Alternative technology necessitates an alternative design process that eliminates the distinction between designer and user. Designers must realise that

> 'we are all users, that we are all lay-people, that we are all dominated by the design process.'

The essential weakness of this position is that it determines to bring about

change by changing men's ideas. The practical action that might ensue from 're-design' of unacceptable features of society is individualistic. Radical design methodology liberates the mind, but has no component that empowers successful collective action.

4. Evaluation of Jackson and Keys Framework

Three aspects of the contingency framework of Jackson (1987a) and Jackson & Keys (1984) will be evaluated here. These are:

[1] The success of the framework in producing an insightful categorisation of problem-solving methodologies.

[2] The success of the framework in making appropriate recommendations for choice of problem-solving methodology.

[3] The soundness of the philosophical underpinning of the framework.

The framework is theoretically underpinned by Habermas' distinction between the human technical, practical and emancipatory interest. Human technical interest lies in the understanding and control of natural and social systems. Some systems can be studied empirically and mathematical models of them constructed. The understanding gained by this form of modelling enables man to intervene effectively in the world. This represents the paradigm of operations research and systems engineering. Jackson & Keys describe systems that can be optimised or engineered as 'mechanical'. Progress cannot be made with an operations research or systems engineering project unless the objectives of the system being studied or designed are clear. Choice of desirable ends for human activity cannot derive from man's technical interest alone. The human practical interest lies in sustaining satisfactory social relationships without which collaborative activity performed to pursue technical interest cannot take

place. Agreement about suitable purpose is a social not a technical phenomenon. Jackson & Keys describe a problem context as mechanical-unitary if a simple system is to be optimised or engineered, and there is agreement about its purpose. Where there is lack of agreement but there is potential for consensus, the context is described as mechanical-pluralist. Where not all the participants have equal power and so do not equally contribute to the decision-making, the context is described by Jackson as mechanical-coercive. The need to be free from coercion derives from man's emancipatory interest.

Some problems with the framework may be observed at this point. Operations researchers and systems engineers have always been conscious of the need to clarify objectives, and this clarification is part of OR and SE methodology. It cannot be predicted in advance of starting a project quite what difficulties might be encountered in resolving differences of opinion. No description of OR and SE constrains the process to unitary situations. De Neufville and Stafford (1971) in a classic formulation of the OR/SE paradigm state that definition of objectives is the first phase of the process of what they describe as 'systematic analysis'. Their account of objective-setting includes consideration of pluralist situations -

> 'Much of the value of systematic analysis lies in the identification of objectives and the clarification of issues, not in their concealment.'

The essence of the criticism, then, is that a distinction between mechanical-unitary and mechanical-pluralist contexts does not adequately partition problem-solving methodologies. OR/SE is intended for both types of context, and attempts to bring about a transition from a pluralist to a unitary situation. The history of OR/SE demonstrates a decreasing emphasis on mathematical modelling, and an increasing concern about strategy and objective-setting. The OR/SE that is located by Jackson and

Keys in the mechanical-unitary category, is the OR/SE of the 1960's not the 1980's. An objection that can be raised to the mechanical-coercive category is that it is empty - no methodologies are available for coercive contexts. Jackson (1987a) locates Ulrich's Critical Systems Heuristics in this category. The critical component of Ulrich's work is that he insists that *all* those affected by proposed change in society should have their interests considered - not just those that powerful people see fit to consult. The method he proposes to use to ensure that all interests are considered is polemical debate. He proposes no way of equalising power differences between participants in the debate. In a coercive situation, moreover, those with power control the agenda of the debate. There seems insufficient justification to discriminate Ulrich's work from that of Checkland, Ackoff and Churchman, all of whom are concerned with bringing about accommodations in pluralist contexts.

So far we have considered methodologies suitable for problem-solving in the analysis and design of mechanical systems. It is arguable that there is just *one* paradigm applicable here:

[1] Establish requirements.

[2] Build a model of a system to be optimised or constructed.

[3] Implement the solution implied by the model.

The relationship between those affected by the system will determine whether the context is unitary, pluralist or coercive. This will affect requirements determination. The history of OR/SE shows an increasing concern with establishing objectives in pluralist contexts. In coercive contexts the interests of the powerful are liable to prevail, and there is no problem-solving methodology that can solve this meta-problem. The categorisation of methodologies suitable for systemic contexts will now be

considered.

Some systems are too complex to be amenable to mathematical modelling and their functioning cannot be optimised. A model of such a system is necessarily 'conceptual' rather than analytic. The conceptual models used in SSM, for example, are linguistic rather than mathematical. An SSM conceptual model is an abstract system where each component is an expression of purpose - it defines part of a purposeful system. The purpose is expressed in natural language. Reliance on abstraction is one way of solving the problem of complexity when analysing or designing systems. To use purpose as a principle of abstraction is commonplace in engineering - a complex assembly is thought of in terms of its function with respect to other systems components. This abstraction is useful in design even if the system or sub-system *is* amenable to mathematical modelling, and may be essential if it is not. If a design for a system is presented as a conceptual model reliant on some principle of abstraction, the question arises as to whether this design will meet requirements or bring about desirable ends. The question of validity is answered in OR/SE by 'running' the mathematical model and proving that the results are optimal. The answer provided by Beer is that a design will be satisfactory if the principles embodied in the VSM are followed. Beer's position is that particular systems may be too complex to model, but there are universal structural principles determining the functioning of complex systems. If these principles are understood and applied in design, then the absence of a mathematical model is not important. Structuralist accounts of human activity do not usually stress the importance of subjectivity in defining systems (one such account that *does* is that of Espejo (1987)). Checkland's position is that human activity systems do not exist in the

world, but are ways of perceiving and making sense of the world. An SSM conceptual model, therefore, is correct to the extent that it is a defensible derivation from a root definition. Whether a root definition defines a system that will meet human requirements, or bring about desirable ends, depends on the *Weltanschauung* of the person considering it.

The Jackson & Keys framework allocates Beer's VSM to the systemic-unitary category. This is because Beer's work is not primarily concerned with resolving conflict over desirable purpose. Checkland's SSM is allocated to the systemic-pluralist category. This is because Checkland's subjectivist stance makes SSM in principle suitable for modelling systems from diverse points of view. SSM also encourages debate among interested parties. There is no reason to suppose, however, that SSM is not suitable for systemic-unitary contexts and mechanical-pluralist contexts. Checkland has also argued the relevance of SSM to coercive contexts (see the debate in Chapter 2, Section 6). So the Jackson & Keys framework does not seem to securely locate SSM. With respect to the 'coercive' column of the contingency matrix, no methodologies are located in the systemic-coercive category. Jackson (1987a) suggests that 'an approach based upon radical structuralism is more apt in systemic coercive contexts', but the problem is that no methodology exists that takes such an approach.

The Jackson & Keys framework is suggestive but not ultimately persuasive in categorising problem-solving methodologies. This is perhaps because the authors were not primarily concerned with constructing a taxonomy and illuminating what exists, but with constructing a framework for action, and pointing in some cases to what ought to, but does

not yet, exist. Jackson's accounts of the framework indicate its status as a contingency scheme -

> 'Mechanical-unitary contexts require traditional management science. Systemic-unitary contexts require treatment from organisational cybernetics. Mechanical-pluralist and systemic-pluralist contexts are best tackled using soft system thinking. Critical management science should be employed to deal with mechanical-coercive and systemic-coercive contexts.' (Jackson 1987a).

The conclusion must be that Jackson intended the framework to be underpinning theory for choice of problem-solving methodology (as indeed it was assumed to be in this thesis). In Jackson (1987b) a number of case-studies are presented deriving from the work of the Community OR Centre at Hull University. The Jackson & Keys framework was clearly used as a means of choosing a suitable methodology to tackle particular problems. When the framework was used in *this* research the problem that was encountered was that *all* social systems seemed to be complex and coercive. To make use of the framework a subjectivist stance had to be adopted, where problem contexts were not assumed to have objective characteristics but were essentially *however you chose to view them.* So one person might see a mechanical-unitary context and another might see a systemic coercive-context and yet they might both be investigating the same problem. The success of the framework as a contingency scheme, therefore, depends on the soundness of its philosophical basis. If the subjectivist approach is correct then it is not meaningful to ask whether the framework successfully chooses appropriate methodologies for particular problem contexts. A subjective interpretation of the course of a project is all that can be expected.

Subjectivism is implicit in the Jackson & Key's framework. The paradigms underpinning some of the methodologies categorised by the framework are opposed to each other. For example, OR/SE is implicitly

realist in its assumption that systems have objective existence. SSM denies the objective existence of systems. Beer's VSM is a structuralist model and structuralism is inherently deterministic in contrast to Checkland's voluntarism. Critical management science is in part inspired by the Marxist philosophy of dialectical materialism. Marxist theorists would diagnose ideology in the OR/SE, cybernetic and soft systems paradigms, and proponents of these would diagnose dogma in the work of Marxists. The essential point is that the methodologies, approaches and philosophies categorized by Jackson & Keys are theoretically incompatible. If methodologies A, B and C each make incompatible philosophical assumptions, then it cannot be rational to argue that *under some circumstances* it may be correct to choose A and *under other circumstances* correct to choose B. The only way to refute this criticism is to adopt a subjectivist stance that maintains that there is no absolute truth. Truth has meaning only within some theoretical framework - so there can be positive truth deriving from a positivist epistemology, and interpretive truth deriving from an anti-positivist epistemology. A subjectivist defence of the Jackson & Keys framework is clearly possible, but it imparts a serious weakness to the process of methodology choice. The action researcher is invited to behave as if there were objective truth to be had about the complexity of the system being studied and the relationship of the participants in it. On the basis of what is true about these factors a problem-solving methodology is chosen. The contradiction can arise, however, that the methodology chosen can appear to make theoretical assumptions incompatible with the theory used to choose it. For example Checkland's SSM is underpinned by the assumption that there is no objective definition possible of a problem situation. The action researcher, however, was acting only 'as if' there

was objective knowledge to be had to guide methodology choice. In which case it is impossible to judge what would count as a correct choice of methodology. The contradiction at the heart of the Jackson & Keys framework is that if methodology choice is based on objective criteria then a methodology underpinned by subjectivism cannot rationally be chosen; and if methodology choice is based on subjective criteria then a methodology that depends on objective truth cannot rationally be chosen.

Jackson (1990) claims to see a way out of this labyrinth. The categorisation scheme must *not* be used to establish the objective nature of a problem context in terms of its complexity or the relationship between the participants. It should be used to explore the consequences of adopting any particular methodology. Any problem context can be viewed in a variety of ways, ranging from mechanical-unitary to systemic-coercive. The consequences of adopting different viewpoints should be explored, and different stakeholders may come to different conclusions as to the right approach to adopt.

By taking this line Jackson denies that there can be an objectively 'correct' choice of methodology for some problem context. There remains the problem, however, of allowing subjective or contingent choice of an approach that insists that there is objective truth to be known about the world. For example, an employee may see a situation as coercive and an employer may not. If the employee is a Marxist she will believe that the coercion is objectively and demonstrably present in the structure of society, and not merely a subjective response to society. Furthermore, a Marxist will be impatient with the notion of following a tortuous contingency procedure with respect to a series of problems over time, say in industrial relations, where the possible existence of pluralism or coercion

is deliberated upon. To a Marxist the existence of coercion is taken for granted, and to treat it as an open question to be decided afresh on each occurrence of a problem is a-historical. Similarly, to a managing director, the request to view reality through the eyes of a Marxist is liable to be unacceptable. In fact, Jackson's ideas are acceptable only to those who do not adhere to what he describes as an 'isolationist' or an 'imperialist' stance. These categories, along with those of 'pragmatist' and 'pluralist' are defined in Jackson (1987d).

An isolationist strategy is one that rests upon a particular paradigm, and rejects or ignores all others. For example, it would be isolationist to analyse all problems by attempting to build and optimise a mathematical model of the system argued to contain the problem. The problem of paradigm incommensurability does not arise from an isolationist standpoint.

An imperialist strategy rests upon a particular paradigm, but seeks to ensure that the paradigms of other approaches are subsumed by the dominant one. Checkland, for example, argues that hard approaches are a special case of a soft approach.

> '--- the relation between "hard" and "soft" systems thinking is not like that between apples and pears: it is like that between apples and fruit. The well-defined problem needing solution is the special case within the general case of issues calling for accommodations.' (Checkland 1985).

A pragmatist strategy is distrustful of theory and concentrates on gaining proficiency in what works well in practice. A pragmatic approach may be taken, for example, by OR practitioners in commerce and industry, or by academics who act as consultants.

Jackson rejects isolationism, imperialism and pragmatism and recommends what he describes as pluralism. Using this strategy, methodologies deriving from different paradigms are respected and studied, and meta-

theory is developed that allows choice among the competing approaches. A pluralist must have difficulty reconciling the Marxist perspective, arguably the most historically significant reaction to systemic-coercive situations that has been made. Marxism is imperialist in that is based upon the philosophy of dialectical materialism, and explains systems-based problem-solving methodologies as responses to the needs of capitalism. Jackson must either accept Marxism as a valid response to systemic coercion, and such acceptance is simultaneously called for by the pluralist position and destructive of it, or he must reject it. If he rejects it then the systemic-coercive category of problem context appears to be completely intractable with respect to problem-solving methodology.

CHAPTER 7 - SUMMARY & CONCLUSIONS

1. Summary

This thesis describes action research into a soft problem relating to software production and use. The problem relates to the usability of software intended to perform specific tasks, when used by specific users, in a specific environment. The problem arises because, for commercial reasons, general-purpose software products are developed for a wide market, not for specific customers. The problem was presented by a computer manufacturer that supplied software products with its machines. The manufacturer wished to improve the usability of its software. Important actors in the software production process were marketeers and designers. The manufacturer felt that there was a communication gap between marketeers and designers. Marketeers were supposed to be close to customers and end-users, and generated product specifications for implementation by designers. These product specifications were vague, ambiguous and incomplete from the designer's point of view. From the marketeer's point of view, designers were technicians out of touch with the market-place.

The research project was funded by Alvey, and controlled by a project manager from the computer manufacturer. This person rapidly supplied a problem analysis and plan for the problem solution, and attempted, largely successfully, to force the academics associated with the project to implement this plan. The project manager's solution to the

problem was to run training courses for software marketeers and designers. The project manager's approach to the research and his problem solution, is treated as part of the problem in the research described in this thesis.

The areas surveyed to establish a body of knowledge relevant to the research were software engineering, information systems development, design methodology, systems methodology and contingency theory for problem-solving. A radical design methodology was used in this research rather than SSM, which might have been an alternative. The contingency theory of Jackson (1987a) influenced this choice. The reason for the choice lay in the nature of the problem situation, the control that the computer manufacturer had over the research, and the unwillingness of the project manager to allow the project to deviate from a pre-planned course. Successful use of SSM requires a willingness to seek reconciliation of opposed world-views on the part of the participants. This willingness did not exist in the research team, and the atmosphere was coercive. Use of a radical design methodology meant that the author of this thesis effectively withdrew from the Alvey project, and took an individualistic approach.

The methodology used derived from the OU, and involved use of a Problem Identification Game, followed by application of some design methods chosen according to a loose contingency structure. PIG was initially addressed not to the problem of improving the usability of software, but to the problems perceived in the project set up to investigate the problem. The process of enquiry was seen to be problematic because what purported to be academic research was under the control of a commercial firm with no concern for academic values. The analysis generated by

playing the game indicated that the core problem associated with the project was the fact that the computer manufacturer's solution to the problem (training courses for software marketeers and designers) excluded software users. An ineradicable distinction between designers and users was likely to be perpetuated by the project in its present form. One of the objectives of radical design is to abolish the distinction between designers and users.

Problem identification was followed by the application of three design methods, Objectives Tree, Counterplanning and User Trip. During the attempt to define a hierarchy of objectives for the project (an Objectives Tree), SSM root definitions and conceptual models were constructed. This activity can be regarded as the eclectic incorporation of an additional design method. Counterplanning requires the reversal of the assumptions that have underpinned the project so far. It is a dialectical process that is intended to result in the synthesis of a new plan. The counterplanning process was supplemented by further use of root definitions and conceptual models. The User Trip took the form of secondment to a team of software specialists who were prototyping a piece of software shortly to be released by the computer manufacturer. A subjective understanding of usability issues and the organisational features of software production was enhanced during this period of time.

2. Conclusions

The conclusions drawn from this piece of work will be categorised under the headings of:

[1] Software Usability.

[2] Radical Design.

[3] Contingency Theory.

[4] The USTM Project.

2.1. Software Usability

Usability is a nebulous concept. A product or product feature is presumably usable if it is used - but it may be used because there is no adequate alternative, or because the user has been coerced into using it; its use may give rise to frustration and inefficiency. A product or product feature that is not used is not necessarily unusable. The user may have received inadequate training, may not understand the documentation or may be resisting changes in work practice for other reasons. Eason's suggestion (Eason, 1984) that a user's reaction to a system is the result of a complex interaction of system functions, user characteristics and task characteristics was found valuable in this research. Usability is an emergent property of software in use by a particular person for a particular task. Usability is not a property of software in isolation.

It may be misleading to focus attention on a single sôftware product when analysing usability. A user is performing a series of tasks throughout a working day. More than one software product might be used during the day, and more than one product might be used during one task. The user's discretionary power over which product or product feature to use will vary from one product to the next. From the users point of view a software *system* is being used, and it is the usability of the overall system as well as that of any particular product that is important. For example, it is possible to buy separate spreadsheet, word-processing and DBMS packages. The usability of all three packages will be greatly enhanced if they share a common interface. Software vendors

need to concentrate more on defining, modelling and supporting user tasks than on producing software as traditionally conceived.

The structure of economic activity, and the software business in particular, is not conducive to the production of well-designed products that support human tasks. There is irrational variability in administrative, commercial and industrial systems, given a capitalist mode of production. The information technology industry is similarly irrationally fragmented. In the case of computer systems, for example, there are many incompatible models of computer on the market, and complete software systems have been developed for each competing range. There is also irrational variability in systems software. Scarce design talent is misused by replicating software products for incompatible machines and incompatible systems software. The user's time is wasted learning different interfaces to different products. More resources could be allocated to meeting usability requirements if there was less irrational product differentiation on the part of software vendors. There is also irrational variability in requirements due to lack of systems standardisation in user industries.

Software products *inherently* bring with them usability problems because of the distance of the designer from the end-user. A model of software specification that implies elicitation of user requirements is inappropriate. There are too many potential users to make this feasible, and at the time of software specification it cannot be established with certainty *who* the users will be. The software may be intended to support tasks that are possible only because of the existence of the software. E-type systems create requirements as well as responding to them. Only if users become designers can this problem be overcome. This solution is unrealistic on economic grounds - software products are cheaper than

bespoke systems. It is also politically naive - IT products may be intended to produce job degradation, as for example in the printing and publishing industries.

From a software vendor's point of view, usability is a problem only insofar as it impacts on costs and revenue. If a product is bringing in maximum achievable revenue the vendor is unlikely to be concerned with improving its usability, unless maintenance costs attributable to poor usability are unacceptably high. In a seller's market maintenance costs can be kept low by deferring requests for enhancement. Enhancing product usability is likely to to increase the vendor's costs. A product's market success may well depend on its perceived usability, however, particularly in a competitive market. Software vendors sell to corporations not to direct end-users. The interests of the two are not the same. Direct end-users wish to work at tasks that are interesting and do not impose undue stress. Badly designed software can be a considerable cause of stress and frustration. Automation may bring redundancy and job degradation.

In this research the problem of software usability was found to be a soft problem and not, therefore, the sort of problem amenable to 'solution'. By the end of the research a richer picture had been formed of the problem situation.

2.2. Radical Design Methods

Radical Design has two essential characteristics. Firstly, it is committed to the belief that there is a common process underlying all design practice, and that design is the bringing about of change in the man-made world. Design covers a wider sphere of intervention than classical civil, mechanical or electrical engineering, or newer branches of

engineering such as systems and software engineering. It includes interventions in systems of human activity. Not only is the scope of design very wide from the radical perspective, but the process of design is extended to include making decisions about what to design. Secondly, radical design is committed to the task of abolishing the distinction between designer and user, and empowering users to act in the design process. This aim would not be achieved by merely bringing about user participation in design, because user participation does not challenge the conventional design process. Bringing about fundamental change in the design process is important from the radical perspective because the *weltanschauung* of radical design implies rejection of the values of industrial society, based as they are upon careless exploitation of natural resources, pollution of the natural environment and alienating work.

Historically, radical design has similar origins to soft-systems methodology. Both arose in the 1970's as a reaction against conventional hard-systems engineering. Soft-systems methodology, in particular the work of Checkland, has a much higher content of explicit social theory. The absence of social theory from radical design weakens its potential for effective collective action. Use of radical design methods inspires creativity but encourages an individualistic response to problem situations. Checkland's SSM is better equipped to produce effective collective action, but at the risk of it being based on a false consensus in coercive situations. The strength of radical design is its explicit commitment to a value system that causes it to take certain types of problem seriously - that is, in urgent need of solution, rather than being a particular way of looking at the world. The implication of a soft systems approach is that in pluralist situations accommodations should be sought. The implication of a

radical approach is that in certain types of pluralist situation there is a *correct* point of view to be held.

2.3. Contingency Theory

The contingency theory of Jackson (1987a), based on that of Jackson & Keys (1984), was used in this research. Jackson attempts to categorise problem-solving methodologies into six categories, each of which corresponds to a type of problem context. The categorisation scheme appears to have been intended as a means of choosing a methodology appropriate to a particular problem situation. The categorisation is not completely convincing because some methodologies seem to be relevant to more than one category. A major problem arises in using the categories to choose an appropriate problem-solving methodology. This is the problem of paradigm incommensurability defined in Chapter 6. From a radical perspective the social world appears objectively coercive. It is not acceptable from this perspective to *choose to view the world otherwise.* Only from a subjectivist perspective is it rational to choose among methodologies that make opposed philosophical assumptions. It is, however, paradoxical to choose a methodology that denies the validity of a subjectivist stance.

2.4. The USTM Project

The claim has been made in Chapter 2 that this thesis describes an action research project. The 'official' USTM project was not action research but rather *consultancy,* because what was allegedly previously established scientific knowledge was being packaged, and presented to people who arguably needed to know it (ie marketeers, designers and

technical authors). The work of Mansell was an *attempt* to perform action research in an inimical environment. In terms of Susman & Evered's (1978) model of action research, comprising action planning, action taking, evaluation and specification of the learning that has taken place, the greatest problem that Mansell encountered was in the *taking of action.* In the spirit of action research this should comprise an intervention in the problem situation informed by relevant theory. Having defined the problem situation as coercive, and as a result of this reaction to what was going on having impaired the relationship with the client and the other problem-solvers, Mansell had made action difficult to take. In this respect the situation can be compared to that of Berry et al. (1986) described in Chapter 2. If would-be action researchers reject the primacy of the client then they may find it impossible to complete the project in the ideal-typical manner. The *action* components of Mansell's work consist of the User Trip, which put Mansell in the position of a problem-owner struggling to use a software package, and publication of the results of the research. The latter *is* a form of action, and may be the only one that it is possible to take in a coercive situation (as in the case of Berry et al. 1986). For example Mansell (1989) delivered a paper at an academic conference describing the problems encountered on the project. The reaction to the verbal delivery of this paper was hostile. Rejection of client-dominated research seemed to trigger deep emotions in academics. One reaction, from a senior academic, was that Mansell had ruined a perfectly good project by his intransigence.

If publication is to count as action in an action research project then it becomes difficult to say at what point the project ends. In conventional scientific research submission of a thesis marks the end of a piece of

research. In the case of action research it could mark a new beginning.

3. Future Research

This piece of work has exposed the inadequacy of design methodology, however radical in intent, in making progress in coercive situations. Jackson's work alerts the practitioner to the existence of coercion, and is helpful in analysing the intellectual origins of alternative methodologies for action research. It provides little guidance, however, in how to proceed in coercive situations. Its subjectivist orientation gives rise to contradiction when guiding methodology choice. It may be more valuable in the future, therefore, to concentrate on enhancing some existing methodology, rather than tolerating a profusion of inconsistent and contradictory approaches. For example, recent elaborations of SSM (Checkland & Scholes 1990) give the impression that this methodology is healthily responsive to change. Enhancement of SSM to cater for systemic coercion would be a worthy research project for the future. In Jackson's terminology this would be an example of imperialism rather than his favoured alternative of pluralism.

REFERENCES

Ackoff R. L. & Sasieni M. W. (1968), *Fundamentals of Operations Research*, Wiley.

Ackoff R. L. (1981a), "The Art and Science of Mess Management", *Interfaces* 11, pp 20 - 26.

Ackoff R. L. (1981b), *Creating the Corporate Future*, Wiley.

Argyris C. Putnam R. & Smith D. M. (1985), *Action Science*, Jossey-Bass.

Argyris C. & Schon D. (1978), *Organizational Learning*, Addison-Wesley.

Asimow M. (1962), *Introduction to Design*, Prentice Hall.

Avison D. E. (1985), *Information System Development: A Database Approach*, Blackwell Scientific Publications.

Banathy B. H. (1988), "Matching Design Methods to System Type", *Systems Research* 5, pp 27 - 34.

Beer S. (1972), *The Brain of the Firm*, McGraw-Hill.

Beer S. (1985), *Diagnosing the System for Organisations*, Wiley.

Berry A. J. Capps T. Cooper D. Hopper T. & Lowe E. A. (1986), "The Ethics of Research in a Public Enterprise", in Heller F. (ed), *The Use and Abuse of Social Science*, Sage.

Bjorn-Anderson N. (1988), "Are Human Factors Human", *Comp. J.* 31 (5), pp 386 - 390.

Bowey A. M. (1980), "Approaches to Organization Theory", in Lockett M. & Spear R. (eds), *Organizations as Systems*, OU Press.

Burrell G. & Morgan G. (1979), *Sociological Paradigms and Organisational Analysis*, Heinemann.

Carter P. Jackson M. C. Jackson N. & Keys P. (1987), "Community OR at Hull University", *Dragon* 2 (2).

Carter R. Martin J. Mayblin B. & Munday M. (1984), *Systems Management and Change*, Harper & Row.

Checkland P. B. & Scholes J. (1990), *Soft Systems Methodology in Action* Wiley.

Checkland P. B. (1981), *Systems Thinking Systems Practice*, Wiley.

Checkland P. B. (1982), "Soft Systems Methodology: A Reply to M. C. Jackson", *Jnl. Appl. Sys. Anal.* 9, pp 37 - 39.

Checkland P. B. (1983), "OR and the Systems Movement: Mappings and Conflicts", *J. Opl. Res. Soc.* 34, pp 661 - 75.

Checkland P. B. (1985), "From Optimizing to Learning: A Development of Systems Thinking for the 1990's", *J. Opl. Res. Soc.* 36 (9), pp 757 - 767.

Chen P. P. (1976), "The Entity-Relationship Model - Towards a Unified View of Data", *ACM TODS* 1 (1).

Child J. (1987), "Managerial Strategies: New Technology and the Labour Process", in Finnegan R. Salaman G. & Thompson K. (eds), *Information*

Technology: Social Issues, Hodder & Stoughton.

Churchman C. W. Ackoff R. L. & Arnoff E. L. (1957), *Introduction to Operations Research*, Wiley.

Churchman C. W. (1979), *The Design of Enquiring Systems*, Basic Books.

Clark A. W. (1976), (ed) *Experimenting with Organisational Life*, Plenum.

Codd E. F. (1970), "A Relational Model of Data for Large Shared Data Banks", *Comm. ACM* 13, pp 377 - 387.

Cohen B. Harwood W. T. & Jackson M. I. (1986), *The Specification of Complex Systems*, Addison-Wesley.

Cross N. & Roy R. (1975), *T262 Man-made Futures: Design and Technology, Units 13 - 16, Design Methods Manual*, OU Press.

Cross N. (1982a), "Designerly Ways of Knowing", *Design Studies* 3 (4), pp 221 - 227

Cross N. (1982b), "Design Education for Laypeople", in Evans B., Powell J. & Talbot R. (eds), *Changing Design*, Wiley.

Cross N. (1984) (ed), *Developments in Design Methodology*, Wiley.

Darke J. (1984), "The Primary Generator and the Design Process", in Cross (1984).

Davis G. B. & Olson M. (1985), *Management Information Systems*, McGraw-Hill.

De Marco T. (1979), *Structured Analysis and System Specification*, Prentice Hall.

De Neufville R. & Stafford J. H. (1974), *Systems Analysis for Engineers and Managers*, McGraw-Hill.

Dickson D. (1974), *Alternative Technology*, Fontana.

Eason K. D. (1984), "Towards the Experimental Study of Usability", *Beh. and Inf. Tech.* 3 (2), pp 133 - 143.

Eden C. (1987), "Problem-Solving or Problem-finishing", in M. C. Jackson & Keys P. (eds), *New Directions in Management Science*, Gower.

Eden C. Jones S. & Sims D. (1983), *Messing About in Problems*, Pergamon.

Emery F. E. & Trist E. L. (1960), "Socio-technical Systems", in Churchman C. W. & Verhulst M. (eds), *Management Science Models and Techniques*, Vol 2, Pergamon.

Espejo R. (1987), "From Machines to People and Organisations: A Cybernetic Insight on Management", in Jackson M. C. & Keys P. (eds), *New Directions in Management Science*, Gower.

Flood R. L. & Carson E. R. (1988), *Dealing With Complexity*, Plenum.

Flood R. L. (1989), "Six scenarios for the future of systems 'problem solving'", *Systems Practice* 2, pp 75 - 99.

Floyd C. & Keil R. (1987), "Adapting Software Development for System Design with the User", in Galliers R (ed), *Information Analysis - Selected Readings*, Addison-Wesley.

Foster M. (1972), "An introduction to the theory and practice of action research in work organisations", *Human Relations*, 25 (6).

Gane C. & Sarson T. (1979), *Structured Systems Analysis: Tools and Techniques*, Prentice Hall.

Geuss R. (1981), *The Idea of a Critical Theory*, Cambridge University

Press.

Habermas J. (1972), *Knowledge and Human Interests*, Heinemann.

Habermas J. (1974), *Theory and Practice*, Heinemann.

Habermas J. (1979), *Communication and the Evolution of Society*, Heinemann.

Harker S. D. P. & Eason K. D. (1984), "Representing the User in the Design Process", *Design Studies* 5 (2), pp 79 - 85.

Harker S. D. P. & Eason K. D. (1985), "Task Analysis and the Definition of User Needs", in *Proceedings of the IFAC Conference 'Man-Machine Systems'*, Sept. 1985.

Heller F. (1986), "Introduction and Overview", in Heller F. (ed), *The Use and Abuse of Social Science*, Sage.

Howe D. R. (1983), *Data Analysis for Data Base Design*, Arnold.

Hutt A. Donnelly N. Macaulay L. Fowler C. & Twigger D. (1987), "Describing a Product Opportunity: A Method of Understanding the User's Environment", in Diaper D. & Winder R. (eds), *People and Computers III*, Cambridge University Press.

Jackson M. C. & Keys P. (1984), "Towards a System of Systems Methodologies", *J. Opl. Res. Soc.* 35, pp 473 - 86.

Jackson M. C. (1982), "The Nature of Soft Systems Thinking: the Work of Churchman, Ackoff and Checkland", *Jnl. Appl. Sys. Anal.* 9, pp 17 - 29.

Jackson M. C. (1983), "The Nature of Soft Systems Thinking: Comment on the Three Replies", *Jnl. Appl. Sys. Anal.* 10, pp 109 - 113.

Jackson M. C. (1987a), "New Directions in Management Science", in Jackson M. C. & Keys P. (eds), *New Directions in Management Science*, Gower.

Jackson M. C. (1987b), "Community Operational Research: Purposes, Theory and Practice", *Dragon* 2 (2), pp 47 - 73.

Jackson M. C. (1987c), "Systems strategies for information management in organisations which are not machines", *Int. J. Inf. Mgt.* 7, pp 187 - 195.

Jackson M. C. (1987d), "Present Positions and Future Prospects in Management Science", *Omega* 15 (6), pp 455 - 466.

Jackson M. C. (1990), "Beyond a System of Systems Methodologies", *J. Opl. Res. Soc.*, 41, p 657.

Jacques R. & Talbot R. (1975), *T262 Instruction Manual*, OU Press.

Jacques R. (1982), "Changing Assumptions about Design Problems", in Evans B., Powell J. & Talbot R. (eds), *Changing Design*, Wiley.

Jones J. C. (1980), *Design Methods*, Pitman.

Karapin R. S. (1986), "What's the use of social science? - A review of the literature", in Heller F. (ed), *The Use and Abuse of Social Science*, Sage Publications.

Kast F. E. & Rosenzweig J. R. (1985), *Organization and Management: A Systems and Contingency Approach*, McGraw-Hill.

Keys P. (1988), "A Methodology for Methodology Choice", *Systems Research* 5 (1), pp 65 - 76.

Kuhn T. (1962), *The Structure of Scientific Revolutions*, University of Chicago Press.

Lamb D. A. (1988), *Software Engineering: Planning for Change*, Prentice Hall.

Langefors B. (1973), *Theoretical Analysis of Information Systems*, Auerbach.

Lehman M. M. (1980), *Programs, Programming and the Software Life-Cycle*, Dept. of Computing and Control, Imperial College, University of London.

Lewin K. (1947), "Group Decision and Social Change", in Newcomb T. M. & Hartley E. L. (eds), *Readings in Social Psychology*, Holt, Rinehart & Winston.

Lilienfeld R. (1978), *The Rise of Systems Theory*, Wiley.

Long J. (1986), "Designing for Usability", in Harrison M. D. & Monk A. F. (eds), *People and Computers: Designing for Usability*, Cambridge University Press.

Lundeberg M. Goldkuhl G. & Nilsson A. (1981), *Information Systems Development - A Systematic Approach*, Prentice Hall.

Lyytinen K. (1987), "Information Systems Development: Theoretical Constructs and Recommendation", in Boland R. J. & Hirschheim R. A. (eds), *Critical Issues in Information Systems Research*, Wiley

Macro A. & Buxton J. (1987), *The Craft of Software Engineering*, Addison-Wesley.

Mansell G. J. (1989), "System Design and Social Science", in Flood R. L. Jackson M. C. & Keys P. *Systems Prospects*, Plenum.

March L. J. (1984), "The Logic of Design", in Cross (1984).

Mumford E. (1983), *Designing Human Systems*, Manchester Business School.

Nissen H. E. (1984), "Acquiring Knowledge of IS", in Mumford E. Hirschheim G. Fitzgerald G. & Wood-Harper A. T., *Research Methods in Information Systems*, North Holland.

Popper K. (1959), *The Logic of Scientific Discovery*, Harper & Row.

Oliga J. C. (1988), "Methodological foundations of systems methodologies", *Systems Practice*, 1, pp 87 - 112.

Pressman R. (1987), *Software Engineering - A Practitioners Approach*, McGraw-Hill.

Rapoport R. N. (1970), "Three Dilemmas in Action Research", *Human Relations*, 23 (6) pp 499 - 513.

Rittel H. W. J. & Webber M. M. (1981), "Dilemmas in a General Theory of Planning", in Emery F. E. (ed), *Systems Thinking Vol. 2*, Penguin.

Rivett B. H. P. & Ackoff R. L. (1963), *A Manager's Guide to OR*, Wiley.

Ross D. T. & Schoman K. E. (1977), "Structured Analysis for Requirements Definition", *IEEE Trans. on Software Engineering* SE-3 (1).

Ross D. T. (1977), "Structured Analysis - A Language for Communicating Ideas", *IEEE Trans. on Software Engineering*, SE-3 (1) pp 16 - 34.

Roy R. (1975), *Design Project Guide*, OU Press.

Roy R. (1975), *T262 Man-made Futures: Design and Technology, Unit 12, Design Project Guide*, OU Press.

Silverman D. (1970), *The Theory of Organizations*, Heinemann.

Simon H. A. (1969), *The Sciences of the Artificial* MIT Press.

Sommerville I. (1982), *Software Engineering*, Addison-Wesley.

Susman G. & Evered R. D. (1978), "An assessment of the scientific merits of action research", *Administrative Science Quarterly*, 23, December.

Ulrich W. (1983), *Critical Heuristics of Social Planning*, Berne: Haupt.

Willcocks L. & Mason D. (1987), *Computerising Work*, Paradigm.

Willmott H. (1989), "O.R. as a Problem Situation: From Soft Systems Methodology to Critical Science", in Jackson M. C. Keys P. & Cropper S. A. (eds), *Operational Research and the Social Sciences*, Plenum.

Wilson B. (1984), *Systems: Concepts, Methodologies and Applications*, Wiley.

Wood-Harper A. T. Antill L. & Avison D. E. (1985), *Information Systems Definition: The Multiview Approach*, Blackwell Scientific Publications.

Yourdon E. & Constantine L. (1979), *Structured Design*, Prentice Hall.